DASAR PEMROGRAMAN

Institut Teknologi Padang

PENGANTAR

Tujuan umum matakuliah:

- Membekali mahasiswa cara berpikir dalam pemecahan persoalan dengan menggunakan beberapa paradigma pemrograman, kemudian mahasiswa memiliki kemampuan membuat menyelesaikan masalah pemrograman tanpa tergantung pada bahasa pemrograman apapun.
- Membekali mahasiswa dengan modul dasar dari algoritma yang sering dipakai dalam pemrograman, termasuk dalam mengeksekusi program tersebut dengan salah satu bahasa pemrograman yang sederhana, sebagai alat untuk mengeksekusi program dengan mesin yang tersedia.

PENGANTAR

Tujuan khusus mata kuliah:

Mahasiswa diharapkan mampu untuk :

- Memecahkan masalah dengan beberapa paradigma pemrograman dan menuliskan spesifikasi dan algoritmanya tanpa tergantung pada bahasa pemrograman apapun.
- Menulis algoritma dari suatu masalah dengan menggunakan metodologi dan skema standard yang terstruktur.
- Menulis program yang baik sesuai dengan kriteria dalam bahasa pemrograman yang ada, dengan menggunakan aturan translasi yang diperkenankan.
- Menghasilkan program yang terstruktur walaupun bahasa pemrogramannya bukan bahasa yang terstruktur.
- Menuliskan dan menerjemahkan penyelesaian algoritmik untuk beberapa persoalan menjadi program yang dapat dieksekusi oleh mesin dalam salah satu bahasa tingkat tinggi yang biasa digunakan.

KEPUSTAKAAN

- Aho, Hopcroft, Ullman, "Data Structures and Algorithms", Prentice Hall, 1987.
- Knuth, D.E., "The Art of Computer Programming", Vol. 1 : "Fundamentals Algoritms", Addison Wisley, 1968.
- Sedgewick R., "Algorithms", Addison Wisley, 1984.
- Wirth, N., "Algorithms & Data Stuctures", Prentice Hall, 1986.
- Munir, R dan Lidya, L. 2001. *Algoritma dan Pemrograman Dalam Bahasa Pascal dan C*. Bandung: Informatika.
- Kadir, A dan Heriyanto. 2005. *Algoritma Pemrograman Menggunakan C++*. Yogyakarta: Penerbit Andi.
- Pranata, A. 2005. *Algoritma dan Pemrograman*. Yogyakarta: Penerbit Graha Ilmu.
- P.J. Deitel, H.M. Deitel, “C How to Program”, Pearson International Edition Fifth Edition, 2007.
- Stephen Prata, “C Primer Plus”, SamsPublishing Fifth Edition, 2005.
- Fathul Wahid, “Dasar-Dasar Algoritma & Pemrograman”, PenerbitAndi, Yogyakarta, 2004.
- Ellis Horowitz, Sartaj Sahni, Sanguthevar Rajasekaran. “Computer Algorithms / C++, Computer Science Press. 1998.
- Thomas H Cormen, Charles E Leiserson, Ronald L. “Introduction to Algorithms”,2nd Edition. The MIT Press. New York. 1990.
- Robert Setiadi. “Algoritma Itu Mudah”, PT Prima Infosarana Media, Kelompok Gramedia. Jakarta. 2008

SISTEM PERKULIAHAN

1. Tugas Mandiri 20%
2. UTS 30%
3. UAS 50%

Syarat kehadiran adalah 80% dari total keseluruhan perkuliahan.

PENGERTIAN

- Pemrograman berkaitan dengan komputer, yang digunakan untuk membantu menyelesaikan persoalan.
- Strategi penyelesaian masalah oleh komputer mesti ditanamkan pada mesin tersebut oleh manusia melalui suatu program oleh suatu bahasa pemrograman.
- Untuk menghasilkan program, manusia mesti mengunakan paradigma yang memiliki prioritas dan keterbatasan.

PENGERTIAN

- Program merupakan pernyatan yang disusun menjadi satu kesatuan prosedur yang berupa urutan langkah yang disusun secara logis dan sistematis untuk menyelesaikan masalah.
- Bahasa pemrograman adalah prosedur penulisan program, umumnya terdiri dari 3 faktor utama:
  - Sintaks, merupakan aturan penulisan bahasa pemrograman.
  - Semantik, adalah arti atau maksud yang terkandung pada statemen.
  - Kebenaran logika, adalah berhubungan benar tidaknya urutan statemen.

PENGERTIAN

- Produk yang dihasilkan oleh seorang pemrogram adalah program dengan rancangan benar (metodologis & sistematis) yang dapat dieksekusi oleh mesin, berfungsi dengan benar, sanggup melayani segala kemungkinan masukan dan didukung dengan adanya dokumentasi.

PENGERTIAN

- Tujuan pengajaran pemrograman membentuk mahasiswa menjadi perancang (designer) program.
- Sedangkan tujuan pengajaran bahasa program membentuk mahasiswa menjadi juru kode (coder).

PENGERTIAN

- Pada prakteknya, suatu rancangan harus dapat di-kode, untuk dieksekusi oleh mesin. Sehingga belajar pemrograman dan bahasa program saling berkaitan satu sama lain.
- Metoda terbaik untuk pengajaran tersebut melalui contoh nyata program yang merupakan pola solusi, sehingga dapat melihat, mengalami sendiri dan melakukannya.

PENGERTIAN

- Belajar memrogram dan belajar bahasa program mempunyai tingkatan kesulitan yang berbeda.
- Belajar memrogram lebih bersifat pemahaman persoalan, analisis dan sintesis, yang merupakan belajar tentang strategi pemecahan masalah, metodologi dan sistematika pemecahan masalah tersebut, kemudian menuangkannya dalam suatu notasi yang disepakati bersama.

PENGERTIAN

- Belajar bahasa program adalah bagaimana cara memakai suatu bahasa, aturan sintaks (tatabahasa), setiap instruksi yang ada dan tata cara pengoperasian kompilator atau interpreter bahasa yang bersangkutan pada mesin tertentu.
- Belajar bahasa program lebih kepada ketrampilan dari pada analisis dan sintesis, yang memanfaatkan instruksi-instruksi dan kiat atau cara yang dapat dipakai secara spesifik hanya pada bahasa yang digunakan tersebut.

PENGERTIAN

- **Bahasa Mesin**, merupakan bahasa pemograman pada mesin yang hanya mengenal 2 keadaan yang berlawanan yaitu keadaan 0 dan 1 atau benar atau salah.
- **Bahasa Tingkat Rendah**, atau dikenal juga Mnemonics (pembantu untuk mengingat)
- **Bahasa Assembler**
- **Bahasa Tingkat Tinggi**
- **Bahasa generasi ke 4 (4GL)**

PENGERTIAN

- Bahasa pemrograman berkembang dengan cepat sejak tahun enampuluhan, seringkali dianalogikan dengan menara Babel yang berakibat manusia menjadi tidak lagi saling mengerti bahasa masing-masing.

PENGERTIAN

Paradigma dalam pemrograman:

- Prosedural
- Fungsional
- Deklaratif
- Berorientasi Objek
- Konkuren
- Relasional

Untuk setiap paradigma, tersedia bahasa pemrograman yang mempermudah implementasi rancangan penyelesaian masalahnya.

PENGERTIAN

- Contoh bahasa-bahasa pemrograman yang ada :
  - 1. Prosedural : Algol, Pascal, Fortran, Basic, Cobol, C ...
  - 2. Fungsional : LOGO, APL, LISP
  - 3. Deklaratif/Lojik : Prolog
  - 4. Object oriented murni: Smalltalk, Eifel, Jaca, C++..
  - 5. Konkuren : OCCAM, Ada, Java
  - 6. Relasional: SQL pada basisdata relasional
- Paradigma Objek mulai ditambahkan pada bahasa-bahasa yang ada. Pemroses bahasa Pascal dan C versi terbaru dilengkapi dengan fasilitas terorientasi objek, misalnya Turbo Pascal (mulai versi 5.5) pada komputer pribadi (PC) dan C++. Ada beberapa versi LISP dan Prolog yang juga memasukkan aspek OO.
- Suatu program dalam bahasa pemrograman tertentu akan diproses oleh pemroses bahasanya. Ada dua kategori pemroses bahasa, yaitu kompilator dan interpreter.
- Dalam melakukan implementasi program, tersedia bahasa pemrograman visual atau tekstual.

Paradigma Pemrograman Prosedural

- Paradigma ini didasari oleh konsep mesin Van Neumann, dimana sekelompok tempat penyimpanan (Memori), yang dibedakan menjadi memori instruksi dan memori data, masing-masing dapat diberi nama dan harga
- Instruksi akan dieksekusi satu per satu secara sekuensial oleh sebuah pemroses tunggal
- Data diperiksa dan dimodifikasi secara sekuensial juga

Paradigma Pemrograman Prosedural

- Program dalam paradigma ini didasari pada strukturisasi informasi dalam memori dan manipulasi dari informasi yang disimpan tersebut
- Program = Algoritma + Struktur Data
- Pemrograman pada paradigma ini mesti berpikir dalam batasan mesin namun efisien dalam eksekusi

Paradigma Pemrograman Fungsional

- Paradigma ini didasari oleh konsep pemetaan dan fungsi pada matematika
- Disini tidak dipermasalahkan memorisasi dan struktur data, tidak ada pemisahan antara data dan program maupun pengertian tentang variabel
- Semua kelakuan program adalah mata rantai transformasi dari sebuah keadaan awal menuju ke suatu rantai keadaan akhir, yang mungkin melalui keadaan antara maupun melalui aplikasi fungsi

Paradigma Pemrograman Fungsional

- Pada pemrograman fungsional, pemrogram tidak perlu lagi mengetahui bagaimana mesin mengeksekusi atau bagaimana informasi disimpan pada memori, namun hanya memperhatikan keadaan awal dan akhir saja
- Dibandingakan pemrograman prosedural, pemrograman fungsional memiliki kelemahan pada efisiensi dan kinerjanya

Paradigma Pemrograman Deklaratif

- Paradigma ini didasari oleh pendefinisian relasi antar individu yang dinyatakan sebagai predikat
- Pemrograman ini menguraikan sekumpulan fakta dan aturan-aturan, ketika program dieksekusi, pemakai mengajukan pertanyaan danprogram akan menjawab, apakah pernyataan itu dapat dideduksi dari aturan dan fakta yang ada

Paradigma Berorientasi Objek

- Paradigma ini didasari oleh Kelas dan Objek
- Paradigma ini menawarkan konsep modularitas, penggunaan kembali, dan kemudahan modifikasi

Paradigma Konkuren

- Paradigma ini didasari oleh kenyataan bahwa dalam keadaan nyata, sebuah sistem komputer harus menangani beberapa program (task) yang harus dieksekusi bersama dalam sebuah lingkungan
- Paradigma konkuren, yang erat hubungannya dengan arsitektur perangkat keras yang memungkinkan pemrosesan secara paralel atau perangkat lunak sistem terdistribusi yang mengelola akses konkuren

Paradigma Relasional

- Paradigma relasional, yang didasari entity dan relasi, dan pemrograman dalam bahasa Query yang memungkinkan diperolehnya suatu himpunan nilai.

APA ITU ALGORITMA?

- Cara menyelesaikan suatu proses
- Terdiri atas langkah-langkah yang terdefinisi dengan baik
- Bisa dihitung (*computable*) atau bisa diukur (*measurable*)
- Menerima input, menghasilkan output

- An algorithm is a sequence of computational steps that transform the input into the output
- An algorithm is a tool for solving a well-specified computational problem

DEFINISI ALGORITMA MENURUT PAKAR

- Menurut Abu Ja'far Mohammad Ibn Musa Al Khoarismi:

  "Suatu metode khusus untuk menyelesaikan suatu persoalan."

- Menurut Goodman Hedet Niemi:

  "Urut-urutan terbatas dari operasi-operasi terdefinisi dengan baik, yang masing-masing membu-tuhkan memory dan waktu yang terbatas untuk menyelesaikan suatu masalah."

ALGORITMA

- Algoritma adalah cara yang dapat ditempuh oleh komputer dalam mencapai suatu tujuan, terdiri atas langkah-langkah yang terdefinisi dengan baik, menerima input, melakukan proses, dan menghasilkan output. Meskipun tidak selalu, biasanya sebuah algoritma memiliki sifat bisa dihitung (*computable*) atau bisa diukur (*measurable*).

SYARAT CORRECTNESS

- Sebuah algoritma dikatakan BENAR (*correct*) jika algoritma tersebut berhasil mengeluarkan output yang benar untuk semua kemungkinan input.

- Bagaimana dengan 99% benar?
  - 99% benar artinya sebuah algoritma adalah SALAH (*incorrect*)

PSEUDOCODE

- Adalah cara untuk menuliskan sebuah algoritma secara *high-level*
- Biasanya dituliskan dengan kombinasi bahasa Inggris dan notasi matematika
  - Lebih terstruktur daripada bahasa Inggris biasa
  - Tidak detil dibanding program

- Isu-isu detil dalam program yang sifatnya teknis tidak dibahas dalam pseudocode

PSEUDOCODE

- **Kode-palsu** atau dalam bahasa inggris lebih dikenal sebagai **pseudocode** merupakan deskripsi tingkat tinggi informal dan ringkas atas algoritma pemrograman komputer yang menggunakan konvensi struktural atas suatu bahasa pemrograman, dan ditujukan untuk dibaca oleh manusia dan bukan oleh mesin. Kode palsu biasanya tidak menggunakan elemen detail yang tidak diperlukan untuk kebutuhan pemahaman manusia atas suatu algoritma, seperti deklarasi variabel, kode ataupun subrutin untuk sistem yang bersifat spesifik

CONTOH PSEUDOCODE (1)

- Algoritma untuk menampilkan 7 buah simbol #

```
1 for i=1 to 7 do
2   display ”#”
3 end for
```

CONTOH PSEUDOCODE (2)

- Algoritma untuk menghitung Faktorial dari N

```
1 iTampung=1
2 for i=1 to N do
3   iTampung=iTampung*i
4 end for
5 display ”Faktorial dari ”,N,” adalah ”,iTampung,NL
```

CONTOH PSEUDOCODE (3)

- Algoritma untuk menampilkan 8 bilangan Fibonacci

```
1 f1=0
2 f2=1
3 for i=1 to 8 do
4   iFibo=f1+f2
5   display ”Angka ke-”,i,” adalah
  ”,iFibo,NL
6   f1=f2
7   f2=iFibo
8 end for
```

KOMPONEN PSEUDOCODE

- Variabel
  - Merupakan tempat penyimpanan sebuah nilai
- Perulangan (*loop*)
  - Teknik for-do
  - Teknik repeat-until
  - Teknik while-do
- Percabangan (*branch*)
  - Teknik if-then
  - Teknik select-case
- Modul
  - Procedure / Sub
  - Function
  - Teknik rekursif

VARIABEL

- Merupakan tempat penyimpanan sebuah nilai
- Memiliki nama
- Dapat dimasukkan sebuah nilai
- Dapat dipanggil nilainya
- Menampung tipe data tertentu
  - Numerik
  - Karakter
  - String

- Beberapa variabel yang tipenya sama dapat dirangkai menjadi sebuah array

TEKNIK FOR-DO

- Perulangan berdasar :
  - Variabel penentu perulangan
  - Batas bawah
  - Batas atas

- Contoh algoritma menampilkan N bilangan Fibonacci pertama (pseudocode 2.9b)

```
1 f1=0
2 f2=1
3 for i=1 to N do
4   iFibo=f1+f2
5   display ”Angka ke-”,i,” adalah ”,iFibo,NL
6   f1=f2
7   f2=iFibo
8 end for
```

TEKNIK REPEAT-UNTIL

- Perulangan berdasarkan kondisi (*true* atau *false*)
- Kondisi diperiksa di akhir perulangan
- Perulangan terjadi selama kondisi belum terpenuhi
- Minimal terjadi 1x perulangan

- Contoh algoritma menampilkan N bilangan Fibonacci pertama (pseudocode 2.9c)

```
 1 f1=0
 2 f2=1
 3 iFibo=f1+f2
 4 i=1
 5 repeat
 6   display ”Angka ke-”,i,” adalah ”,iFibo,NL
 7   i=i+1
 8   f1=f2
 9   f2=iFibo
10   iFibo=f1+f2
11 until i>N
```

TEKNIK WHILE-DO

- Perulangan berdasarkan kondisi (*true* atau *false*)
- Kondisi diperiksa di awal perulangan
- Perulangan terjadi selama kondisi masih terpenuhi
- Bisa terjadi 0 perulangan jika dari awal kondisi tidak terpenuhi

- Buatlah contoh algoritma menampilkan N bilangan Fibonacci pertama dengan menggunakan teknik while-do!

TEKNIK IF-THEN

- Percabangan berdasarkan kondisi (*true* atau *false*)
- Terdapat 2 segmen, akan dieksekusi berdasarkan kondisi

- Contoh algoritma dengan if-then (pseudocode 2.13b)

```
1 if iUmur >= 17 then
2   display ”Anda boleh masuk”,NL
3 else
4   display ”Maaf Anda tidak boleh masuk”,NL
5 end if
```

TEKNIK SELECT-CASE

- Percabangan berdasarkan nilai sebuah variabel ordinal
- Terdapat banyak segmen, akan dieksekusi berdasarkan kondisi

- Contoh algoritma dengan select-case (pseudocode 2.15)

```
1 display ”Masukkan jumlah sisi bangun : ”
2 read iSisi
3 switch iSisi
4   case 3 : display ”segitiga”,NL
5   case 4 : display ”kotak”,NL
6   case 5 : display ”segilima”,NL
7   else   : display ”nama bangun tdk terdaftar”,NL
8 end switch
```

NOTASI MATEMATIKA

Penjumlahan dan sifat-sifatnya

1. $a_1 + a_2 + ... + a_n = \sum_{k=1}^{n} a_k$

2. Sifat Linearitas $\sum_{k=1}^{n} (c\, a_k + b_k) = c \sum_{k=1}^{n} a_k + \sum_{k=1}^{n} b_k$

3. $\sum_{k=1}^{n} \theta(f(k)) = \theta\left(\sum_{k=1}^{n} f(k) \right)$

DERET

1. Deret Aritmatika

$$\sum_{k=1}^{n} k = 1 + 2 + ... + n = \frac{n(n+1)}{2} = \theta(n^2)$$

2. Deret Harmonis

$$H_n = 1 + \tfrac{1}{2} + \tfrac{1}{3} + \tfrac{1}{4} + ... + \tfrac{1}{n}$$

$$= \sum_{k=1}^{n} \tfrac{1}{k} = \ln n + 0(1)$$

LATIHAN

- Buatlah sebuah pseudocode untuk menampilkan N bilangan pertama secara terbalik.
  - contoh : 10, 9, 8, 7, 6, 5, 4, 3, 2, 1
  - Buat dengan menggunakan teknik for-do!
  - Buat dengan menggunakan teknik repeat-until!
  - Buat dengan menggunakan teknik while-do!

- Buat sebuah pseudocode yang akan menerima sebuah bilangan X dari user. Tampilkan pesan “benar” jika X habis dibagi 2, 3 atau 7 dan tampilkan “salah” jika tidak habis dibagi.

REVIEW

- Apa yang sudah dipahami?
- Apa yang akan dibahas selanjutnya?

Pertemuan 2

Dasar Pemrograman

Learning Outcomes:

Pada akhir pertemuan ini, diharapkan mahasiswa akan mampu :

- Menjelaskan definisi algoritma dan pembuatan algoritma

Outline Materi

Algoritma dan Pemrograman

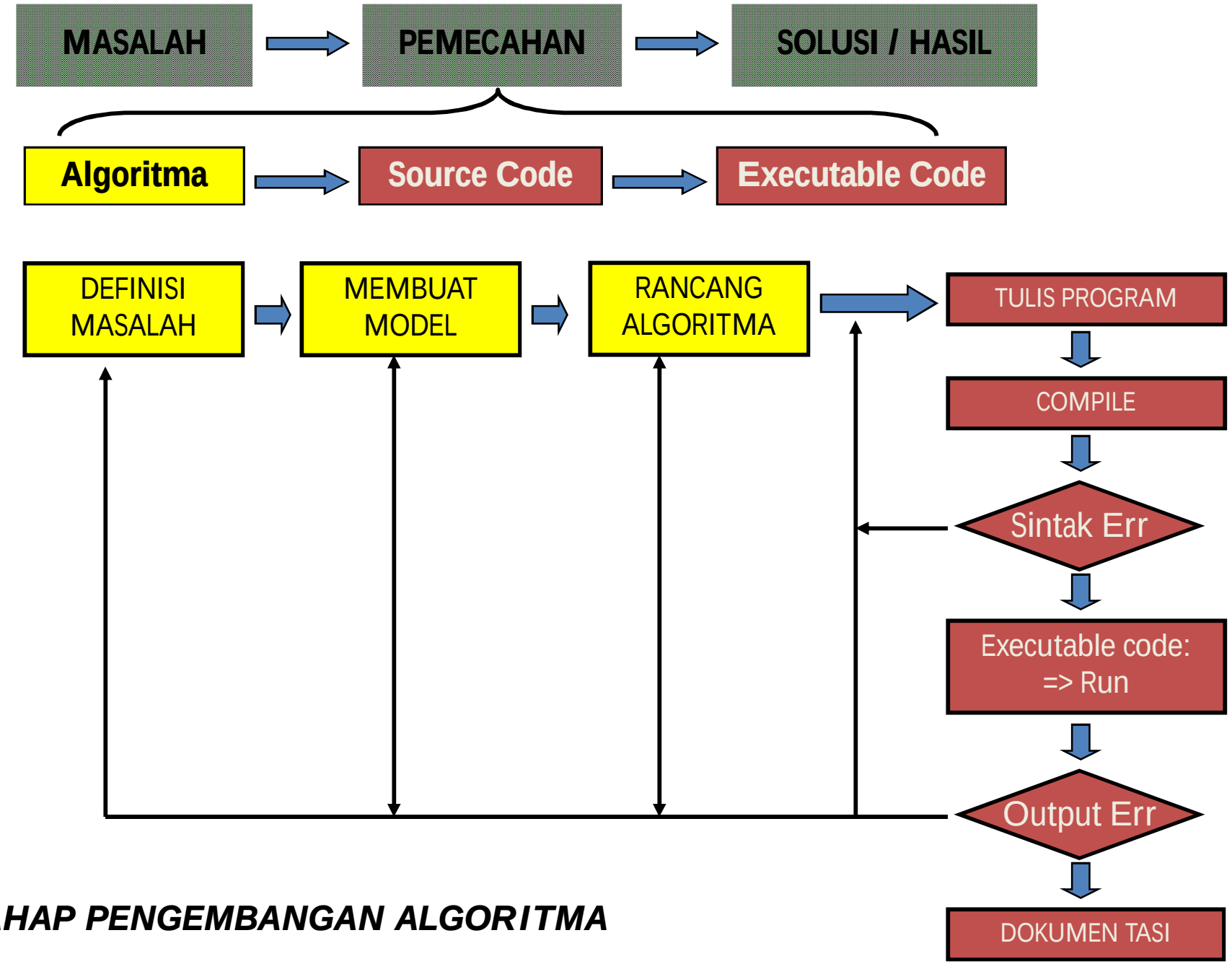
- **Definisi Algoritma**
- **Tahap Pengembangan algoritma**
- **Penyajian algoritma**
- **Menulis Speudocode**
- **Contoh Algoritma dengan Pseudocode**
- **Membuat Flow Chart**
- **Contoh Algoritma dengan Flow Chart**
- **Kriteria Algoritma Yang Baik**
- **Teorema Terstruktur**
- **Latihan**

Rujukan

- Brian W. Kernighan, Dennis M. Ritchie (1988). The C Programming Language, Second Edition, Prentice Hall
- Deitel, H.M. and Deitel, P.J. (2001). *C HOW TO PROGRAM*. 3rd edition. Prentice Hall, NJ.
- Gottfried, B.S. (1996). *Schaum's Outline Series. Theory and Problems of Programming with C*. McGraw Hill, NY.
- Ngoen. Th. S. (2004). *Pengantar Algoritma dengan Bahasa C*. Penerbit Salemba Teknika.
- Sedgewick, R. (1992). *Algorithms in C++*. Addison Wesley.

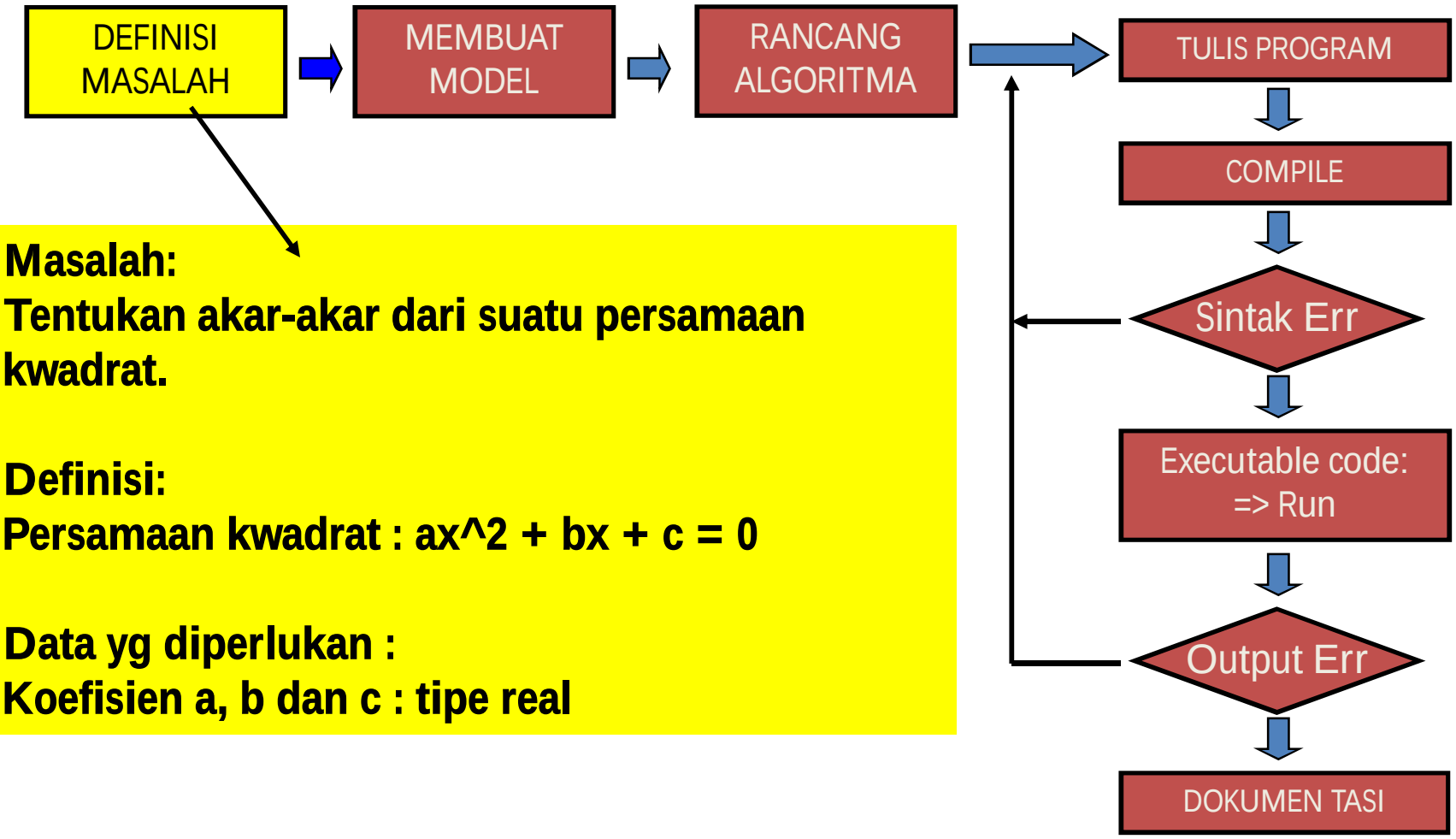
Definisi Algoritma

- Algoritma adalah sekumpulan langkah-langkah terbatas untuk mencari solusi suatu masalah.
- Berasal dari kata algoris dan ritmis. Awalnya diungkapkan oleh Al Khowarizmi.
- Di pemrograman, algoritma didefinisikan sebagai metode yang terdiri dari langkah-langkah terstuktur untuk mencari solusi suatu masalah dengan ban-tuan komputer.



TAHAP PENGEMBANGAN ALGORITMA

TAHAP PENGEMBANGAN ALGORITMA

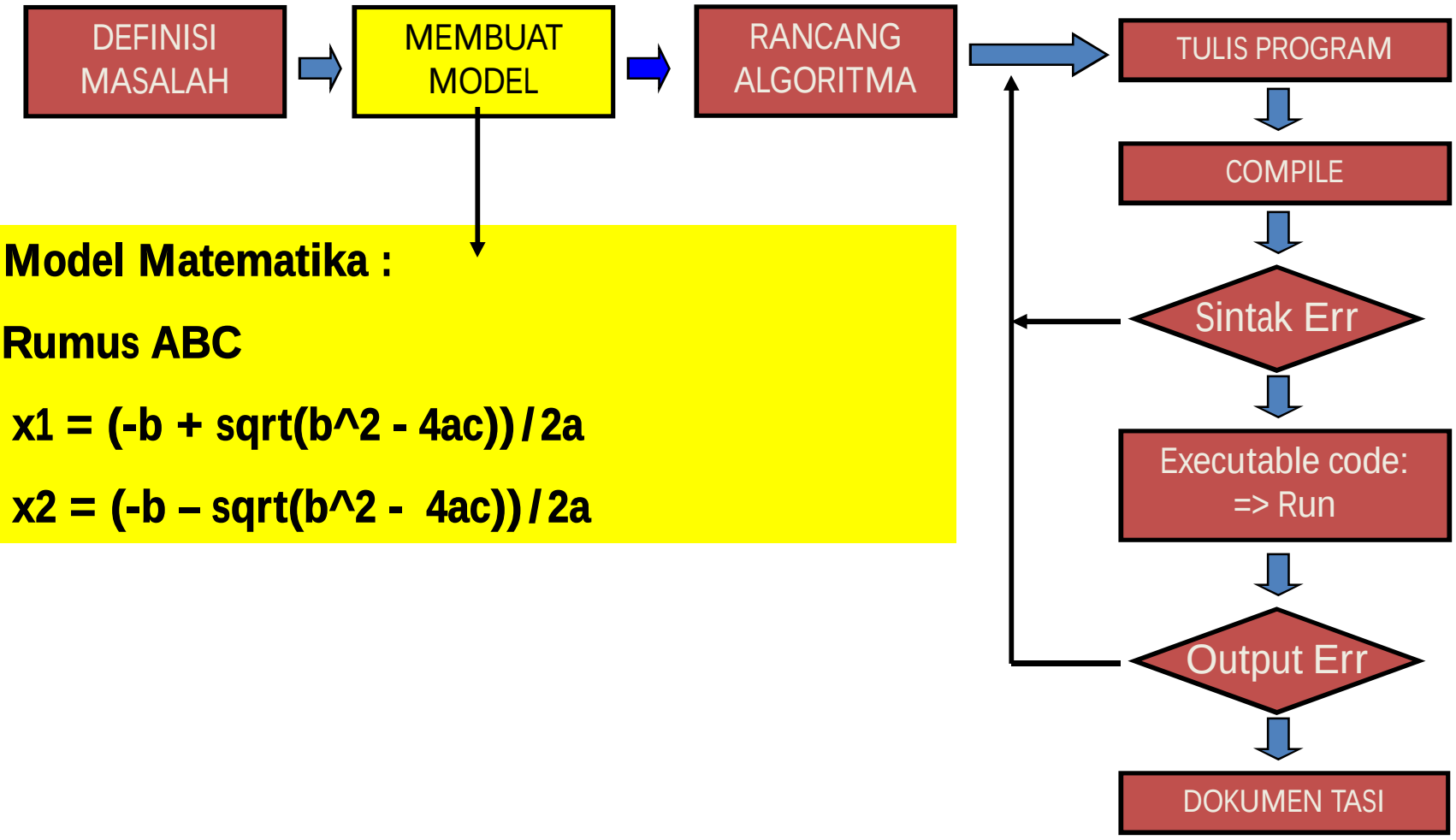


Masalah:
Tentukan akar-akar dari suatu persamaan kwadrat.

Definisi:
Persamaan kwadrat : ax^2 + bx + c = 0

Data yg diperlukan :
Koefisien a, b dan c : tipe real

TAHAP PENGEMBANGAN ALGORITMA

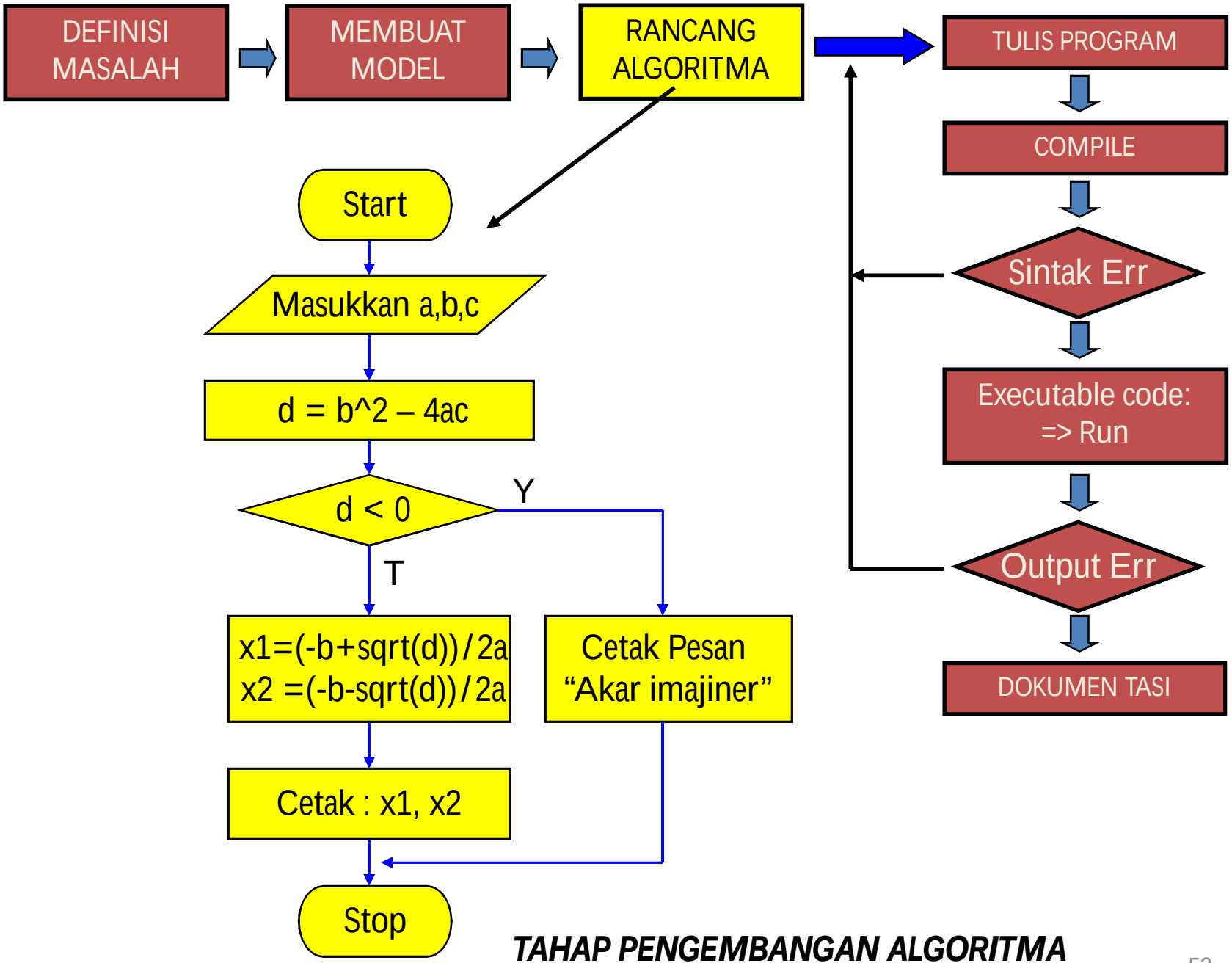


Model Matematika :

Rumus ABC

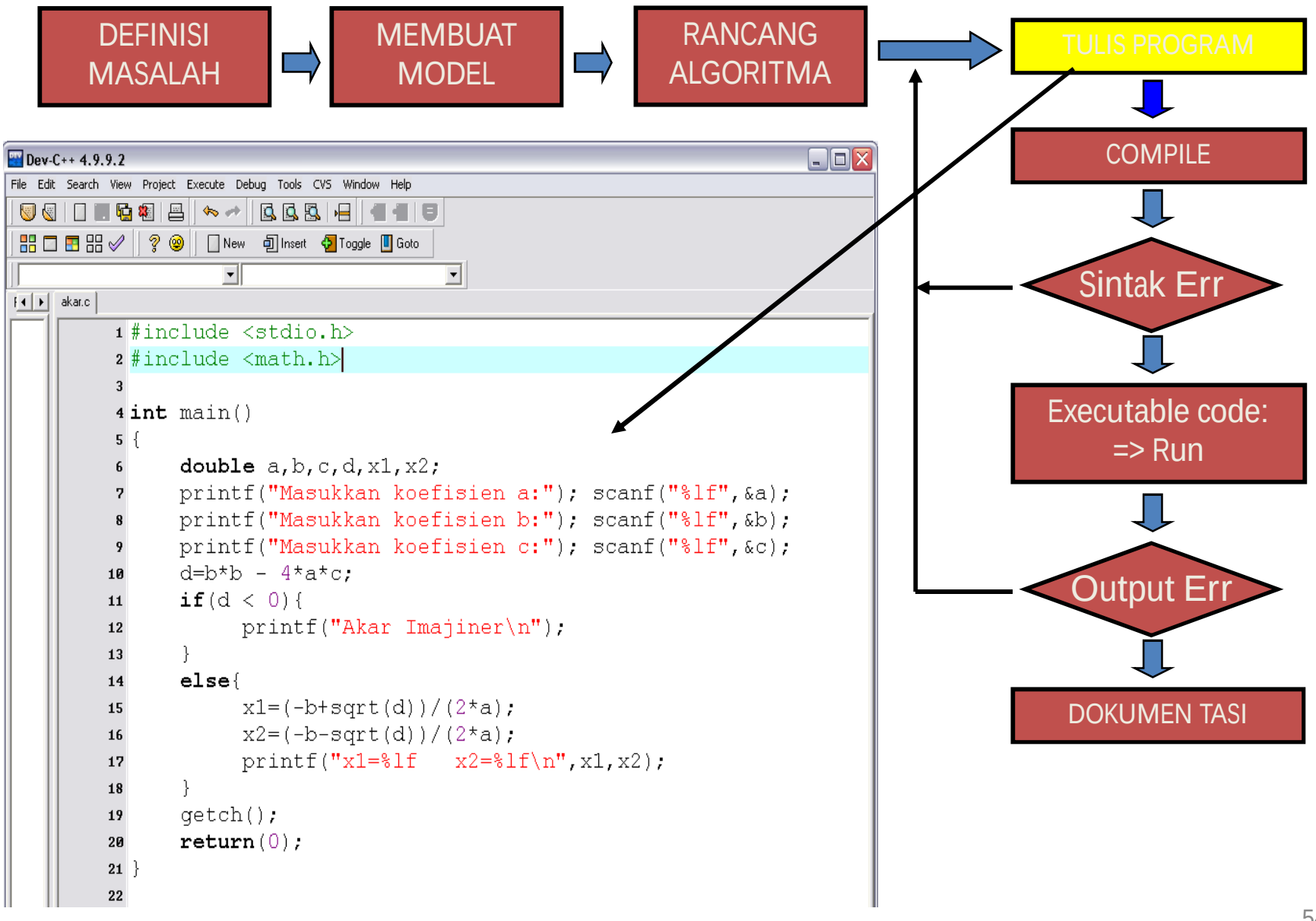
x1 = (-b + sqrt(b^2 - 4ac)) / 2a

x2 = (-b – sqrt(b^2 - 4ac)) / 2a



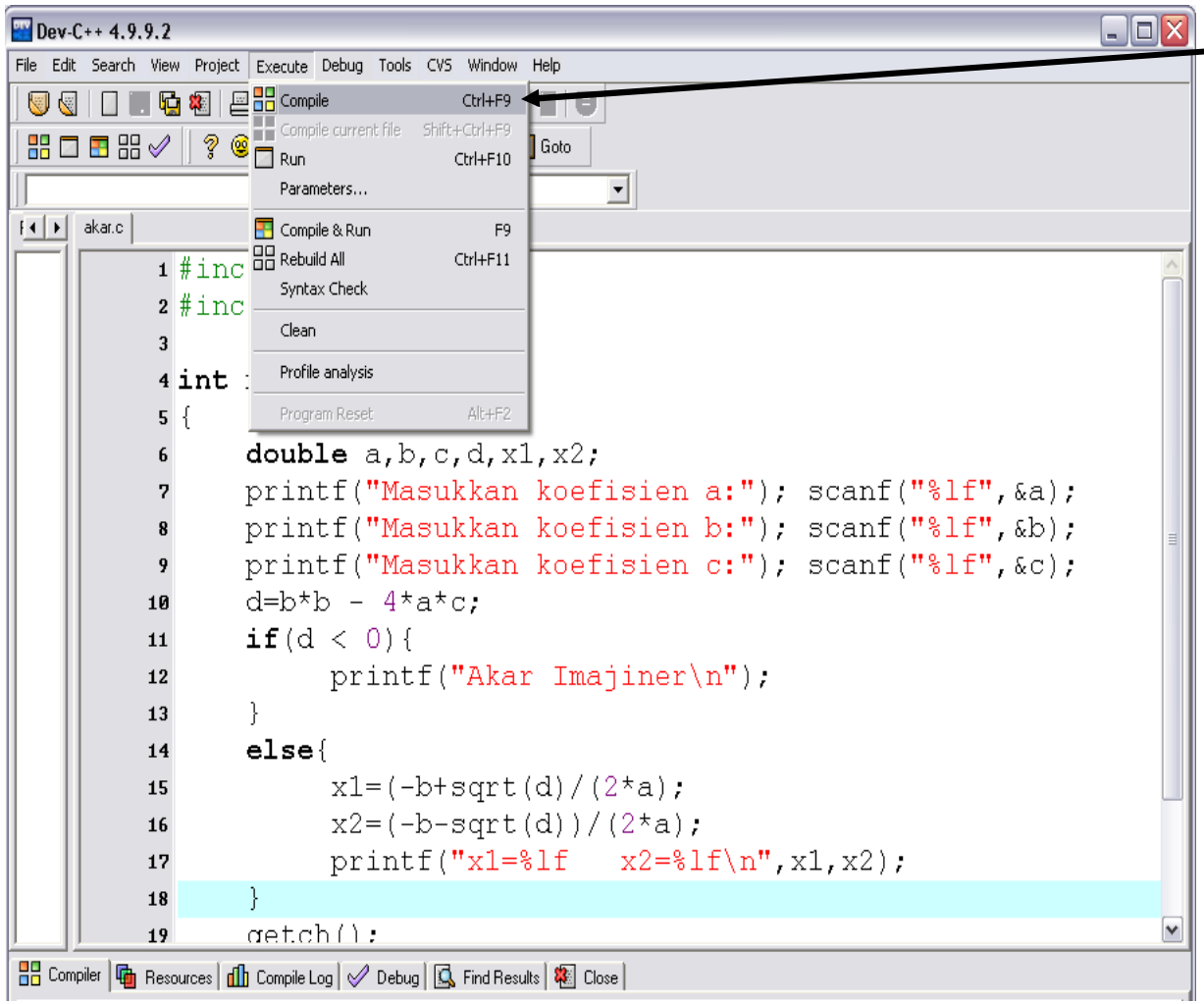
TAHAP PENGEMBANGAN ALGORITMA

TAHAP PENGEMBANGAN ALGORITMA



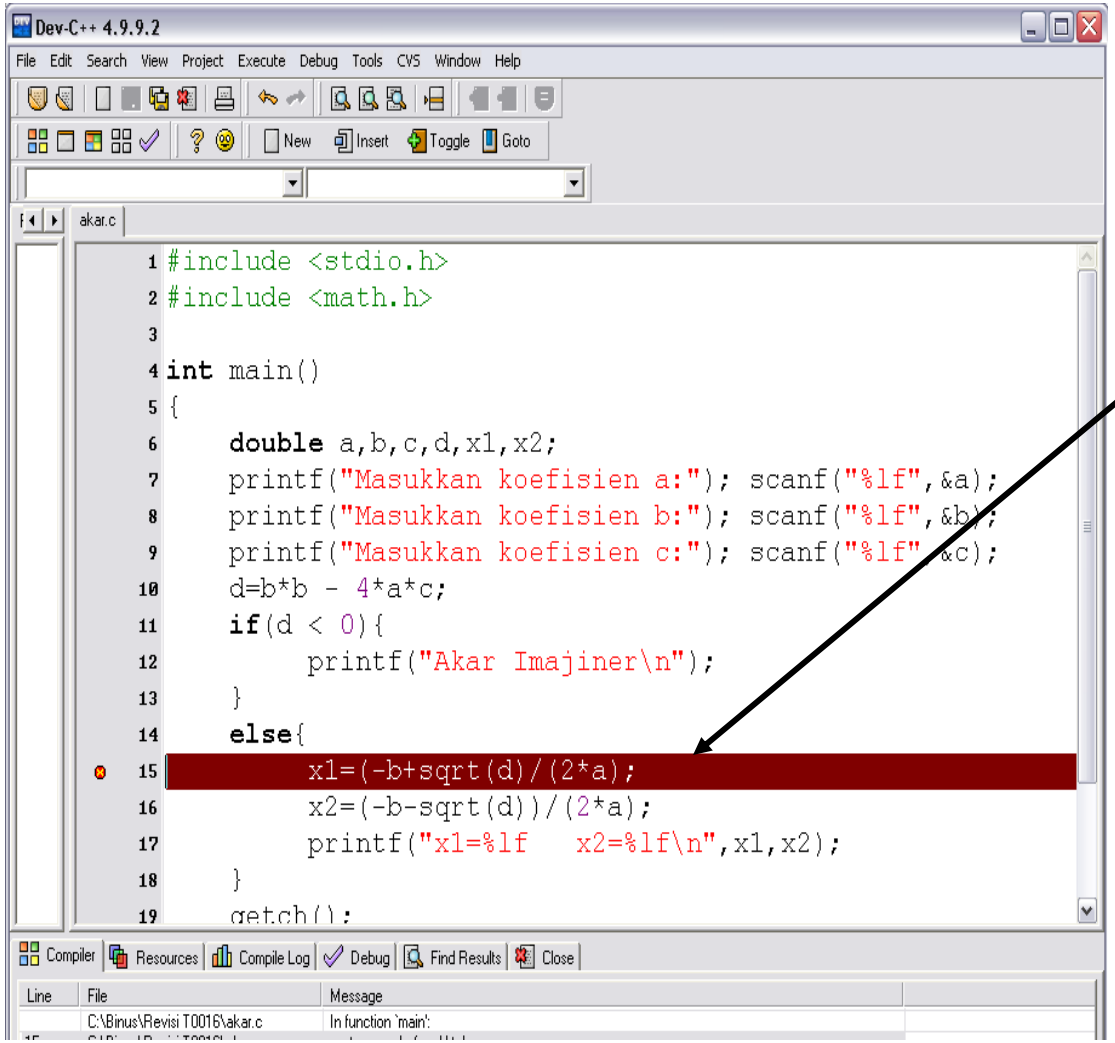
TAHAP PENGEMBANGAN ALGORITMA

DEFINISI MASALAH → MEMBUAT MODEL → RANCANG ALGORITMA → TULIS PROGRAM → COMPILE → Sintak Err → Executable code: => Run → Output Err → DOKUMEN TASI

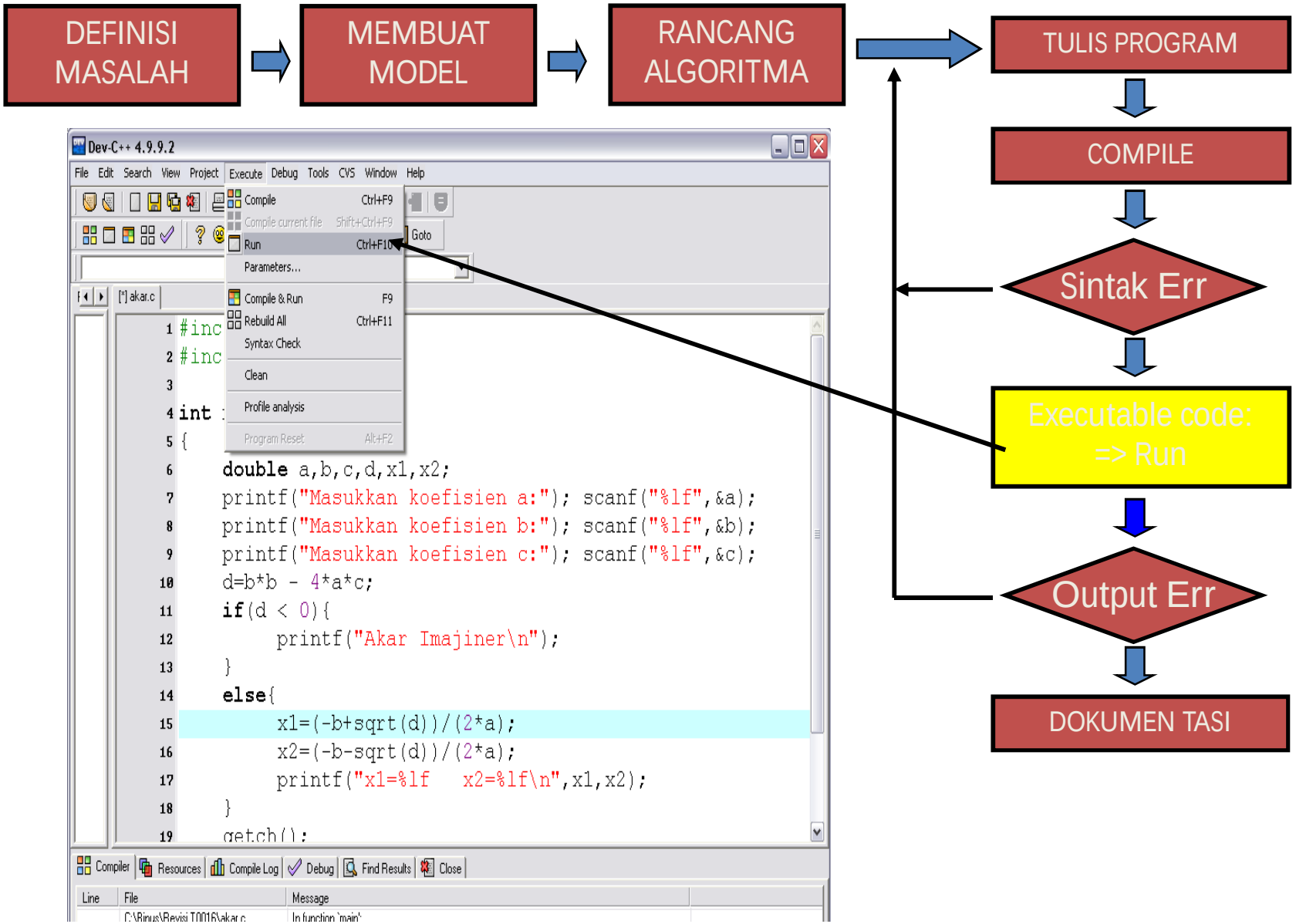


TAHAP PENGEMBANGAN ALGORITMA

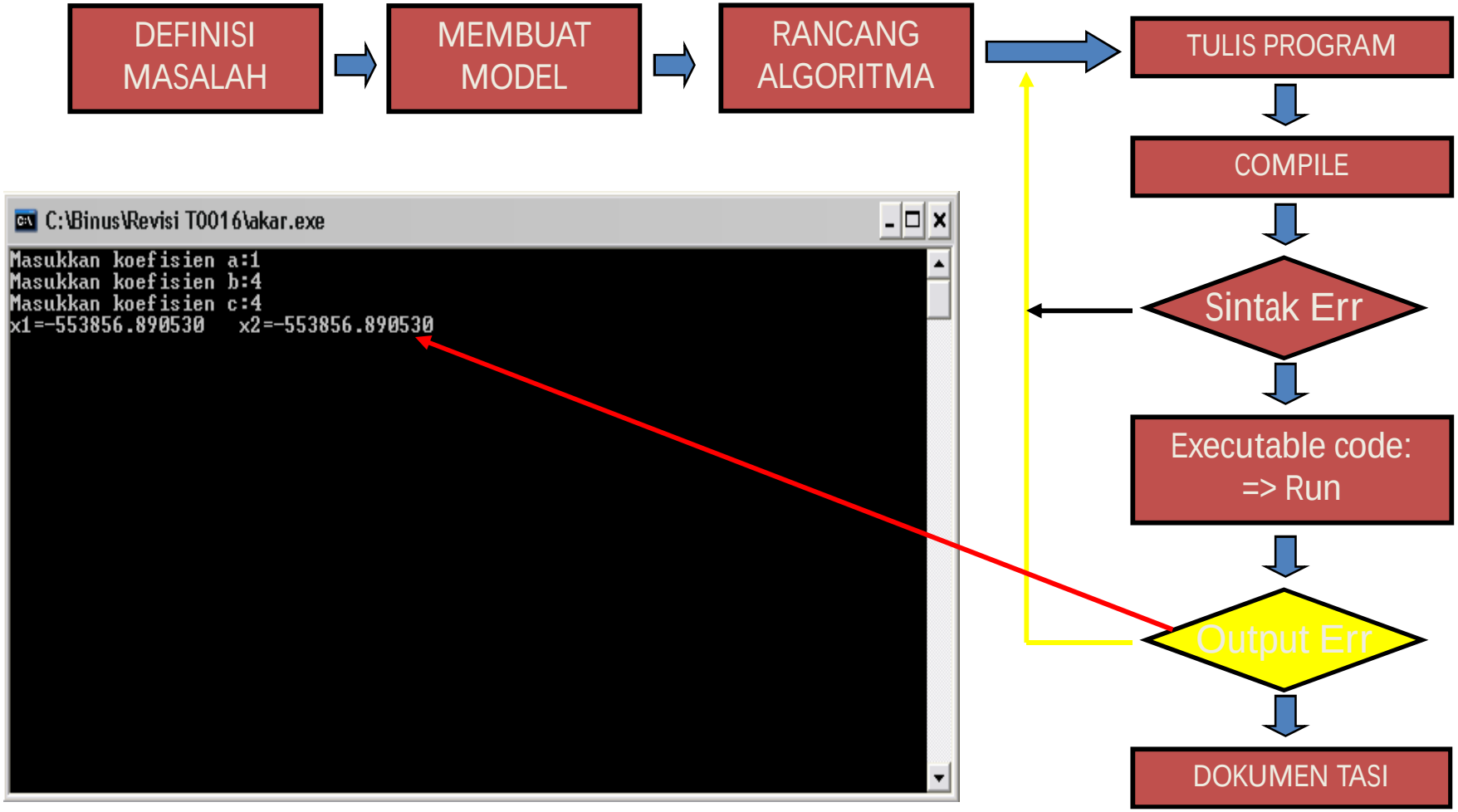
DEFINISI MASALAH → MEMBUAT MODEL → RANCANG ALGORITMA → TULIS PROGRAM → COMPILE → Sintak Err → Executable code: => Run → Output Err → DOKUMEN TASI



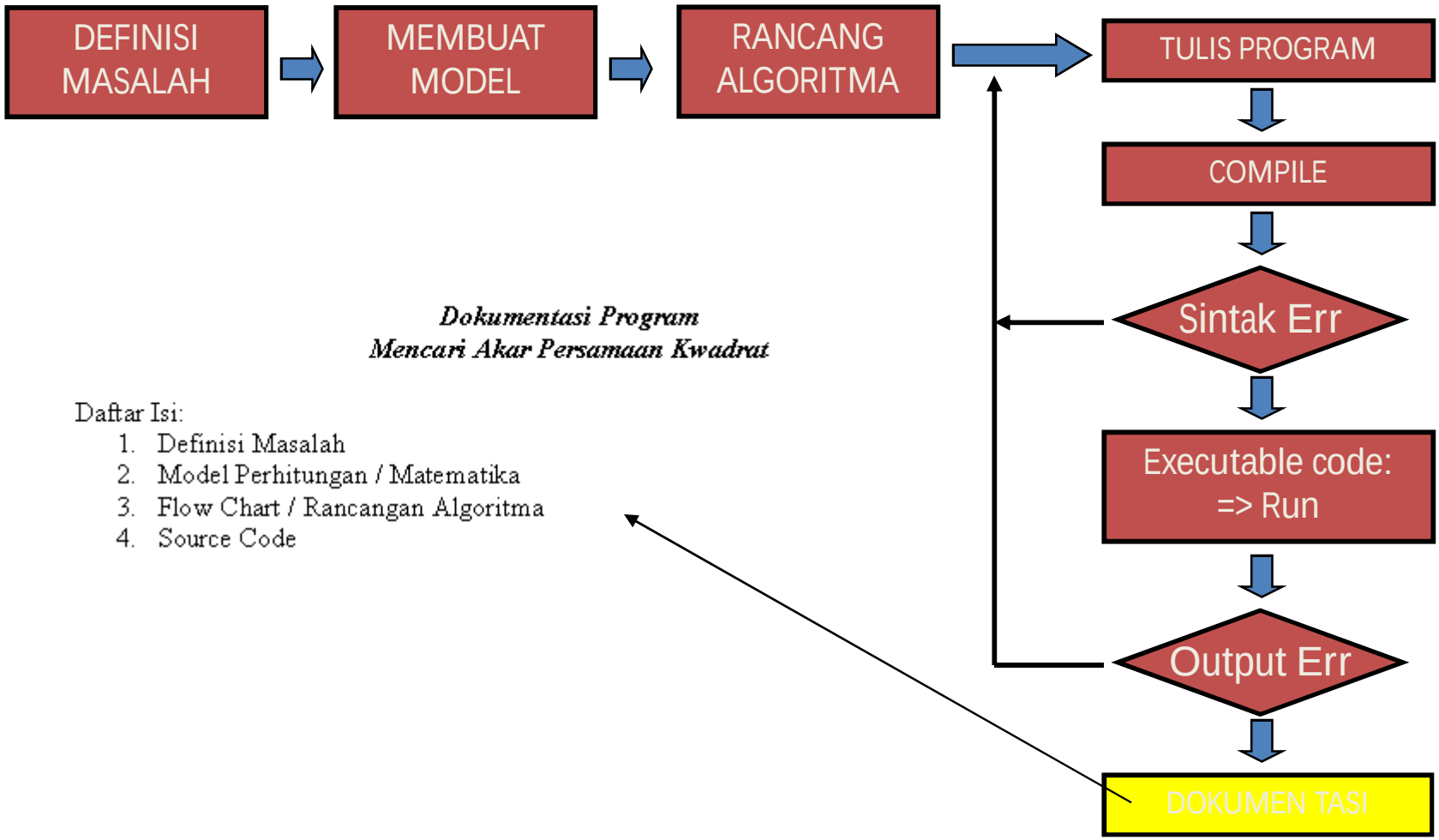
TAHAP PENGEMBANGAN ALGORITMA



TAHAP PENGEMBANGAN ALGORITMA



TAHAP PENGEMBANGAN ALGORITMA



Penyajian Algoritma

- Algoritma bisa dibuat dengan:
  - Teknik tulisan seperti : Structure english dan Pseudocode.
  - Teknik gambar seperti : Flow chart.

Pseudocode

- *Outline* dari sebuah program komputer
- Ditulis dalam bahasa Inggris atau Indonesia sederhana
- Kata kunci (*keyword*) digunakan untuk menjelaskan **struktur kendali** (misalnya: “jika”, “ulangi”, “sampai”,”if”,”repeat”, “until”)

Pseudocode

Enam operasi dasar komputer:

1. Menerima informasi (Input)
2. Menampilkan Informasi (Output)
3. Melakukan perhitungan aritmetika (Compute)
4. Memberikan nilai ke suatu identifier (Store)
5. Membandingkan dan Memilih (Compare)
6. Melakukan pengulangan (Loop)

1. Menerima Informasi

- Sewaktu komputer menerima informasi atau *input*, maka *statement* yang biasa digunakan adalah “Read”, “Get”, “Baca” ,”Input” atau “KeyIn”
- Contoh:

  Read Bilangan

  Get kode_pajak

  Baca nama_mahasiswa

2. Menampilkan Informasi

- Sewaktu komputer menampilkan informasi ataupun *output*, maka *statement* yang biasa digunakan adalah “Print”, “Write”, “Put”, “Output”, “Display” ataupun “Cetak”
- Contoh:

  Print “ITP”

  Cetak “Metode Perancangan Program”

  Output Total

3. Melakukan perhitungan Aritmetika

- Untuk melakukan operasi aritmetika digunakan pseudocode berikut:
  + untuk penjumlahan (add)
  - Untuk pengurangan (subtract)
  * Untuk perkalian (multiply)
  / Untuk pembagian (divide)
  () Untuk kurung
- Statement “Compute”, “Calculate” ataupun “Hitung” juga dapat digunakan.
- Contoh:

  Add number to total
  Total = Total + number

4. Memberikan nilai ke Identifier

- Ada tiga cara untuk memberikan nilai ke dalam variabel :
  - Memberikan nilai awal, menggunakan *statement* “Initialize” atau “Set”
  - Memberikan nilai sebagai hasil dari suatu proses, maka tanda “=“ digunakan
  - Untuk menyimpan suatu nilai maka *statement* “Save” atau “Store” digunakan
- Contoh:

  Set Counter to 0

  Total = Harga * Jumlah

5. Membandingkan dan memilih

- Salah satu operasi terpenting yang dapat dilakukan komputer adalah membandingkan dan memilih salah satu alternatif solusi.
- Keyword yang digunakan : “IF”, “THEN” dan “ELSE”
- Contoh

```
IF Pilih=‘1’ THEN
    Discount = 0.1 * harga
ELSE
    Discount = 0.2 * harga
ENDIF
```

6. Melakukan pengulangan

- Jika ada beberapa perintah yang harus diulang, maka dapat digunakan *keyword* “DOWHILE” dan “ENDDO”.
- Contoh

```
DOWHILE bil < 10
   cetak bil
   bil = bil +1
ENDDO
```

Contoh Algoritma dgn Pseudocode

Contoh : Algoritma Menggunakan Kalkulator

Mulai

Nyalakan kalkulator

Kosongkan Kalkulator

Ulangi

Input harga

Tekan tombol Plus (+)

Sampai semua harga diinput

Tampilkan total harga

Matikan kalkulator

Selesai

Contoh Algoritma dgn Pseudocode

Contoh : Algoritma Berangkat Kuliah

Mulai

Bangun dari tempat tidur

Mandi Pagi

Sarapan Pagi

Pergi Ke Kampus

Cari Ruang Kuliah

Masuk kelas untuk Kuliah

Selesai

Contoh Algoritma dgn Pseudocode

Contoh : Algoritma Sarapan Pagi

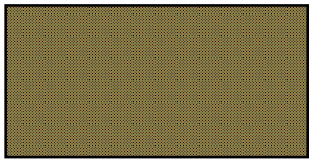
Mulai

Ambil piring
Masukkan nasi dan lauk dalam piring
Ambil sendok dan garpu
Ulangi
    Angkat sendok dan garpu
    Ambil nasi dan lauk
    Suapkan ke dalam mulut
    Taruh sendok dan garpu
    Kunyah
Sampai (nasi dan lauk habis) ATAU kekenyangan
Bereskan piring, sendok dan garpu

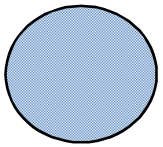
Selesai

FLOW CHART

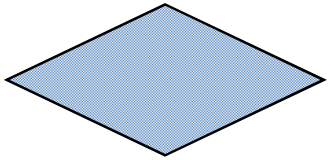
Terminator, MULAI, SELESAI



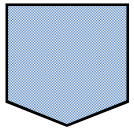
Proses



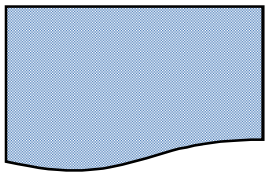
Konektor



Pemilihan



Konektor antar halaman

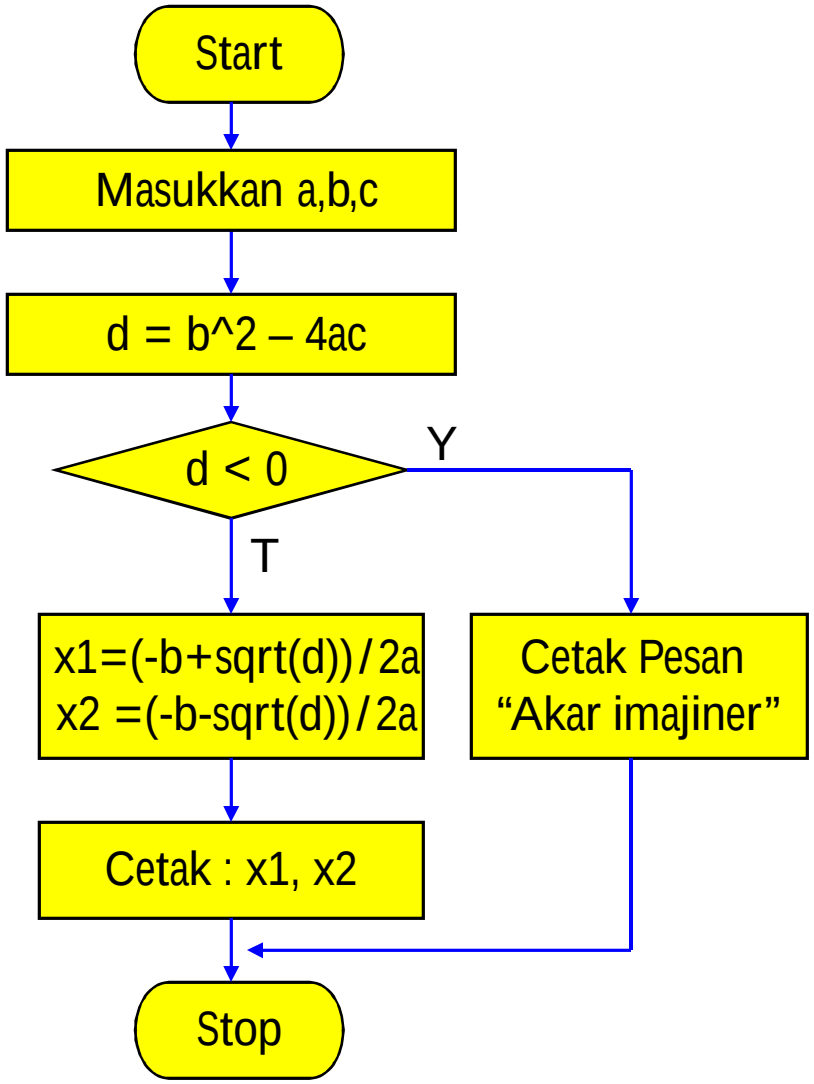


Dokumen

Arah

Contoh Algoritma dgn FLOW CHART

Contoh Flow Chart Untuk Mencari Akar Persamaan Kwadrat



Kriteria Algoritma Yang Baik

- Mempunyai logika yang tepat untuk memecahkan masalah.
- Menghasilkan output yang benar dalam waktu yang singkat.
- Ditulis dengan bahasa baku terstruktur sehingga tidak menimbulkan arti ganda.
- Ditulis dengan format baku sehingga mudah diimplementasikan kedalam bahasa pemrograman.
- Semua operasi didefinisikan dengan jelas dan berakhir sesudah sejumlah langkah.

Teorema Terstruktur

- Teorema terstruktur memungkinkan untuk menulis program komputer hanya dengan menggunakan tiga struktur kontrol yaitu:
  1. Sequence
  2. Selection
  3. Repetition

1. Sequence

- Sequence merupakan urutan pengerjaan dari perintah / *statement* pertama sampai dengan perintah / *statement* terakhir.

- Umumnya bahasa pemrograman mempunyai sequence (urutan pengerjaan dari perintah / *statement*) mulai dari atas ke bawah dan dari kanan ke kiri.

Sequence

- Contoh:

  Cetak “Jumlah Mahasiswa”

  Set Jumlah to 49

  Cetak “Tambahan mahasiswa baru”

  Baca mhs_baru

  Jumlah = Jumlah + mhs_baru

  Cetak “Jumlah Mahasiswa”

  Cetak jumlah

- Penjelasan
  - Urutan pengerjaan adalah mulai dari urutan pertama sampai dengan urutan terakhir, jika mhs_baru diisi dengan 2, maka jumlah yang tercetak adalah 51

2. Selection

- Struktur Kontrol Selection adalah penggambaran sebuah kondisi dan pilihan diantara dua aksi.
- Statement Pertama akan dikerjakan jika kondisi bernilai benar, jika tidak maka akan mengerjakan perintah setelah keyword “else” (jika ada).

Selection

- Contoh :

  IF Hari=1 THEN

  Cetak “Senin”

  ELSE

  Cetak “Bukan hari Senin”

- Penjelasan
  - Tulisan “Senin” akan ditampilkan jika Hari bernilai 1, jika tidak maka tulisan “Bukan hari Senin” yang akan ditampilkan

3. Repetition

- Beberapa *statement* / perintah dapat diulang dengan menggunakan struktur kontrol *repetition*.
- *Statement* / perintah akan tetap diulang selama kondisi perulangan memenuhi (jika menggunakan DOWHILE – ENDDO)

Repetition

- Contoh:

```
Bintang = 0
DOWHILE bintang < 5
    Cetak bintang
    bintang = bintang + 1
ENDDO
```

- Penjelasan:
  - Pertama kali bintang akan diisi dengan 0, setelah itu isi dari bintang akan dicetak sebanyak lima kali, sehingga tampilannya akan sebagai berikut:

    0 1 2 3 4

Latihan

1. Buatlah algoritma menggunakan pseudocode untuk menghitung luas persegi panjang.
2. Buatlah algoritma menggunakan pseudocode untuk mengubah jam dan menit yang diinput ke dalam satuan detik.
3. Buatlah algoritma menggunakan pseudocode untuk menentukan apakah bilangan yang diinput adalah bilangan ganjil atau bilangan genap.

Latihan

4. Buatlah algoritma menggunakan pseudocode untuk menghitung luas lingkaran.
5. Buatlah algoritma menggunakan pseudocode untuk menginput 3 buah bilangan, kemudian tentukan bilangan terbesar, terkecil dan rata-ratanya.

Latihan

- Ulangi latihan no. 1 s/d no. 5 diatas dengan menggunakan Flow Chart.

Pertemuan 3

Pengantar Bahasa C

Learning Outcomes

Pada akhir pertemuan ini, diharapkan mahasiswa akan mampu :

- Menjelaskan elemen dan struktur pemrograman C

Outline Materi

Pengantar Bahasa C

- Sejarah Bahasa C
- Karakter, Identifier, Keyword
- Tipe Data, Konstanta, Variabel
- Struktur Program dan Komentar

Sejarah Bahasa C

- Dirancang oleh Denis M. Ritchie tahun 1972 di Bell Labs.
- Pengembangan dari bahasa BCPL (Martin Richard, 1967) dan bahasa B (Ken Thompson, 1970)
- Dibantu Brian W. Kernighan, Ritchie menulis buku *The C Programming Language* (1978). Dikenal dengan nama K-R C atau C klasik.
- Versi C yang lebih baru Ansi C 1989, iso C 99.

Mengapa C?

- Flexibility : mendekati low level language namun mudah dimengerti.
- Portability : dipakai mulai dari komputer mikro sampai superkomputer
- Bahasa yang banyak digunakan dalam ilmu komputer untuk membuat O/S dan program aplikasi, dll.
- Didukung oleh banyak pustaka (libraries)

Struktur Program

- Bahasa C adalah salah satu bahasa pemrograman yang terstruktur
- Bahasa C terdiri dari fungsi-fungsi
- Tidak ada perbedaan antara prosedur dengan fungsi
- Setiap program C mempunyai satu fungsi dengan nama “main” (program utama).
- Program akan dieksekusi dimulai dari statement pertama pada fungsi “main” tsb.
- Huruf besar dengan huruf kecil diartikan berbeda (case-sensitive).
- Setiap statement diakhiri dengan semi-colon (titik koma (;)).

Struktur Program

- Format penulisan fungsi main

1.
```
main()
{
    statements;

}
```

2.
```
void main()
{
    statements;

}
```

3.
```
main()
{
    statements;
    return (0);
}
```

4.
```
int main()
{
    statements;
    return (0);
}
```

Struktur Program

- Sering dijumpai beberapa format penulisan fungsi main seperti contoh diatas, tetapi tidak semua compiler mengenalnya.
- Penulisan fungsi main yang standard seperti contoh No. 3 atau 4 diatas.
- return (0), menyatakan program exit secara normal.
- Fungsi main dan juga fungsi yg lainnya jika tidak diberikan tipe maka defaultnya integer (int). Pada contoh diatas No. 3 dan 4 artinya sama.
- Contoh:
  - dengan menggunakan compiler Turbo C 2.0 (DOS) dan Microsoft Visual C++ (windows), (2), (3) dan (4) => Success, tetapi (1) warning
  - dengan menggunakan compiler Dev-C (windows), dan gcc (linux) (1), (3) dan (4) => Success, tetapi (2) warning

Struktur Program

- Contoh:

```
int main()
{
    printf("Selamat datang di ITP\n");
    return 0;
}
```

Jika di kompilasi dengan Turbo C 2.0 program ini akan error, dgn Error Message: Function printf should have a function prototype.

```
#include <stdio.h>
int main()
{
    printf("Selamat datang di ITP\n");
    return (0);
}
```

#include adalah sebuah directive/arahan untuk memberitahu compiler bahwa function prototype untuk fungsi printf ada pada header file stdio.h

Struktur Program

- Penulisan **return 0;** sama artinya dengan **return (0);**
- **#include <stdio.h>**
  - **File stdio.h akan dicari mulai dari directory tempat header file tersebut di-install. Jika tdk ada akan dilanjutkan mencarinya ke current / working directory.**
  - **Contoh pd Turbo C 2.0, header file di install pada directory ... \ INCLUDE**

Struktur Program

- **#include ”stdio.h”**
  - **File stdio.h akan dicari mulai dari current / working directory, dan jika tdk ada akan dilanjutkan mencarinya ke directory tempat header file tersebut disimpan pada saat menginstall compiler-nya.**
- **Directive #include umumnya ditulis di awal program**
- Style penulisan (tergantung kesenangan programmer) :

```
#include <stdio.h>
int main()
{
    printf(“”Selamat datang\n”);
    return (0);
}
```

Sama

```
#include <stdio.h>
int main() {
    printf(“”Selamat datang\n”);
    return (0);
}
```

Komentar

- Menggunakan pasangan /* dan */
- Digunakan agar program lebih mudah dibaca dan dimengerti
- Diabaikan oleh *compiler*
- Untuk komentar 1 baris cukup menggunakan tanda // diawal baris
- Contoh program C sederhana :

```
/*-------------------------------
  Program Pertama
  -------------------------------*/
#include<stdio.h>
void main() {
            printf("Halo, Tuan\n");
} //Program mencetak tulisan Halo, Tuan
```

Escape Sequences

- \a bell, alert, system beep
- \b back space
- \t horizontal tab
- \n new line, line feed
- \v vertical tab
- \r carriage return
- \’ single quote
- \” double quote
- \\ backslash
- \xdd notasi hexadecimal
- \ddd notasi octal

Karakter

- Program C ditulis menggunakan *subset* karakter ASCII yaitu:
    - Huruf besar A .. Z
    - Huruf kecil a .. z
    - Digit 0 .. 9
    - Karakter khusus seperti ‘!’ , ‘&’, ‘+’, ‘\’, ‘_’ dan sebagainya.
- ASCII = American Standards Committee for Information Interchange

ASCII Character Codes Chart 1

Ctrl	Dec	Hex	Char	Code
^@	0	00		NUL
^A	1	01	☺	SOH
^B	2	02	☻	SIX
^C	3	03	♥	EIX
^D	4	04	♦	EOI
^E	5	05	♣	ENQ
^F	6	06	♠	ACK
^G	7	07	•	BEL
^H	8	08	◘	BS
^I	9	09	○	HI
^J	10	0A	◙	LF
^K	11	0B	♂	VI
^L	12	0C	♀	FF
^M	13	0D	♪	CR
^N	14	0E	♫	SO
^O	15	0F	☼	SI
^P	16	10	►	SLE
^Q	17	11	◄	CS1
^R	18	12	↕	DC2
^S	19	13	‼	DC3
^I	20	14	¶	DC4
^U	21	15	§	NAK
^V	22	16	▬	SYN
^W	23	17	↨	EIB
^X	24	18	↑	CAN
^Y	25	19	↓	EM
^Z	26	1A	→	SIB
^[	27	1B	←	ESC
^\	28	1C	∟	FS
^]	29	1D	↔	GS
^^	30	1E	▲	RS
^_	31	1F	▼	US

Dec	Hex	Char
32	20	sp
33	21	!
34	22	"
35	23	#
36	24	$
37	25	%
38	26	&
39	27	'
40	28	(
41	29	)
42	2A	*
43	2B	+
44	2C	,
45	2D	-
46	2E	.
47	2F	/
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	:
59	3B	;
60	3C	<
61	3D	=
62	3E	>
63	3F	?

Dec	Hex	Char
64	40	@
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z
91	5B	[
92	5C	\
93	5D	]
94	5E	^
95	5F	_

Dec	Hex	Char
96	60	`
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z
123	7B	{
124	7C	|
125	7D	}
126	7E	~
127	7F	⌂

Identifier

- Nama berbagai elemen program seperti nama variabel, fungsi, konstanta, dsb
- Diawali dengan huruf atau garis bawah(*underscore*) _ dan diikuti dengan huruf, digit atau _
- Huruf besar dianggap berbeda dengan huruf kecil (*case sensitive*)
- Panjang maksimum Identifier tergantung dari compiler sebagai contoh Turbo C 2.0 (DOS), max 32 karakter
- Tidak boleh menggunakan ***keyword*** (misal *for, while, dll.*)
- Contoh identifier : name, x1, _total, cubic()
- Contoh identifier yang salah: 1kali, int

Keyword

- **Keywords** are words that have special meaning to the C compiler.
- Contoh Keyword

Keywords			
auto	**double**	**int**	**struct**
break	**else**	**long**	**switch**
case	**enum**	**register**	**typedef**
char	**extern**	**return**	**union**
const	**float**	**short**	**unsigned**
continue	**for**	**signed**	**void**
default	**goto**	**sizeof**	**volatile**
do	**if**	**static**	**while**

Keyword

- Beberapa compiler akan memberikan warna yang berbeda untuk keyword, sperti pada dev-C atau Visual C++, dibawah ini.

```c
#include <stdio.h>
int main()
{
   int x;
   for(x=0; x<5; x++) printf("Hello\n");
   return(0);
}
```

Pada Dev-C keyword dicetak BOLD

Pada Visual C++ keyword dicetak dengan warna biru

```c
#include "stdafx.h"

int main(int argc, char* argv[])
{
    printf("Hello World!\n");
    return 0;
}
```

Variabel

- Variabel : Identifier yang digunakan untuk menampung data/informasi
- Setiap variabel memiliki **Nama**, **alamat** (L-Value), tipe, **size**(rentang nilai) dan **data** (R-Value).
- Data atau isi variabel bisa dirubah-ubah pada saat Run time
- Format deklarasi variabel:

  <tipe> <nama_variabel>;

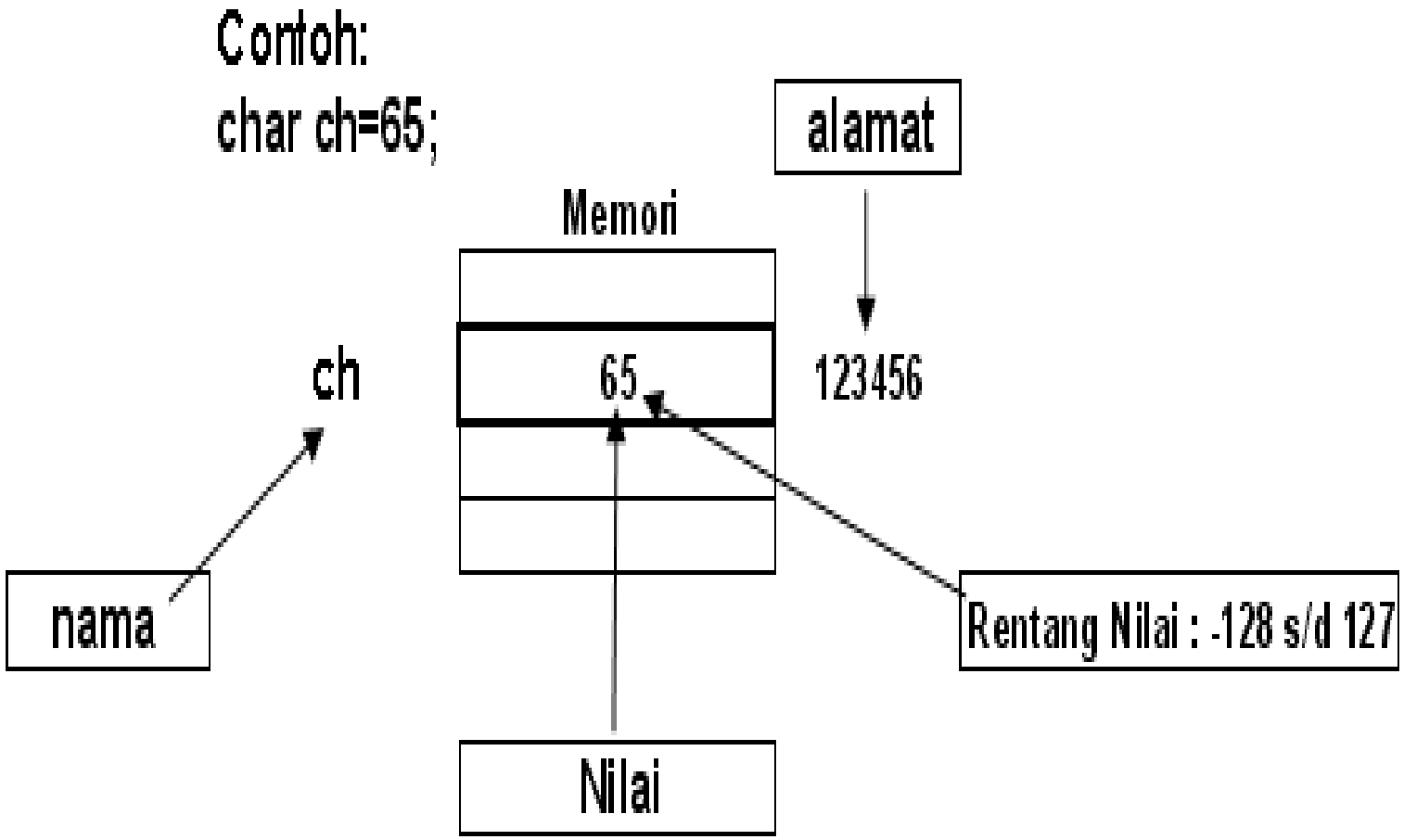
  <tipe> <nama_variabel> = <initial_value>;
- Contoh:

  int a, b, c, jumlah;

  float gaji, bonus;

  int jml_mhs = 20;

Variabel


Contoh:
char ch=65;
alamat
Memori
ch
65
123456
nama
Rentang Nilai : -128 s/d 127
Nilai

Variabel

- **Deklarasi Variabel:**
  - Variabel dpt dideklarasikan disetiap awal blok statement.
  - Blok statement disebut juga “compound statement” adalah statement-statement yang berada diantara { dan }
  - Contoh deklarasi variabel:

```
int  x;
int  y;
int  z;

atau bisa ditulis :

int  x, y, z;

atau bisa juga ditulis :

int x; int y; int z;
```

Tipe Data

- Pada dasarnya tipe data pada bahasa C ada 5, dan ditambah 4 tipe Modifier yaitu :

Basic Data Types	**Keyword**
1. Character 2. Integer 3. Floating point 4. Double floating point 5. Void	1. char 2. int 3. float 4. double 5. void

Tipe Data

- Empat tipe Modifier adalah sbb :
  - signed
  - unsigned
  - long
  - short
- Tipe data dalam bahasa C merupakan kombinasi antara Basic Data Types dengan Modifier.
- Contoh : signed char, unsigned int, long int, dll.

Tipe Data dalam Bahasa C

CONTOH : TIPE DATA DAN RENTANG NILAI PADA TURBO C 2.0 (DOS)

Tipe Data	Penulisan	Memori	Rentang Nilai
character	unsigned char char	1 Byte 1 Byte	0 s/d 255 -128 s/d 127
integer	unsigned int int short int unsigned long long	2 Byte 2 Byte 1 Byte 4 Byte 4 Byte	0 s/d 65535 -32768 s/d 32767 -128 s/d 127 0 s/d 4294967295 -2147483648 s/d 2147483647
float	float double long double	4 Byte 8 Byte 16 Byte	3.4E-38 s/d 3.4E+38 1.7E-308 s/d 1.7E+308 3.4E-4932 s/d 1.1E+4932

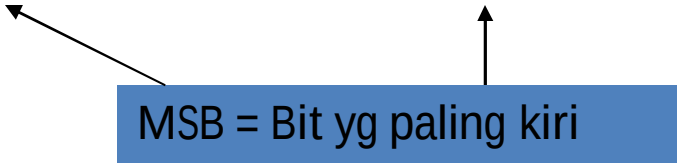
Tipe Data dalam Bahasa C

- Defaultnya signed (bilangan bertanda), sehingga penulisan int sama artinya dgn signed int
- Contoh : int x; sama artinya dgn signed int x;
- short int x; sama artinya dgn signed short int x;
- Rentang Nilai (range) dari tipe data pada bahasa C tergantung dari compiler dan Sistem Operasi.
- Contoh :
  - Tipe integer pada Turbo C 2.0 (DOS), rentang nilainya 2 byte (-32768 s/d 32767)
  - Tipe integer pada Dev-C (Windows), rentang nilainya 4 byte (-2147483648 s/d 2147483647)

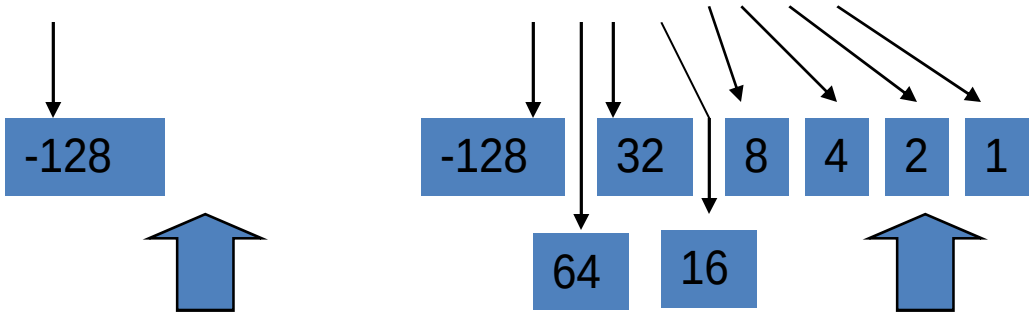
Tipe Data dalam Bahasa C

- Mengapa tipe char rentang nilainya dari -128 s/d 127 ?
- 1 Byte = 8-bit

  00000000 s/d 01111111 (MSB=>0 = Bil Positip)



10000000 s/d 11111111 (MSB=>1 = Bil Negatip)



Jika dijumlahkan hasilnya = -128

Jika dijumlahkan hasilnya = -1

Tipe Data dalam Bahasa C

Rentang Nilai Tipe data signed char

BINER 8-bit	DESIMAL
00000000	0
00000001	1
00000010	2
00000011	3
00000100	4
00000101	5
00000110	6
00000111	7
----------	----
01111111	127
10000000	-128
10000001	-127
10000010	-126
----------	----
11111110	-2
11111111	-1

Tipe Data dalam Bahasa C

Rentang Nilai Tipe data unsigned char

BINER 8-bit	Desimal
00000000	0
00000001	1
00000010	2
00000011	3
00000100	4
00000101	5
00000110	6
00000111	7

01111111	127
10000000	128

11111110	254
11111111	255

Tipe Data dalam Bahasa C

- Contoh

  char ch=127;
  ch=ch+1;
  Berapa nilai ch ?
  Jawaban : -128 bukan 128

- Contoh

  int ch=127;
  ch=ch+1;
  Berapa nilai ch ?
  Jawaban : 128

Tipe Data dalam Bahasa C

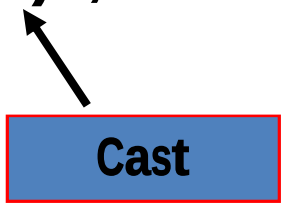
- Keyword void disamping digunakan dalam fungsi untuk tidak mengembalikan nilai, juga digunakan sebagai tipe data.

- Tipe data void : adalah tipe data yang bisa dirubah menjadi tipe apa saja (*akan dibahas lebih lanjut pada saat membahas pointer*)

Cast

- Cast : untuk mengkonversi tipe data pada bahasa C
- Sintak : (type)
- Contoh :

```
int x;
float f = 3.134;
x = (int) f;
```



Symbolic Constant

- Symbolic Constant adalah Identifier yang hanya memiliki nilai (R-Value), dan nilai-nya tidak bisa dirubah-rubah pada saat run time.
- Symbolic Constant tidak memiliki alamat (L-Value)
- Pada bahasa C deklarasi symbolic constant tidak membutuhkan alokasi memori
- Untuk mendeklarasikan symbolic constant pada bahasa C bisa dilakukan dengan pre-processor directive ***#define*** atau dengan keyword ***const.***
- Contoh:

```
const float Pi=3.14;
#define  Pi  3.14
```

Symbolic Constant

```
#define  Pi  3.14
int main()
{
      Pi=3.1475;  //Error
      return 0;
}
```

```
int main()
{
      const float Pi=3.14;
       Pi=3.1475;  //Error
       return 0;
}
```

```
#define  Pi  3.14
int main()
{
      float PHI=3.14;
      PHI = 3.1475;  //OK (variable)
      Pi=3.1475;  //Error
      return 0;
}
```

Konstanta

Konstanta / symbolic constant tidak memiliki alamat (hanya nilai) dan nilainya tdk bisa dirubah saat run time.

Jenis-jenis konstanta:

① *Integer constant* ⇨ -5
② *Floating-point constant* ⇨ 3.14
③ *Character constant* ⇨ 'C' '1' '$'
④ *Escape sequence* ⇨ \n \t \''
⑤ *String constant* ⇨ ''TuAn''
⑥Symbolic *constant* ⇨ *#define PHI 3.14*
⇨ const float PHI=3.14;

- 'H' adalah sebuah *character constant*
- ''H'' adalah sebuah *string constant*
- 1 adalah sebuah *integer constant*
- '1' adalah sebuah *character constant*
- const float Pi= 3.1415926; Pi adalah sebuah symbolic constant

Contoh Program

- Penambahan dua buah bilangan

  Data telah berada di memori (variabel), hasil penjumlahan disimpan di memori (variabel)

```
/* Program Tambah */          /*komentar */
int x,y,z;                    /*Deklarasi Variabel global*/
int main()
{                             /*Program utama mulai*/
      x = 20;                 /*Statement 1*/
      y = 30;                 /*Statement 2*/
      z = x + y;              /*Statement 3*/
      return 0;               /*Statmenet 4*/
}                             /*Program utama selesai*/
```

Contoh Program

- Program menghitung luas lingkaran

  Data jejari dibaca dari keyboard, kemudian hasil perhitungan ditayangkan di layar monitor.

```
/*--------------------------------
  Program Luas_Lingkaran
  --------------------------------*/
#include <stdio.h>
const float Pi = 3.14;                          /*Deklarasi Konstanta*/
int main()                                      /*Program utama mulai*/
{
        float jejari;                           /*Deklarasi local Variabel*/
        float luas;
        scanf("%f",&jejari);                    /*Masukkan dari keyboard*/
        luas = Pi * jejari * jejari;
        printf("Luas = %5.2f", luas);    /*Mencetak ke layar*/
      return (0);
}                                               /*Program utama selesai*/
```

Sizeof

- **sizeof adalah sebuah operator untuk mengetahui jumlah memori (byte) yang diperlukan oleh suatu tipe data pada bahasa C**
- **Sintaknya : sizeof** *expression*
- Contoh :

  sizeof(int) = 4 => pada Dev-V (Windows)

  sizeof(int) = 2 => pada Turbo C versi 2.0 (DOS)

Suffix

- Bahasa C menyediakan Suffix (akhiran) untuk bilangan floating point konstan sbb:
  - F atau f untuk tipe float
  - L atau l untuk tipe long double
  - Default tipenya double

- Contoh:
  - 3.14 => (double)
  - 3.14f => (float)
  - 3.14L => (long double)

Suffix

- Bahasa C menyediakan Suffix (akhiran) untuk bilangan integer konstan sbb:
  - U atau u untuk tipe unsigned integer
  - L atau l untuk tipe long integer
  - UL atau ul atau LU atau lu untuk tipe bilangan unsigned long integer
  - Default tipenya integer

- Contoh:
  - 174 => (integer)
  - 174u => (unsigned integer)
  - 174L => (long integer)
  - 174ul => (unsigned long integer)

Suffix

- Beberapa compiler akan memberikan warning karena perbedaan tipe data seperti compiler pada Visual C++ sbb:
- Contoh:

```
float  x;
x = 3.14;
```

warning : truncation from 'const double' to 'float'

- Cara mengatasi sbb:

```
float x;
x = (float)3.14;  //menggunakan cast atau
x = 3.14f;        //menggunakan suffix
```

Suffix

```
#include <stdio.h>

int main()
{
    printf("Sizeof Floating Point Constan :\n");
    printf(" - dgn suffix f  = %d\n",sizeof(3.14f));
    printf(" - tanpa suffix  = %d\n",sizeof(3.14));
    printf(" - dgn suffix L  = %d\n",sizeof(3.14L));

    getch();
    return 0;
}
```

Output :

```
Sizeof Floating Point Constan :
 - dgn suffix f  = 4
 - tanpa suffix  = 8
 - dgn suffix  L = 12
```

Latihan

1. Apakah yang dimaksud dengan fungsi library dalam bahasa C ?
2. Jelaskan apa yang dimaksud dengan identifier pada bahasa C
3. Berapakah data terbesar yang bisa ditampung oleh variabel x yang bertipe integer dengan ukuran 20-bit ?
4. Sebutkan tipe data (boleh lebih dari satu) yang tepat untuk mengolah data umur !
5. Sebutkan tipe data yang tepat untuk menampung data Nim Mahasiswa !
6. Sebutkan tipe data yang tepat untuk menampung data saldo rekening !

LATIHAN

7. Kapan sebaiknya menggunakan directive:
   - #include <header.h> atau
   - #include "header.h"

8. Jika tipe integer ukurannya hanya 3 byte, berapakah rentang nilainya (range) ?

9. Jika tipe unsigned integer ukurannya hanya 3 byte, berapakah rentang nilainya (range) ?

10. Apakah beda Variabel dengan Konstanta ?

11. Jika x variabel bertipe integer dan ukurannya 2 byte, jika x=32767 maka berapakah nilai x jika nilai x ditambah 1 ?

12. Diketahui bilangan biner 10-bit sbb:
    1010110011
    a. Jika bilangan biner tersebut adalah bilangan signed berapa nilai desimalnya ?
    b. Jika bilangan biner tersebut adalah bilangan unsigned berapa nilai desimalnya ?

LATIHAN

13. Jelaskan tentang sistem bilangan
    - Desimal
    - Biner
    - Oktal
    - Heksadesimal

14. Jelaskan bagaimana cara mengkonversi sistem bilangan desimal ke biner, oktal dan Heksadesimal, dan sebaliknya

15. Jelaskan tentang sistem bilangan two's complement, one's complement.

16. #define PHI 3.142857

    Apa keuntungan kita menggunakan konstanta PHI spt diatas? Mengapa tdk langsung saja menggunakan angka 3.142857

Pertemuan 4

Operasi Input Output

Learning Outcomes

Pada akhir pertemuan ini, diharapkan mahasiswa akan mampu :

- Menggunakan standard library function yang berhubungan dengan operasi input dan output

Outline Materi

Operasi Input-Output

- Standard Library Function yang berhubungan dengan operasi Output spt: printf, putchar, putch, puts, dll.
- Format Output
- Standard Library Function yang berhubungan dengan operasi input spt: scanf, getchar, getch, getche, gets, dll
- Format Input
- Format Output

KATEGORI FUNCTION

- Standard library function, fungsi-fungsi yang telah disediakan oleh *compiler* C, tinggal digunakan, dengan mencatumkan header file tempat function tersebut didefinisikan (di-include)

- Programmer-defined function, fungsi-fungsi yang dibuat pemrogram untuk digunakan dalam program ***(akan dibahas pada pertemuan berikutnya).***

Operasi Input

- Standard library function yang berhubungan dengan operasi input antara lain:
  - scanf();
  - getchar();
  - getch();
  - getche();
  - gets();
  - dll.
- Operasi input : operasi untuk mengambil data/informasi dari I/O spt, keyboard, disk, dll.

Fungsi scanf

- Header file ada di ***stdio.h***
- *Format:*

  *int scanf(const char *format [,argument]...);*
- *Semua tipe argument pada scanf adalah pointer (alamat dari variabel yang akan diisi data).*
- Untuk mendapatkan alamat dari variabel digunakan tanda *&*
- Fungsi scanf didefinisikan di header file stdio.h
- *Contoh :*

  int NilaiTM;

  scanf*(”%d”,&NilaiTM);*
- Spesifikai *format* adalah : ***”% type”***

  dimana ***type*** bisa diganti dengan salah satu dari sbb:

Fungsi scanf

type	*Digunakan untuk membaca*
d **u** **x** **e,f,g** **c**	- Data integer - Data unsigned integer - Data heksadesimal - Data floating point - Single character
s **O** **[...]** **[^..]**	- Karakter string yg diakhiri dengan whitespace - Data unsigned octal - Karakter string yg diakhiri dengan karakter yg tidak ada didalam [...] - Karakter string yg diakhiri dengan karakter yg ada didalam [..]

Fungsi scanf

- Jika x adalah variabel bertipe integer, Apa beda x dgn &x ?

  Jawab:

y = x; maka y akan berisi data 234

y = &x; maka y akan berisi alamat dari x yaitu 45678

Fungsi scanf

- Fungsi scanf mengembalikan tipe integer, dimana nilai nya menyatakan jumlah field yang sukses di assigned
- Contoh:

```
int x,y,z,w;
x=scanf("%d %d %d",&y,&z,&w);
```

  - Jika di input dari keyboard 3 buah nilai interger 6 7 8, maka nilai x = 3;
  - Jika di input dari keyboard 4 buah nilai interger 6 7 8 9 maka nilai x = 3 (karena 3 nilai yg sukses di- assigned masing-masing ke variabel y, z dan w)

Fungsi scanf

- Program Luas Segi Empat

```
/* Program Luas_Segi_Empat v1*/

#include <stdio.h>
int main()
{
      int panjang, lebar, luas;
      scanf("%d",&panjang);
      scanf("%d",&lebar);
      luas = panjang * lebar;
      return(0);
}
```

Fungsi scanf

- Fungsi scanf dapat memakai lebih dari satu argument

```
/* Program Luas_Segi_Empat v2*/

#include <stdio.h>
int main()
{
      int panjang, lebar, luas;
      scanf("%d %d",&panjang, &lebar);
      luas = panjang * lebar;
      return(0);
}
```

Fungsi scanf

- Tipe data untuk setiap variabel dalam argumen boleh berbeda

```
/** Program Argumen Tipe Beda**/

#include <stdio.h>
int main()
{
      int nomor; char inisial; float saldo;
      scanf("%d %c %f" ,&nomor ,&inisial, &saldo);

      <statement selanjutnya>

      return(0);
}
```

Fungsi scanf

- Mengambil data string dari keyboard dengan fungsi scanf menggunakan format: %s.
- Contoh:

```
char ss[40];
scanf("%s",ss);
```

- Perhatikan contoh diatas, karena variabel ss tipenya sudah pointer *(Topik tentang pointer akan dibahas tersendiri),* maka tidak perlu ditambah tanda & lagi (&ss).
- String yang diambil hanya sampai ketemu karakter whitespace.

Fungsi scanf

- Karakter Space, tab, linefeed, carriage-return, formfeed, vertical-tab, dan newline disebut ***"white-space characters"***
- Contoh :
  - Pada potongan program diatas, jika dimasukkan string "Selamat Pagi Pak" dari keyboard maka yg dimasukkan ke variabel ss hanya "Selamat" saja.
- Untuk mengambil string yang diakhiri karakter tertentu (misalnya ENTER), dengan scanf, menggunakan format [^\n]

Fungsi scanf

- Contoh:

```
char ss[40];
scanf("%[^\n]",ss);
```

  – Pada potongan program diatas, jika dimasukkan string "Selamat Pagi Pak" kemudian tekan ENTER dari keyboard maka variabel ss berisi string "Selamat Pagi Pak"

Fungsi scanf

- Contoh:

```
char ss[40];
scanf("%[a-z]",ss);
```

  – Pada potongan program diatas, jika dimasukkan string: http://palanta.itp.ac.id kemudian tekan ENTER dari keyboard maka variabel ss hanya berisi string: http karena titik dua (:) tidak ada di antara a s/d z, dan titik dua dianggap sebagai akhir dari string.

Fungsi scanf

- Contoh:

```
int x;
scanf("%o", &x);
```

  – Pada potongan program diatas, jika dimasukkan bilangan : 44 kemudian tekan ENTER dari keyboard maka variabel x akan berisi nilai : 36 desimal, karena 44 dianggap bilangan berbasis oktal.

Fungsi scanf

- Contoh:

```
int x;
scanf("%x", &x);
```

  – Pada potongan program diatas, jika dimasukkan bilangan : 44 kemudian tekan ENTER dari keyboard maka variabel x akan berisi nilai : 68 desimal, karena 44 dianggap bilangan berbasis heksadesimal.

Fungsi getchar()

- Sintak:

  int **getchar**(void):

- Fungsi:
  - mengembalikan sebuah karakter (nilai ASCII) berikutnya dari buffer keyboard.
  - Karakter ditampilkan di layar monitor
  - Menunggu sampai ada ENTER
  - Header file ada di **stdio.h**
- Contoh:

```
char ch;
ch = getchar();
```

Fungsi getch()

- Sintak:

  int **getch**(void):

- Fungsi:
  - mengembalikan satu karakter dari buffer keyboard
  - karakter tidak ditampilkan di layar monitor (no echo)
  - Tidak menunggu sampai ada ENTER
  - Cocok untuk membuat password
  - Header file ada di **conio.h**
- Contoh:

```
char ch;
ch = getch();
```

Fungsi getche()

- Sintak:

  int getche(void)

- Fungsi :
  - mengembalikan satu karakter dari keyboard
  - Karakter ditampilkan di layar (echo)
  - Tidak menunggu sampai ada ENTER
  - Header file ada di **conio.h**
- Contoh:

```
char ch;
ch = getche();
```

Fungsi gets()

- Sintak:

  char *gets(char *buffer)
- Fungsi:
  - membaca string dari keyboard sampai ketemu new-line dan disimpan pada buffer.
  - Kemudian new-line di replace dengan null character
  - Mengembalikan nilai NULL jika ada error dan mengembalikan argument-nya (buffer) jika sukses.
- Contoh:

```
char buffer[40];
char *ptr;
ptr = gets(buffer);
```

Operasi Output

- Operasi ini digunakan untuk menampilkan data ke layar monitor. Beberapa fungsi (stndard library function) yang yang ada pada bahasa C antara lain:

  printf();

  putchar();

  putch();

  puts();

  dll.

Fungsi printf

- Menampilkan sejumlah data ke standard output, dengan format tertentu.
- Standard output adalah Layar Monitor, sedangkan Standard Input adalah Keyboard.
- Sintak :

  printf(const char *format[,argument, ...]);

- Header file untuk printf : **stdio.h**
- Contoh :

Fungsi printf

```
/** Program Luas_Segi_Empat **/
void main()
{
   int panjang, lebar, luas;          /* local variable */
   printf(”Panjang = ”); scanf(“%d”,&panjang);
   printf(”Lebar = ‘); scanf(“%d”,&lebar);
   luas = panjang * lebar;
   printf(”Luas = %d\n”, luas);
}
```

Fungsi printf

- Spesifikai format sbb:

%[flags][width][.precision] type

type dapat diganti dengan :

d atau i	:	signed decimal
o	:	unsigned octal
u	:	unsigned decimal
x	:	unsigned hexadecimal
f	:	floating point
e	:	floating point (exponent)
c	:	single character
s	:	string
%	:	% character
p	:	pointer

width : menentukan jumlah kolom yang disediakan

precision : menentukan jumlah angka dibelakang koma (untuk bilangan pecahan)

flags dapat diganti sbb:

none	: right justify (rata kanan)
-	: left justify (rata kiri)
+	: untuk bilangan dimulai dgn tanda – jika negatip atau + jika positip

Fungsi printf

- CONTOH 1

  printf(“%6d”, 34); ….34

  printf(”%-6d”, 34); 34….

- CONTOH 2

  printf(“%10s”, “ITP”); …..ITP

  printf(“%-10s”, “ITP”); ITP…..

  printf(“%8.2f”, 3.14159); ….3.14

  printf(“%-8.3f”, 3.14159); 3.141…

Fungsi printf

```
printf("%c\n",65);    //akan ditampilkan A
printf("%x\n",'A');    // akan ditampilkan 41
printf("%o\n",65);   // akan ditampilkan 101
printf("%+d\n",34);  // akan ditampilkan +34
printf("%+d\n",-45); // akan ditampilkan -45
printf("%e\n",3.14); // akan ditampilkan
                              3.140000e+000
```

Fungsi printf

```
#include <stdio.h>
int main(){
    char ss[]="Selamat Datang";
    printf("12345678901234567890123456789\n");
    printf("%.10s di ITP\n",ss);
    printf("%10s di ITP\n",ss);
    printf("%-10s di ITP\n",ss);
    printf("%.20s di ITP\n",ss);
    printf("%20s di ITP\n",ss);
    printf("%-20s di ITP\n",ss);
    printf("%20.10s di ITP\n",ss);
    printf("%-20.10s di ITP\n",ss);
    return 0;
}
```

Fungsi printf

Output Program diatas sbb:

```
12345678901234567890123456789O
Selamat Da di ITP
Selamat Datang di ITP
Selamat Datang di ITP
Selamat Datang di ITP
      Selamat Datang di ITP
Selamat Datang        di ITP
          Selamat Da di ITP
Selamat Da            di ITP
```

Fungsi printf

- Untuk data yang tipe nya long maka ditambahkan l sebelum tipe datanya seperti :
  - long double ➔ (“ %lf “)
  - unsigned long int ➔ (“ %lu ”)
  - long int ➔ (“ %ld “)

Fungsi putchar()

- **Sintak:**

  int putchar(int c)
- Fungsi:
  - Menampilkan karakter ke layar monitor pada cursor, kemudian setelah ditampilkan cursor bergerak ke posisi berikutnya.
  - Mengembalikan EOF jika error, dan mengembalikan karakter yang ditampilkan jika sukses
  - Putchar adalah macro yang sama artinya dengan: **putc(***c***, stdout)**
  - Header File : stdio.h
- Contoh:

```
char ch='A';
putchar(ch);
```

Fungsi putch()

- **Sintak:**
  - **int putch**(int ch)
- Fungsi :
  - menampilkan karakter ascii di ch di monitor tanpa memindahkan kursor ke posisi berikutnya
  - Header file : conio.h
  - Mengembalikan EOF jika error, dan mengembalikan karakter yang di tampilkan jika sukses.
- Contoh:

```
char ch='b';
putch(ch);
```

Fungsi puts()

- **Sintak:**

  int **puts**(const char *str);
- Fungsi:
  - Menampilkan string str ke layar monitor dan memindahkan kursor ke baris baru.
  - Header file: stdio.h
  - Mengembalikan nilai non-negative jika sukses dan EOF jika ada error.
- Contoh:

  puts(”Selamat Datang”);

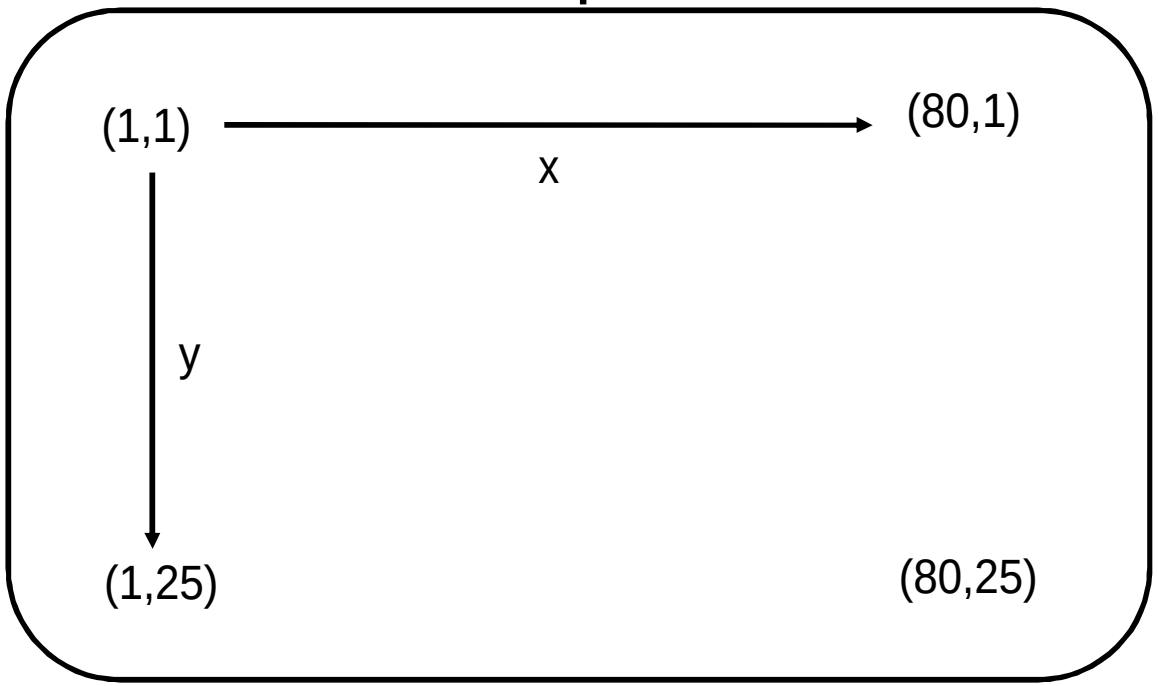
  puts(”Di ITP”);

  Tampilan di layar monitor:

  Selamat Datang

  Di ITP

Penempatan Kursor



- Layar dibagi dalam kolom dan baris, umumnya digunakan ukuran kolom = 80 dan baris = 25.

Penempatan Kursor

- Layar dapat dihapus dengan menggunakan fungsi: ***clrscr();***
- Kursor dapat dipindahkan ke posisi manapun di dalam layar monitor dengan menggunakan fungsi : ***gotoxy(col,row);*** dimana col = kolom dan row = baris
- Sebagian dari baris, mulai posisi kursor hingga akhir baris (end of line), dapat dihapus dengan fungsi: ***clreol();***
- Function prototype untuk fungsi gotoxy(), clrscr(), clreol() pada bahasa C terdapat pada header file : <**conio.h**>

Penempatan Kursor

- Contoh :

```
#include <stdio.h>
#include <conio.h>

int main()
{
        float panjang, lebar, luas, keliling;        /* local variable */
        clrscr();                                    /*membersihkan layar*/
        gotoxy(30,10); printf("Panjang : ");
        scanf("%f",&panjang);
        gotoxy(30,11); printf("Lebar    :" );
        scanf("%f",&lebar);
        keliling = 2 * panjang * lebar;
        luas = panjang * lebar;
        gotoxy(30,13); printf("Keliling = %8.2f ", keliling);
        gotoxy(30,14); printf("Luas    = %8.2f', luas );
        return(0);
}
```

exit()

- Library function exit(), jika dipanggil akan menyebabkan program exit dan kembali ke prompt DOS.
- Contoh:

```
#include <stdio.h>
void cetak(){
    char str[]="Selamat Datang di ITP\n";
    printf(str);
    exit(0);
    printf("%s",str);
}

int main(){
    cetak();
    return 0;
}
```

Latihan

```
int x,y,z,w;
x=scanf("%d %d %d",&y,&z,&w);
```

1. Apa yang terjadi jika pada program diatas di input 2 nilai integer dari keyboard ?
2. Berapa nilai x jika diinput dari keyboard 3 buah character ?

Latihan

```
char ss1[40];
char ss2[40];
x=scanf("%s %s",ss1,ss2);
```

1. Apa isi variabel ss1 dan ss2, jika dari keyboard diinput string "Selamat Pagi Pak" ?
2. Berapa nilai x jika diinput dari keyboard : "Kelas 1PAT"

Latihan

```
char ss[40];
scanf("%4s", ss);
```

1. Apa isi variabel ss, jika dari keyboard diinput string "Selamat Pagi" ?

```
char ch;
ch = getchar();
```

2. Apa isi variabel ch, jika dari keyboard di-input : ITP

```
char ch1, ch2;
ch1 = getchar(); //masukkan kata ITP disini !
ch2 = getchar();
```

3. Apa isi variabel ch1 dan ch2, jika dari keyboard di-input : ITP

Latihan

- Buatlah program untuk menerima input dari keyboard, berupa nilai :
  - Tugas Mandiri (NTM)
  - Nilai UTS (NUTS)
  - Nilai UAS (NUAS)
- Hitung dan tampilkan nilai akhir dengan rumus :

NA = 20 % * Nilai TM + 30 % Nilai UTS + 50 % Nilai UAS

Latihan

```
#include <stdio.h>
int main()
{
  char nama[40];
  int nim;
  char jk;
  printf("Nama:"); scanf("%[^\n]",nama);
  printf("Nim:"); scanf("%d",&nim);
  printf("Jenis Kelamin (L/P):"); jk=getchar();
  return 0;
}
```

Perhatikan Program diatas!

Setelah memasukkan nama dan nim dari keyboard, program langsung keluar, instruksi jk=getchar(); seolah olah tdk pernah dieksekusi, jelaskan mengapa demikian ?

Latihan

```
#include <stdio.h>
int main(){
    char ss[]="10 % 3 = 1\n";
    char str[]="Selamat Datang di ITP\n";
    printf(ss);
    printf("%s",ss);
    printf(str);
    printf("%s",str);
    return 0;
}
```

Apakah output dari program diatas ?

Latihan

- Jelaskan fungsi/kegunaan dari Standard library function sbb:
  - sscanf
  - sprintf
  - fflush
  - cprintf
  - Cscanf

- Jelaskan maksud dari nama sbb:
  - stdin
  - stdout
  - stderr

Pertemuan 5

Operand dan Operator

Learning Outcomes

Pada akhir pertemuan ini, diharapkan mahasiswa akan mampu :

- Membuat statement / instruksi dengan berbagai operator yang ada pada bahasa C

Outline Materi

Jenis-jenis Operator

- Operator Penugasan (assignment)
- Operator Logika
- Operator Aritmetika
- Operator Relasional
- Operator Bitwise

Operator dan Operand

- Operator adalah simbol yang mengolah nilai pada operand dan menghasilkan satu nilai baru.
- Contoh :

  C = A + B

  (= dan + adalah operator, sedangkan A, B dan C adalah operand)
- Operator dapat dibedakan menjadi tiga berdasarkan jumlah operand-nya, yaitu :
  - **Unary operator**
  - **Binary operator**
  - **Ternary operator**
- Unary operator memerlukan satu operand, dan binary operator memerlukan dua operand, sedangkan Ternary operator memerlukan 3 operand.

Operator

- Berdasarkan jenis operasinya, operator dalam bahasa C dapat dikelompokkan :
  - Operator Penugasan (assignment operator)
  - Operator Logika
  - Operator Aritmatika
  - Operator Relasional
  - Operator Bitwise
  - Operator Pointer

Operator Penugasan (assignment)

- Termasuk Binary operator
- Digunakan untuk memberikan nilai kepada suatu Operand. Sintak sbb:

Operand1 = Operand2;

- Operand di sebelah kiri (Operand1) harus memiliki address (L-Value) seperti variabel, sedangkan operand disebelah kanan (Operand2) bisa suatu konstanta, variabel lain, ekspresi atau fungsi.

Operator Penugasan

- Contoh

```
x = 2;                  // konstanta
x = y;                  // variabel lain
x = 2 * y;          // ekspresi
x = sin (y);            // fungsi
```

- Tipe hasil operasi disesuaikan dengan tipe operand sebelah kiri.

```
int x = 7/2;        /*nilai x sama dgn 3 */
float y = 3;        /*nilai y sama dengan 3.000 */
```

Operator Aritmatika

- Digunakan untuk melakukan operasi matematis

Simbol	Fungsi	Contoh
+	Penambahan	x = y + 6;
-	Pengurangan	y = x – 5;
*	Perkalian	y = y * 3;
/	Pembagian	z = x/y;
%	Modulo	A = 10 % 3;
++	Increment	x++;
--	Decrement	z--;
()	Menaikan Priority	x=(2+3)*5

Operator Aritmatika

- Modulo:
  - Simbol : %
  - Termasuk Binary operator
  - Untuk mencari sisa hasil bagi
  - N % 2, dapat digunakan untuk menguji apakah integer n genap atau ganjil
    - N % 2 = 0 $\rightarrow$ n GENAP
    - N % 2 = 1$\rightarrow$ n GANJIL

- Increment dan Decrement:
  - Simbol : ++(increment), --(decrement)
  - Termasuk unary operator
  - Menaikkan (++) dan menurunkan (--) nilai variabel dengan nilai 1
  - Posisinya bisa didepan (pre) atau dibelakang (post) variabel

Operator Aritmatika

- Contoh:

```
N++;  //post increment
++N;  //pre increment
N--;   //post decrement
--N;   //pre decrement
```

- Jika statement increment stand alone. Maka N++; atau ++N; sama artinya N=N+1;
- Jika statement decrement stand alone. Maka N--; atau --N; sama artinya N=N-1;

Operator Aritmatika

- **Contoh:**

```
#include <stdio.h>
int main ()
{
        int x = 44;  int y = 44;
        ++x;
        printf(”x = %d\n”, x);                  /* hasilnya 45 */
        y++;
        printf(”y = %d\n”, y);                  /* hasilnya 45 */
}
```

Operator Aritmatika

- Jika ++n dan n++ sebagai statement yang terikat dalam ekspresi lainnya (sub expresi), keduanya mempunyai arti yang berbeda.
- ++n -> n ditambah 1, baru diproses terhadap ekspresinya
- n++ -> n langsung diproses terhadap ekspresinya tanpa ditambah 1 terlebih dahulu, pada saat selesai baru n ditambah 1

```
int main () {
        int x=44; int y = 44; int z;
        z =  ++x;                       /* z, x nilainya 45 */
        z =  y++;                       /* z nilainya 44 dan y nilainya 45 */
        return(0);
}
```

Operator Aritmatika

- Setiap ekspesi yang berbentuk :

  <Variabel> = <Variabel> <Operator><Exp>;

  dapat diganti dengan :

  <Variabel> <Operator> = <Exp>;

- Operator ini sering disebut dengan **Combined Operator.**

Ekspresi	**Dapat diganti dengan**
a = a + b;	a += b;
a = a – b;	a -= b;
a = a * b;	a *= b;
a = a / b;	a /= b;
a = a % b;	a %= b;
a = a ^ b ;	a ^= b;

Contoh soal :

x *= y +1; artinya sama dengan :

A. x = x * (y + 1);
B. x = x * y + 1;
C. x = x + 1 * y;
D. x = (x + 1) * y;

Jawab: A

Operator Relasional

- Digunakan untuk membandingkan dua nilai, dan hasilnya TRUE atau FALSE

Simbol	Fungsi
= =	Sama Dengan
!=	Tidak Sama Dengan
<	Lebih Kecil Dari
>	Lebih Besar Dari
<=	Lebih Kecil atau Sama Dengan
>=	Lebih Besar atau Sama Dengan
?:	Conditional assignment

Operator Relasional

- TRUE dalam bahasa C nilainya Tidak sama dengan NOL
- FALSE dalam bahasa C nilainya sama dengan NOL
- Sedangkan nilai TRUE yang diset oleh program C saat run time nilainya sama dengan 1

Operator Relasional

- Contoh :

```
#include<stdio.h>
int main()
{
    int x=5,y=6;
    if ( x == y) printf("%d sama dengan %d\n",x,y);
    if ( x != y) printf("%d tidak sama dengan %d\n",x,y);
    if ( x < y) printf("%d lebih kecil daripada %d\n",x,y);
    if ( x > y) printf("%d lebih besar daripada %d\n",x,y);
    if ( x <= y) printf("%d lebih kecil atau sama dengan %d\n",x,y);
    if ( x >= y) printf("%d lebih besar atau sama dengan %d\n",x,y);
    return(0);
}
```

Operator Relasional

- Contoh:

```
int x;
x = (20 > 10);    //nilai x sama dgn 1
X= (20 == 10);   //nilai x sama dgn 0
```

Conditional Expressions

- Stetement sbb:

  if(a > b) z = a;

  else z = b;

- Statement diatas bisa diganti dengan conditional expression.
- Conditional expression menggunakan ternary operator ?:
- Sintak : exp1 ? Exp2 : exp3;
- Contoh yang sama artinya dgn statement diatas:

  z = (a > b) ? a : b;

Conditional Expressions

- Contoh :

```
int main () {
        int kode, diskon=0;
        kode=1;
        diskon = (kode == 1) ? 30 : 10;
        printf("Diskon item = %d \n",diskon);
        Return(0);
}
```

```
int main () {
        int bil, abs;
        bil = 50;
        abs = (bil1 > 0) ? bil : - bil;
        printf("%d \n", bil);

        bil = -50;
        abs = (bil1 > 0) ? bil : - bil;
        printf("%d \n", bil);
        return(0)
}
```

Operator Logika

- Digunakan untuk melakukan operasi logika

Simbol	Fungsi
&&	AND
||	OR
!	NOT

- Hasil operasi bernilai **TRUE** atau **FALSE,** seperti Table Kebenaran berikut:

A	B	!A	A && B	A || B
True	True	False	True	True
True	False	False	False	True
False	True	True	False	True
False	False	True	False	False

Operator Logika

- Operand pada operator logika dianggap operand yang mempunyai nilai TRUE atau FALSE.
- Contoh:

```
int x=5; int y=0;
x && y;  //FALSE
(x > y) && (y>=0);  //TRUE
```

Operator Bitwise

Simbol	Fungsi	Contoh
&	AND	A & B
|	OR	A | B;
^	XOR	A ^ B;
~	Complement 1	~A;
>>	Shift Right	A >> 3;
<<	Shift Left	B << 2;

Operator Bitwise

- Operasi nya bit per bit
- Contoh :

```
int A=24; int B=35; int C;
C=A & B;  //nilai C = 0
C=A | B;   //nilai C = 59
```

A=24 Binernya: 011000

B=35 Binernya: 100011

Bit per bit di AND kan
Hasilnya : 000000 desimalnya 0

JIka Bit per bit di OR kan
Hasilnya : 111011 desimalnya 59

Operator Bitwise

- Dua buah bit di-XOR-kan akan menghasilkan 1 jika kedua bit tersebut berbeda, dan akan menghasilkan 0 jika kedua bit tersebut sama.

- Contoh:

```
int A,B=45;
A=B^75; //Nilai A=102
```

Desimal 45 binernya: 0101101

Desimal 75 binernya: 1001011

Kemudian bit per bit di XOR kan maka hasilnya: 1100110 atau (102 desimal).

Operator Bitwise

- Untuk membuat complement-1, maka setiap bit yg nilainya 0 dirubah menjadi 1 dan sebaliknya.
- Contoh:

```
int A, B=0xC3;
A=~B;  //nilai A=0x3C;
```

0xC3 binernya: 1100 0011

Di-komplemen-1 hasilnya:

0011 1100 atau dlm notasi hexadecimal menjadi 3C

Untuk menulis bilangan dlm notasi hexadesimal pada C digunakan 0x diawal bilangan tersebut.

Operator Bitwise

- Contoh:

```
int A, B=78;
A = B >> 3; //nilai A=9
A = B << 2; //nilai A=312
```

78 binernya: 0100 1110

Geser kekanan yang ke-1 : 0010 0111

Geser kekanan yang ke-2 : 0001 0011

Geser kekanan yang ke-3 : 00001001 => 9 desimal

78 binernya: 0100 1110

Geser kekiri yang ke-1 : 0100 11100

Geser kekiri yang ke-2 : 0100 111000 => 312 desimal

Operator Pointer

- **Pointer operator terdiri dari :**
  - & (address of)
  - * (value of)

AKAN DIBAHAS PADA TOPIK POINTER

Precedence dan Associativity

- Setiap Operator memiliki Presedensi dan assosiativitas.

- Presedensi menentukan urutan pengerjaan / pengeksekusian operator berdasarkan tingkat / prioritas. Operator yang memiliki tingkat presedensi lebih tinggi akan dikerjakan lebih dahulu.

- Assosiativitas menentukan urutan pengerjaan / pengeksekusian operator berdasarkan lokasinya dalam sebuah ekspresi, (apakah dari kiri atau dari kanan).

- Assosiativitas berlaku untuk operator-operator yang memiliki presedensi yang sama.

Tabel Precedence dan Associativity

Precedence	Operator	Associativity
1	**:: (unary)** **::(binary)**	right to left left to right
2	**() [] -> .**	left to right
3	**++ -- + - ! ~ (type)** **& sizeof**	right to left
4	**.* ->***	left to right
5	*** / %**	left to right
6	**+ -**	left to right
7	**<< >>**	left to right
8	**< <= > >=**	left to right
9	**== !=**	left to right
10	**&**	left to right
11	**^**	left to right
12	**|**	left to right
13	**&&**	left to right
14	**||**	left to right
15	**?:**	right to left
16	**= += -= *= /= &= ^=** **|= <<= >>=**	right to left
17	**,**	left to right

Pengaruh Precedence dan Associativity

```c
void main()
{
    float y, x;
    x = 10/3*3;
    y = 3*10/3;
    printf("x = %f\n",x); // x = 9.00
    printf("y = %f\n",y); // y = 10.00
}
```

```c
#include "stdio.h"

void main()
{
    int y=10,x=10;
    y += x = 20;
    printf("x = %d\n",x); // x = 20
    printf("y = %d\n",y); // y = 30
}
```

Latihan

- Jika semua variabel bertipe integer, maka tentukan nilai A sbb:

  B=23; C=12; D=32; E=0;

  1. A = B && E;
  2. A = B & C;
  3. A = C || D;
  4. A = B | D;
  5. A = B > 2;
  6. A = B >> 2;
  7. A = C < 3;
  8. A = C << 3;
  9. A = B = C;
  10. A = B == C;

Latihan

- Sebutkan contoh operator yang termasuk :
  1. Unary Operator
  2. Binary Operator
  3. Ternary Operator

- Sebutkan beberapa operator yang bisa berfungsi sebagai unary dan binary operator, dan berikan contohnya.

Latihan

```
#include <stdio.h>
int main()
{
    int x=10, y=6;
    x *= 5 + y;
    printf("%d\n",x);
   return(0);
}
```

Berapa nilai x yang dicetak dilayar monitor ?

Latihan

```
#include "stdio.h"
void main()
{
     float x;
     x=7/2;
     printf("x=%f\n",x);
     x=7.0/2;
     printf("x=%f\n",x);
     x=7/2.0;
     printf("x=%f\n",x);
     x=7.0/2.0;
     printf("x=%f\n",x);
}
```

Berapa nilai x yang ditampilkan di layar monitor ?

Latihan

```
#include "stdio.h"
void main()
{
     int y=33,x=45;
     y = y && x; printf("%d\n",y);
     y = y &  x; printf("%d\n",y);

     y=33,x=45;
     y = y &  x; printf("%d\n",y);
     y = y && x; printf("%d\n",y);
}
```

Berapa nilai y yang ditampilkan di layar monitor ?

Latihan

```c
#include "stdio.h"

void main()
{
    int y, n = 10;
    y = ++n;
    printf("y = %d\n",y);
}
```

```c
#include "stdio.h"

void main()
{
    int y, n = 10;
    y = n++;
    printf("y = %d\n",y);
}
```

```c
#include "stdio.h"

void main()
{
    int n = 10;
    n++;
    printf("n = %d\n",n);
}
```

```c
#include "stdio.h"

void main()
{
    int n = 10;
    ++n;
    printf("n = %d\n",n);
}
```

Apa yang ditampilkan di layar monitor oleh 4 program diatas ?

Latihan

z = (n > 0) ? f : b;

Perhatikan conditional expression diatas, bolehkan f dan b memiliki tipe data yang berbeda ?

Pertemuan 6

Operasi Looping (Pengulangan)

Learning Outcomes

Pada akhir pertemuan ini, diharapkan mahasiswa akan mampu :

- Mendemonstrasikan penulisan program dengan struktur kendali pengulangan (looping)

Outline Materi

Operasi Pengulangan (looping)

- Konstruksi for, while dan do-while
- Nested dan infinite loop
- Compound atau Block statement
- Break dan Continue

Operasi Repetisi

- Beberapa instruksi diulang untuk suatu jumlah pengulangan yang tertentu
- Jumlah pengulangan dapat diketahui sebelumnya atau ditentukan dalam perjalanan program.
- Operasi repetisi :
    - for
    - while
    - do-while

Operasi Repetisi : for

- Konstruksi for

for(exp1; exp2; exp3) statement;
atau:
for(exp1; exp2; exp3){
    statement1;
    statement2;
    .....
}

exp1 : adalah ekspresi untuk inisialisasi,
exp2 : adalah ekspresi conditional
exp3 : adalah ekspresi increment atau decrement

exp1, exp2 dan exp3 adalah sifatnya optional (boleh ada boleh tidak ada).

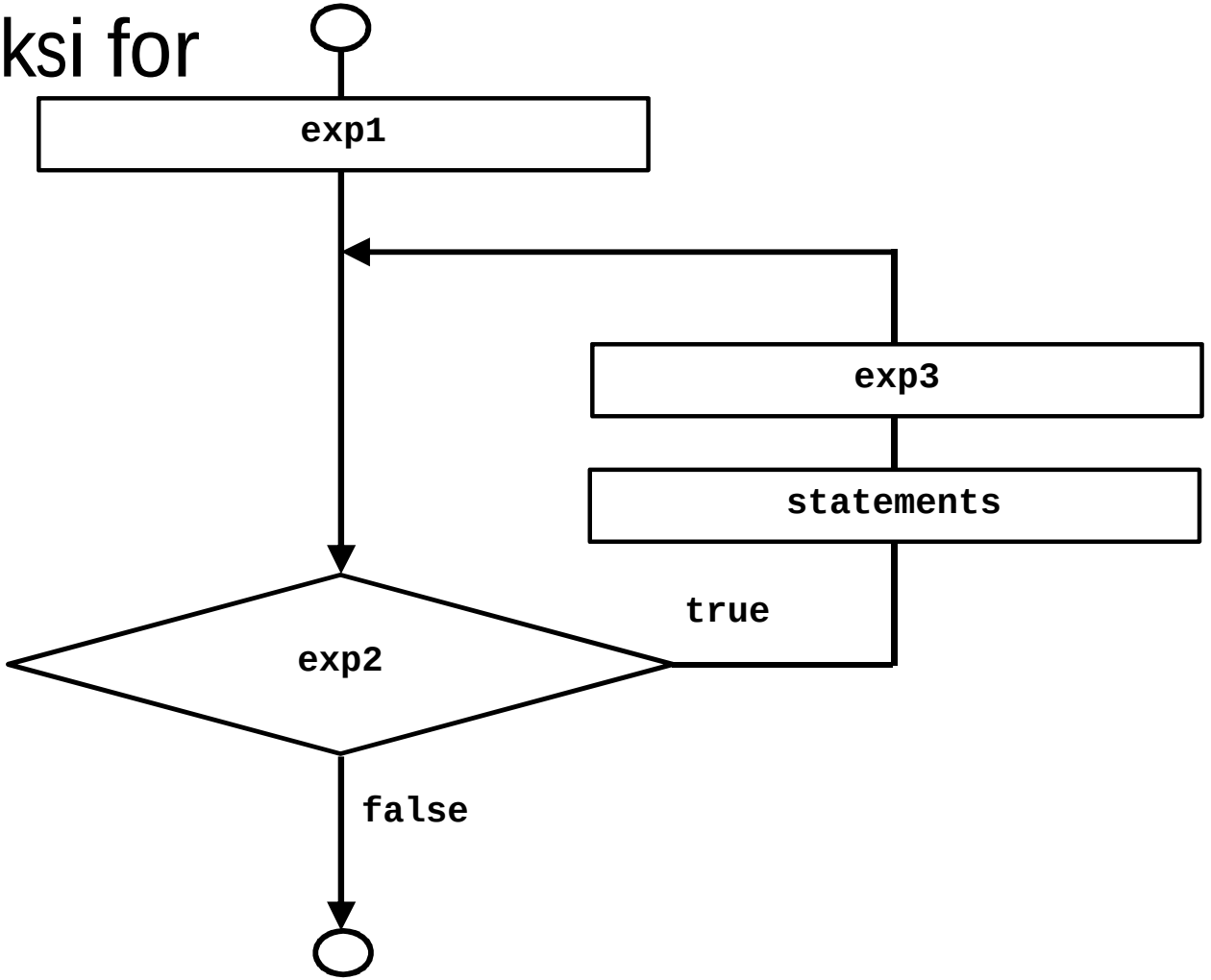
Operasi Repetisi : for

- exp1 dan exp3 boleh terdiri dari beberapa ekspresi yang dipisahkan dengan koma.
- Contoh:

```
void reverse(char ss[])
{
    int c,i,j;
    for(i=0, j=strlen(ss)-1; i<j; i++, j--){
        c=ss[i];
        ss[i]=ss[j];
        ss[j]=c;
    }
}
```

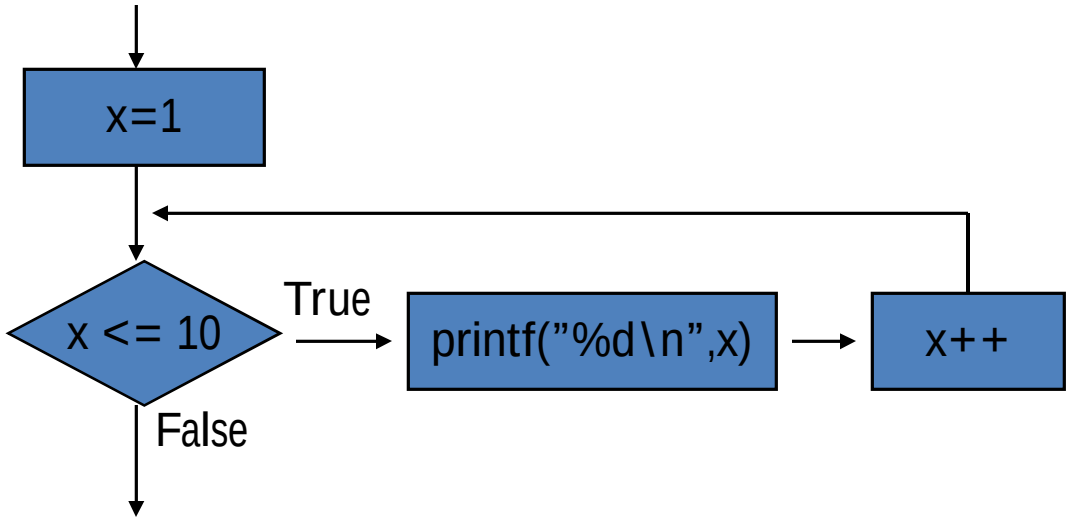
Operasi Repetisi : for

- Konstruksi for



Operasi Repetisi : for

- Contoh :
  - ✓ for (x=1; x <= 10; x++) printf(“%d\n”,x);



Operasi Repetisi : for

- Contoh :
- Program mencetak angka dari 1 sampai 10

```
#include<stdio.h>
int main()
{
   int x;
   for( x = 1 ;  x <= 10 ;  x++ ) printf( "%d\n", x );
   return(0);
}
```

- Program mencetak angka dari 10 sampai 1

```
#include<stdio.h>
int main()
{
   int x;
   for( x = 10 ;  x >= 1 ;  x-- ) printf( "%d\n", x);
   return(0);
}
```

Operasi Repetisi : for

- Contoh penggunaan ***for*** dalam program.
- Ingin mengetahui rata-rata pengeluaran uang untuk bensin selama satu bulan, dengan data sbb :

Minggu	Pengeluaran
1	Rp. 32.000,-
2	Rp. 29.000,-
3	Rp. 33.000,-
4	Rp. 24.000,-

Algoritma :
1. Kosongkan variabel jumlah
2. Baca data dari keyboard dan simpan pada variabel Data
3. Tambahkan Data ke jumlah
4. Ulangi 2 dan 3 sebanyak 4 kali
5. Rerata = Jumlah / 4

Operasi Repetisi : for

- Contoh :

```
/*--------------------------------*/
/* Program Hitung_Rerata  */
/*--------------------------------*/
#include<stdio.h>
int main()
{
      float data, jumlah, rerata;
      int x;
      jumlah = 0.0;
      for( x = 1; x <= 4; x++) {
            printf("Data minggu ke-%d :",x);
            scanf("%f",&data);
            jumlah = jumlah + data;
      }
     rerata = jumlah / 4;
     printf("Rerata = Rp %8.2f\n",rerata);
     return(0);
}
```

Operasi Repetisi : for

- Infinite Loop

  Untuk membuat infinite loop dapat dilakukan dengan menggunakan “for-loop”, dengan menghilangkan ketiga parameter (exp1, exp2, exp3) yang ada pada for loop. Untuk keluar dari loop dapat digunakan statement **break**.

- Nested Loop

  Jika didalam sebuah perulangan terdapat statement yang berisi perulangan. Perulangan akan dijalankan dari yang paling dalam.

Operasi Repetisi : for

NESTED LOOP

Bahasa C

```
int x, y;
for (x=1;x<=5;x++)
    for (y=5; y>=1; y--)
        printf("%d %d ",x,y);
```

Bahasa C++

```
for (int x=1;x<=5;x++)
    for (int y=5; y>=1; y--)
        printf("%d %d ",x,y);
```

Output :

1 5 1 4 1 3 .. 2 5 2 4 .. 5 1

Operasi Repetisi : for

- Contoh :

```
/*--------------------------------
Program Tabel_Kebenaran;
--------------------------------*/
#include <stdio.h>
int main()
{
      int P,Q;
      printf("============================\n");
      printf("P  Q   P or Q   P and Q   Not P  P xor Q\n");
      printf("============================\n");
      for(P=1; P>=0; P--)
         for(Q = 1; Q>=0; Q--)
            printf("%4d %4d %4d %4d %4d %4d\n", P, Q, P||Q, P&&Q, !P, P^Q );
      printf("============================\n");
}
```

Operasi Repetisi : while

- Sintaks :

while (exp) statements;

atau :

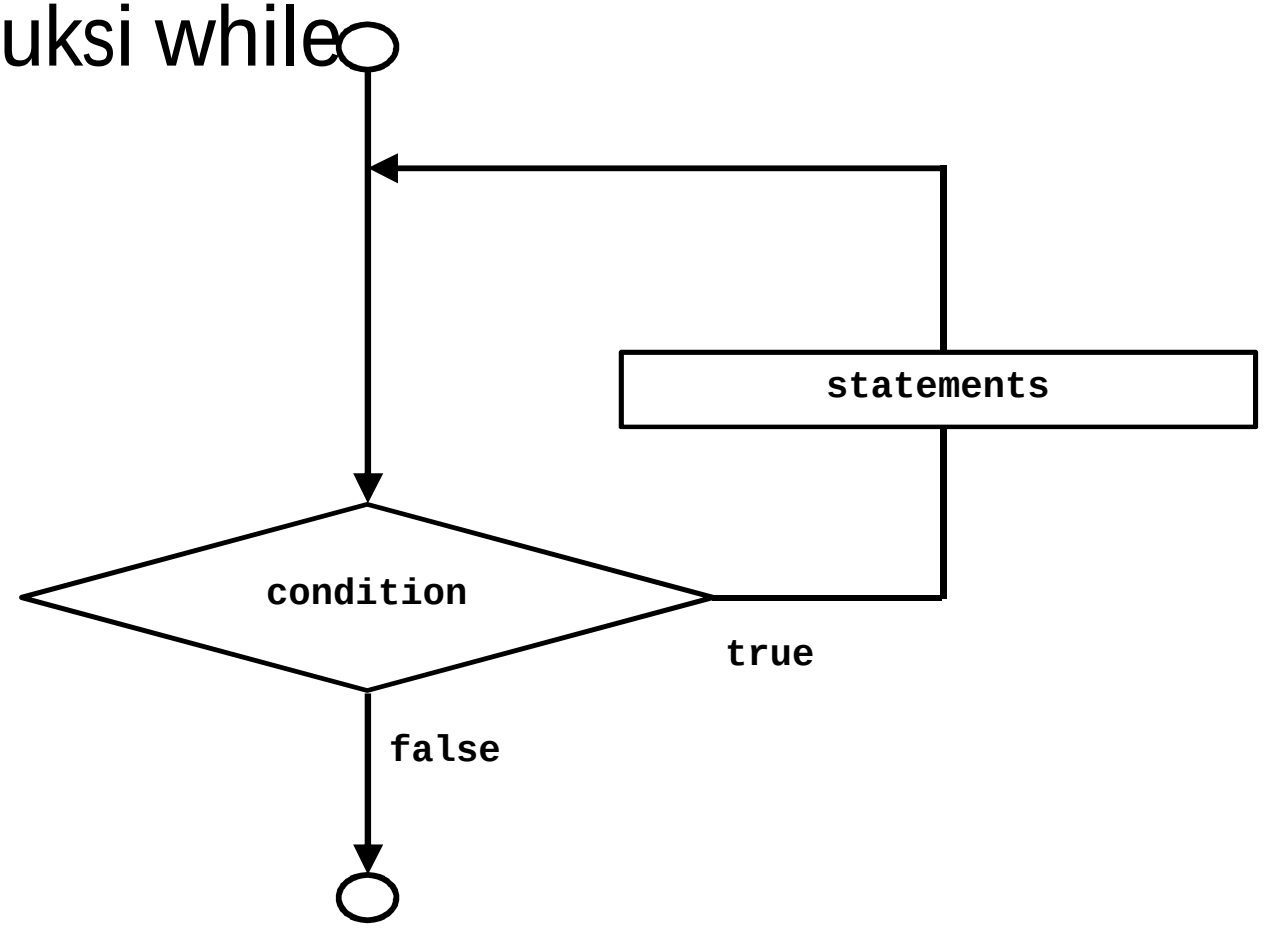
```
while(exp){
   statement1;
   statement2;
   .....
}
```

Contoh :

```
int counter = 1;
while ( counter <= 10 ) {
        printf( "%d\n", counter );
        ++counter;
}
```

Operasi Repetisi : while

- Konstruksi while

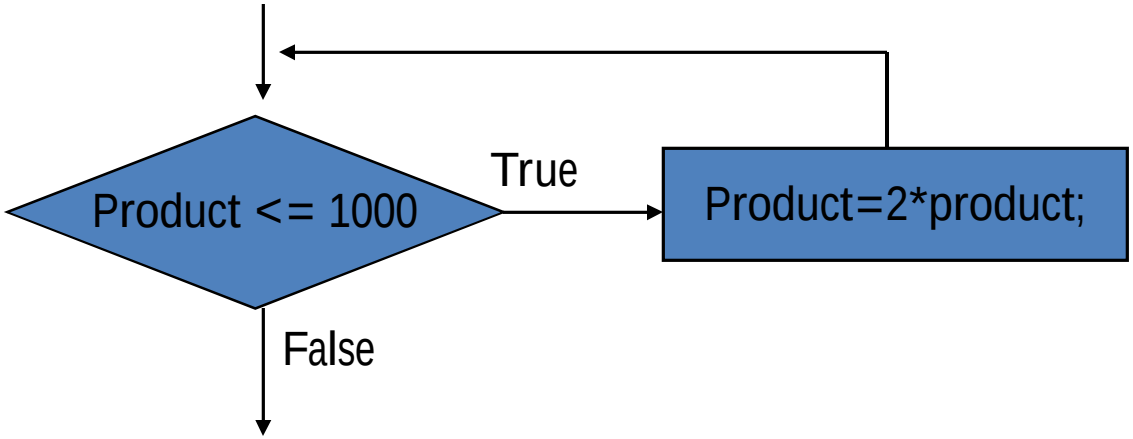


Operasi Repetisi : while

- ***exp*** adalah ekspresi boolean yang menghasilkan nilai True (tidak nol) atau False (sama dengan nol).
- Statement di-eksekusi berulang-ulang selama ***exp*** tidak Nol.
- Pengetesan ***exp*** dilakukan sebelum statements dilaksanakan.

Operasi Repetisi : while

- Contoh :
  - ✓ **while(product <= 1000) product = 2*product;**

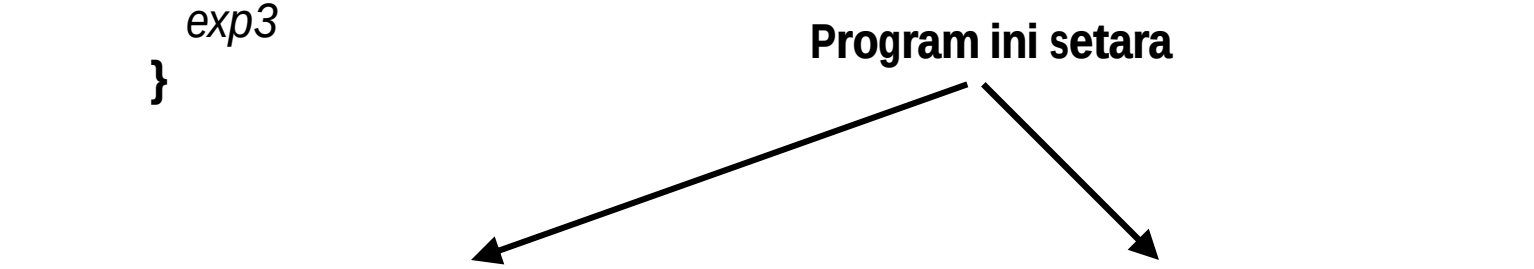


Operasi Repetisi : while

- Perintah for setara dengan while sbb:

```
exp1;
while ( exp2 ) {
  statement1;
  statement2;
  ....
  exp3
}
```

Program ini setara



```
#include<stdio.h>
void main() {
  int x;
  for( x = 1 ;  x <= 10 ;  x++ )
      printf( "%d\n", x );
}
```

```
#include<stdio.h>
void main() {
  int x = 1;
  while (x<=10) {
      printf( "%d\n", x );
      x++;
  }
}
```

Operasi Repetisi : do-while

- Sintaks :

```
do{
    < statements >;
} while(exp);
```

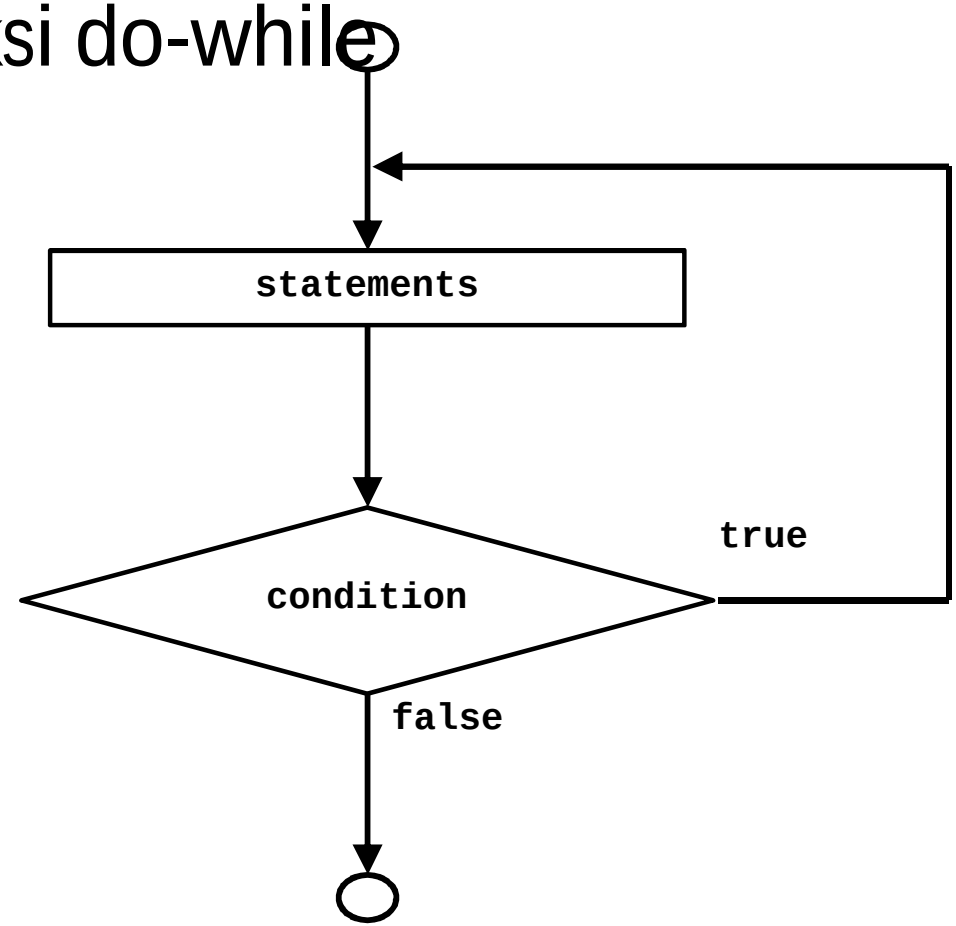
Contoh :

```
int counter=0;
do {
        printf( "%d  ", counter );
        ++counter;
} while (counter <= 10);
```

- Selama nilai exp true maka statement dieksekusi berulang-ulang.
- Pengetesan exp dilakukan setelah meng-eksekusi statement.

Operasi Repetisi : do-while

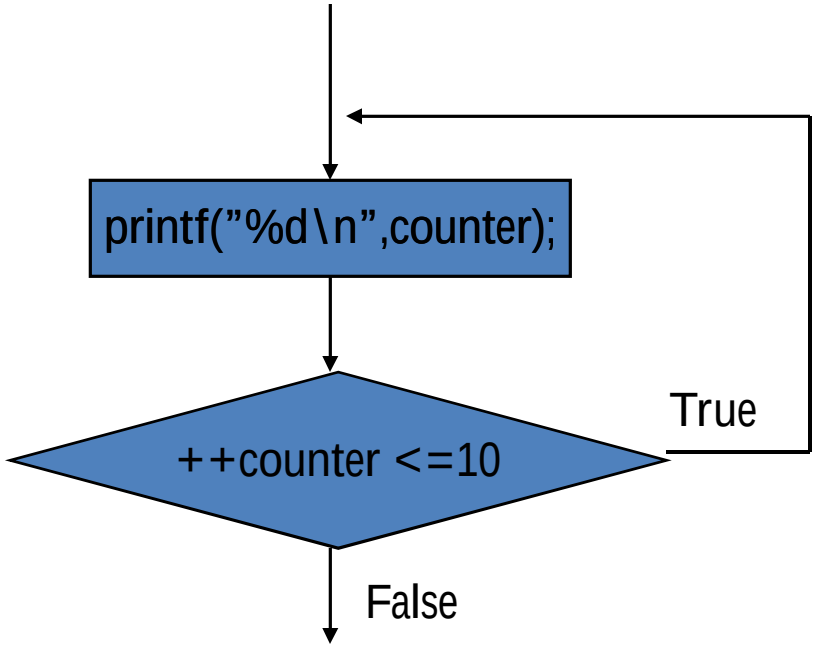
- Konstruksi do-while



Operasi Repetisi : do-while

- Contoh:

```
do{
        printf("%d\n",counter);
} while(++counter <=10);
```



Operasi Repetisi

- Pada konstruksi while, statement atau blok statement mungkin tidak akan pernah dilaksanakan, bila nilai ekspresi boolen (exp) bernilai False, karena sebelum konstruksi pengulangan dimasuki nilai ekspresi boolean (exp) terlebih dahulu diuji.
- Pada konstruksi do-while statement atau blok statement pasti dikerjakan paling sedikit satu kali, karena ekspresi boolean baru diuji pada akhir blok pengulangan.

Operasi Repetisi

- Beberapa cara untuk menghentikan pengulangan, ini dapat dilakukan dengan menambah :
- Sentinel atau Pembatas dengan kode khusus
- Pertanyaan, Apakah pengulangan akan dilanjutkan.

Operasi Repetisi

- Contoh : Cara ‘Pertanyaan’ pada konstruksi While

```
#include <stdio.h>

int main()
{
        int panjang, lebar, luar; char ulang;
        printf(”Teruskan Perhitungan ? (Y/T) :”);
        scanf(”%c”,&ulang);
        while((toupper(ulang)) == ’Y’) {
                printf(”Panjang : ”);
                scanf(”%d”,&panjang);
                printf(”Lebar     : ”);
                scanf(”%d”,&lebar);
                luas = panjang * lebar;
                printf(”Luas = %d\n\n”,luas);
                printf(”Teruskan Perhitungan?(Y/T):”);
                scanf(”%c”,&ulang);
        }
        return(0);
}
```

Operasi Repetisi

- Contoh : Cara ‘sentinel’ pada konstruksi do-while

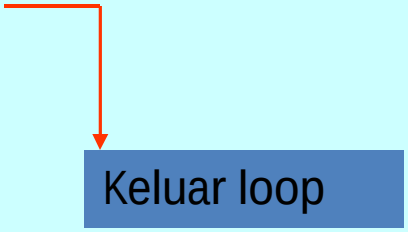
  Sebagai sentinel, digunakan nilai 0 pada variabel panjang atau variabel lebar.

```
#include <stdio.h>
int main()
{
      int panjang,lebar,luas; char ulang;
      do{
            printf(“Panjang [0=selesai] : “);
            scanf(“%d”,&panjang);
            printf(“Lebar     [0=selesai] : “);
            scanf(“%d”,&lebar);
            luas = panjang * lebar;
            printf(“Luas = %d\n”,luas);
      } while((panjang != 0) && (lebar != 0));
      return(0);
}
```

Operasi Repetisi

Break pada loop menyebabkan program keluar dari loop tersebut

```
#include<stdio.h>
int main() {
    int x = 1;
    while (x<=10) {
        printf( "%d\n", x );
        x++;
        break;
    }
    return 0;
}
```



Break vs Continue

- **break**:
  - Digunakan untuk keluar dari loop (for, while dan do-while)
  - Digunakan untuk keluar dari switch

- **continue**: skip sisa instruksi dalam loop, dan eksekusi loop berjalan ke tahap selanjutnya

Break vs Continue

- Contoh :

```
#include <stdio.h>
int main() {
        int x;
        for(x=1; x<=10; x++) {
                if (x == 5) continue;
                printf("%d ", x);
        }
        return 0;
}
```

```
Output : 1 2 3 4 6 7 8 9 10
```

Break vs Continue

- Contoh :

```
#include <stdio.h>
int main() {
        int x;
        for(x=1; x<=10; x++) {
                if (x == 5) break;
                printf("%d ", x);
        }
        return 0;
}
```

```
Output : 1 2 3 4
```

Break vs Continue

```
do {
   scanf("%f", &x);
   if(x<0) {
       printf"\nEror. Negatif");
       break;
   }
   /*Proses nonnegatif */
  ...
} while(exp);
```

```
do {
   scanf("%f", &x);
   if(x<0) {
       printf"\nEror. Negatif");
       continue;
    }
   /*Proses nonnegatif */
  ...
} while(exp);
```

Break vs Continue

- Contoh :

```
#include <stdio.h>
int main() {
        int x;
        for(x=1; x<=10; x++) {
                if (x == 5) goto exit;
                printf("%d ", x);
        }
  exit:
        return 0;
}
```

```
Output : 1 2 3 4
```

Latihan

```
for (i=k; i < n; i++)
   printf(”ITP\n”);
```

1. Jika k < n maka berapa kali kata ITP di cetak dilayar monitor ?
2. Jika k=n maka berapa kali kata ITP di cetak dilayar monitor ?
3. Jika k > n maka berapa kali kata ITP di cetak dilayar monitor ?

Latihan

```
for (i=k; i >= n; i--)
   printf(”ITP\n”);
```

1. Jika k < n maka berapa kali kata ITP di cetak dilayar monitor ?
2. Jika k=n maka berapa kali kata ITP di cetak dilayar monitor ?
3. Jika k > n maka berapa kali kata ITP di cetak dilayar monitor ?

Latihan

- Dapatkah infinite/forever loop ditulis menggunakan for loop, while loop dan do-while loop ?
- Jika p = pernyataan dan e = ekspresi, ubahlah loop for berikut menjadi loop while.

a.for(; e ;) p;

b. for(; ; e) p;

Latihan

- Bandingkan dua buah sintaks for berikut ini :

```
for (i=0, j=1; i<8; i++, j++) printf("%d + %d = %d\n", i, j, i+j);
```

```
for (i=0, j=1; i<8; ++i, ++j); printf("%d + %d = %d\n", i, j, i+j);
```

Jelaskan output dan perbedaannya !

Latihan

APA OUTPUT PROGRAM BERIKUT ??

```
#include <stdio.h>
int main()
{
   int x,y;
   for(x=1;x<=3;x++)
    for (y=3;y>=1;y--)
          printf("%d %d ",x,y);
    return 0;
}
```

```
#include <stdio.h>
int main()
{
   int x,y;
   for(x=1;x<=3;x++);
    for (y=3;y>=1;y--)
          printf("%d %d ",x,y);
    return 0;
}
```

awas! Ada titik koma

Latihan

APA OUTPUT PROGRAM BERIKUT ??

```
#include <stdio.h>
int main()
{
   int x,y;
   for(x=1;x<=3;x++)
    for (y=3;y>=1;y--);
          printf("%d %d ",x,y);
    return 0;
}
```

Awas! ada titik koma

```
#include <stdio.h>
int main()
{
   int x,y;
   for(x=1;x<=3;x++);
    for (y=3;y>=1;y--);
          printf("%d %d ",x,y);
    return 0;
}
```

awas! ada titik koma

Latihan

Jelaskan output program berikut ini :

```
#include<stdio.h>

int main() {
        int nilai = 0, jumlah = 0;
        while(nilai < 10) {
                jumlah += nilai;
                printf("\nNilai =%d, Jumlah =%d", nilai++, jumlah);
        }
       return 0;
}
```

Latihan

- Jelaskan output program berikut ini :

```
#include<stdio.h>

int main() {
     long bil, tmp, x=1;
     printf(”\nInput bil:”); scanf(”%d”, &bil);
     tmp=bil;
     while(bil >= 1) x*=bil--;
     printf(”\n%ld ! = %ld”,tmp, x);

     return 0;
}
```

Latihan

- Buat program untuk menampilkan bilangan ganjil dari 11 s/d 188, dengan menggunakan :
  - for
  - while
  - do - while

Latihan

- Diasumsikan hari 1 menyatakan senin, 2 - selasa, 3 - rabu,..., 7 – minggu. Buatlah sebuah program untuk menampilkan angka hari sebanyak n yang diinput dari keyboard. Perhatikan pola berikut :

  N = 3

  1 2 3

  N = 7

  1 2 3 4 5 6 7

  N = 10

  1 2 3 4 5 6 7 1 2 3

Latihan

Berapa kali kata Hello dicetak di layar monitor oleh potongan program dibawah ini !

```
int x ;
for(x=0 ; x<10 ; x++) printf("Hello\n");
```

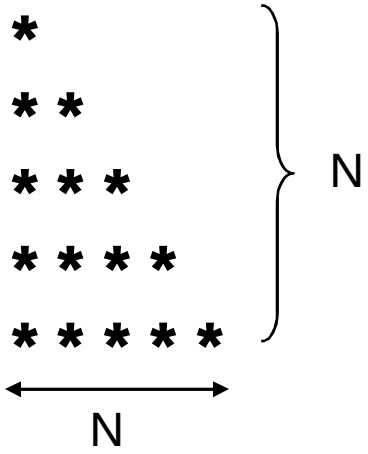
```
int x=0;
for( ; x<10 ; x++) printf("Hello\n");
```

```
int x=0;
for( ; ; x++){
      if(x<10) printf("Hello\n");
     else break;
}
```

```
int x=0;
for( ; ; ){
      if(x<10) printf("Hello\n");
      else break;
      x++;
}
```

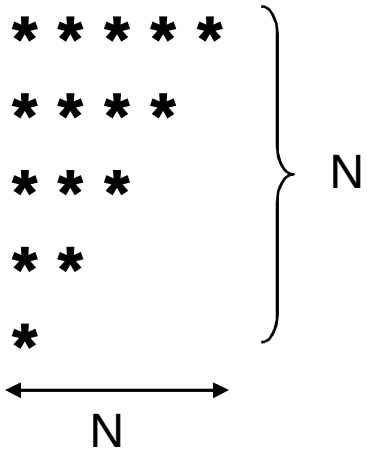
Latihan

- Buatlah program untuk menampilkan gambar seperti contoh dibawah, dimana N variabel integer yg nilainya di-input dari keyboard (dengan for, while atau do-while loop).



Latihan

- Buatlah program untuk menampilkan gambar seperti contoh dibawah, dimana N variabel integer yg nilainya di-input dari keyboard (dengan for, while atau do-while loop).



Latihan

Apa Output program dibawah ?

```
#include <stdio.h>
int main()
{
    int x,y,z;
    for(x=0; x<4; x++){
        for(y=0; y<3; y++){
            for(z=0; z<3; z++){
                if(z==2) break;
            }
            printf("Selamat\n");
        }
        printf("Datang\n");
    }
    printf("di ITP\n");
    getch();
    return(0);
}
```

Pertemuan 7

Operasi Seleksi

Learning Outcomes

Pada akhir pertemuan ini, diharapkan mahasiswa akan mampu :

- Mendemonstrasikan penulisan program dengan struktur kendali pemilihan (C3)

Outline Materi

Operasi Seleksi

- Konstruksi if dan if-else
- Konstruksi nested if-else dan switch-case
- Operator kondisional

Operasi Seleksi

- Dalam sebuah algoritma, seringkali beberapa instruksi harus dilaksanakan bila suatu persyaratan dipenuhi, dan sebaliknya.

- Dalam struktur penyeleksian, suatu instruksi atau sekelompok instruksi dapat dilewati, sementara instruksi lain dijalankan.

- Operasi seleksi :
    - if
    - if-else
    - switch-case

Operasi Seleksi : if

- **Konstruksi if**
- Sintaks :

```
if (ekspresi boolean) statement;
```

atau

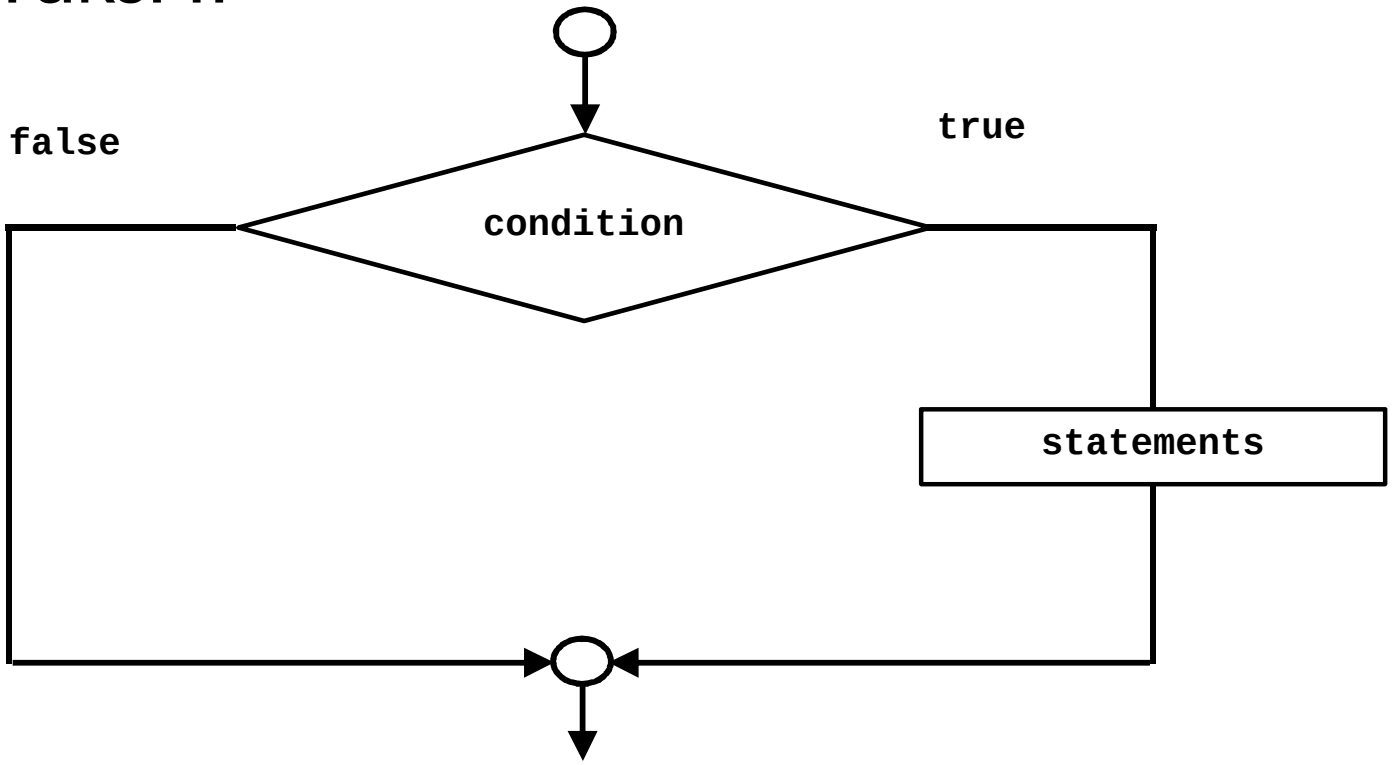
```
if (ekspresi boolean){
   statement1;
   statement2;
        ......
}
```

Blok statement

Jika ekspresi boolean menghasilkan nilai TRUE, maka statement atau blok statement akan dilaksanakan.

Operasi Repetisi : if

- Konstruksi if



Konstruksi : if-else

Sintaks :

```
if (ekspresi boolean) statement1;
else statement2;
```

atau

```
if (ekspresi boolean){
  statement1;
  statement2;
  ......
}
else {
  statement3;
  statement4;
  ...
}
```

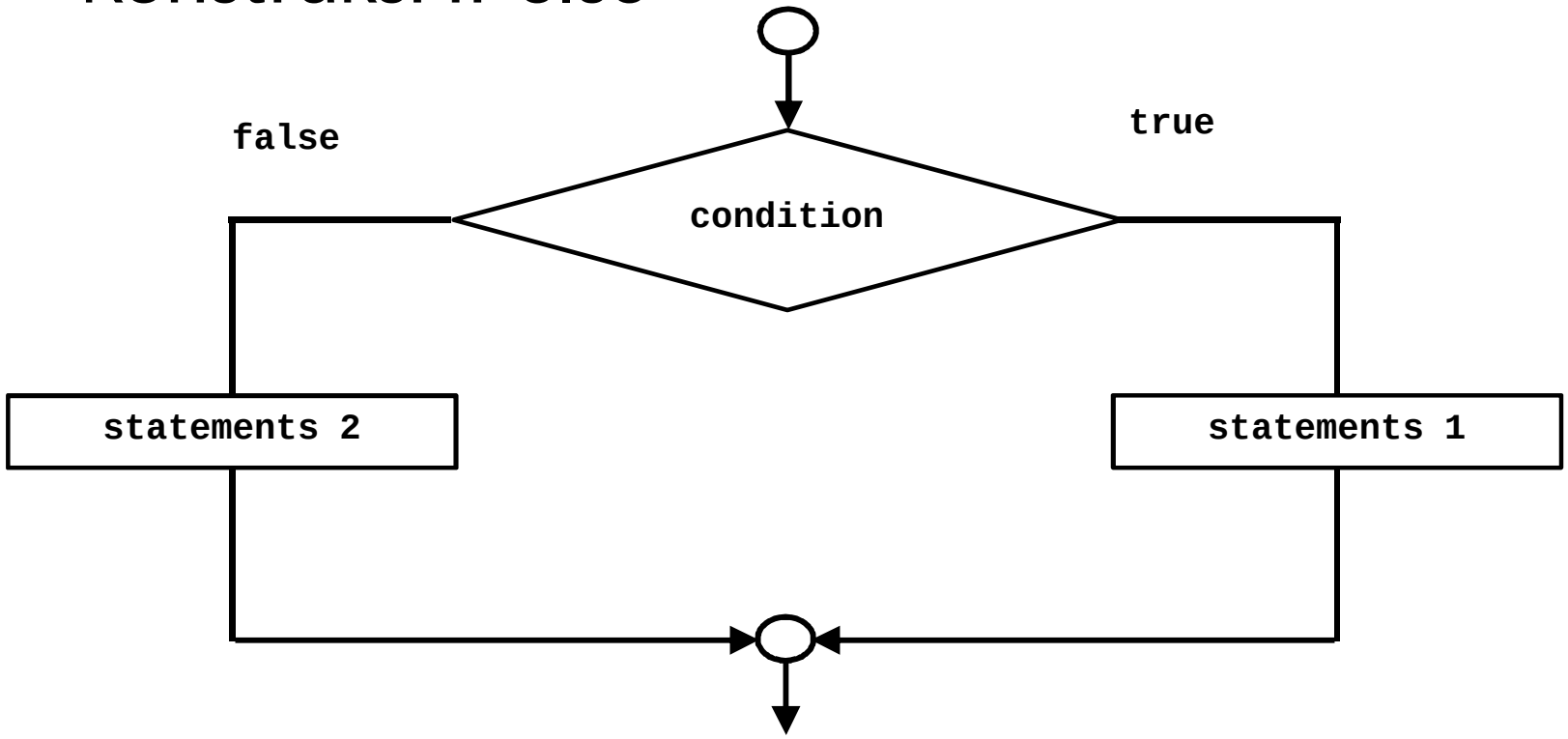
Blok statement1

Blok statement2

Jika ekspresi boolean menghasilkan nilai TRUE, maka statement1 atau blok statement1 yang akan dilaksanakan, jika tidak (FALSE) maka statement2 atau blok statement2 yang akan dilaksanakan.

Operasi Repetisi : if-else

- Konstruksi if-else



Operasi Seleksi

- Contoh Program untuk mendapatkan akar-akar dari persamaan kwardrat.
- Algoritma :
    1. Dapatkan koefisien a, b, dan c dari keyboard
    2. Hitung diskriminan d = b*b – 4*a*c
    3. Bila d >= 0 maka hitung x1 dan x2
       Bila d < 0 maka akar imajiner dan stop
    4. Stop

Dapatkan x1, dengan rumus : $\frac{-b + \sqrt{d}}{2*a}$

Dapatkan x2, dengan rumus : $\frac{-b - \sqrt{d}}{2*a}$

Operasi Seleksi : if

- Contoh :

```
#include <stdio.h>
#include <math.h>
int main()
{
        float a,b,c,d,x1,x2;
        printf("Masukan koef. a : "); scanf("%f",&a);
        printf("Masukan koef. b : "); scanf("%f",&b);
        printf("Masukan koef. c : "); scanf("%f",&c);
        d = b*b - 4 * a * c;
        if (d >= 0){
                x1 = (-b + sqrt(d)) / (2 * a);
                x2 = (-b - sqrt(d)) / (2 * a);
                printf("x1=%f\n  x2=%f\n",x1,x2);
         }
        else printf("Akar Persamaan Imajiner");
        return 0;
}
```

sqrt() adalah fungsi untuk mencari akar suatu bilangan, dan didefinisikan pada <math.h>

Operasi Seleksi : if

Standard Library Function : sqrt()

- Contoh program diatas menggunakan fungsi sqrt().
- sintak :

  double sqrt(double *x* **);**
- Header file <math.h>
- Untuk mencari akar dari bilangan x, dimana x bilangan non-negative.
- Contoh:
  - z = sqrt(45.35);
  - Maka nilai z = 6.73

Operasi Seleksi : if

- Konstruksi if-else dapat digunakan secara bertingkat (nested)
- Contoh:

```
if (ekspresi boolean1)  statement1;
  else if (ekspresi boolean2)  statement2;
    else if (ekspresi boolean3)  statement3;
      .....
        else if (ekspresi booleanN)  statementN;
```

Operasi Seleksi : if

- Contoh :

```
/* Program Kalkulator */
#include<stdio.h>
int main()
 {
       float bil1, bil2;
       char op;
       while(1) {
              printf("\n Ketik bil1 operator  bil2, ( Contoh: 3 * 4) \n");
              scanf("%f %c %f", &bil1, &op, &bil2);
              if(op == '+') printf(" = %f", bil1 + bil2);
              else if(op == '-') printf(" = %f", bil1 - bil2);
             else if(op == '*') printf(" = %f", bil1 * bil2);
             else if(op== '/') printf(" = %f", bil1 / bil2);
             else{
                  printf("error: operator hanya +,-,* dan / \n");
                  break;
              }
      }
      return 0;
}
```

Penulisan if harus jelas maksudnya ...

Penulisan if yg kurang jelas

```
#include <stdio.h>
main()
{
    int suhu;
    printf("Input suhu ? (F):");
    scanf("%d",&suhu);
    if(suhu < 80)
        if(suhu > 30)
        printf("Panas\n");
    else
        printf("Sejuk\n");
}
```

Operasi Seleksi : switch-case

- Konstruksi **switch-case**
- Konstruksi ini digunakan sebagai pengganti konstruksi if-else, bila konstruksi if-else bertingkat terlalu jauh, sehingga menjadi sulit dibaca.
- Sintaks konstruksi switch-case

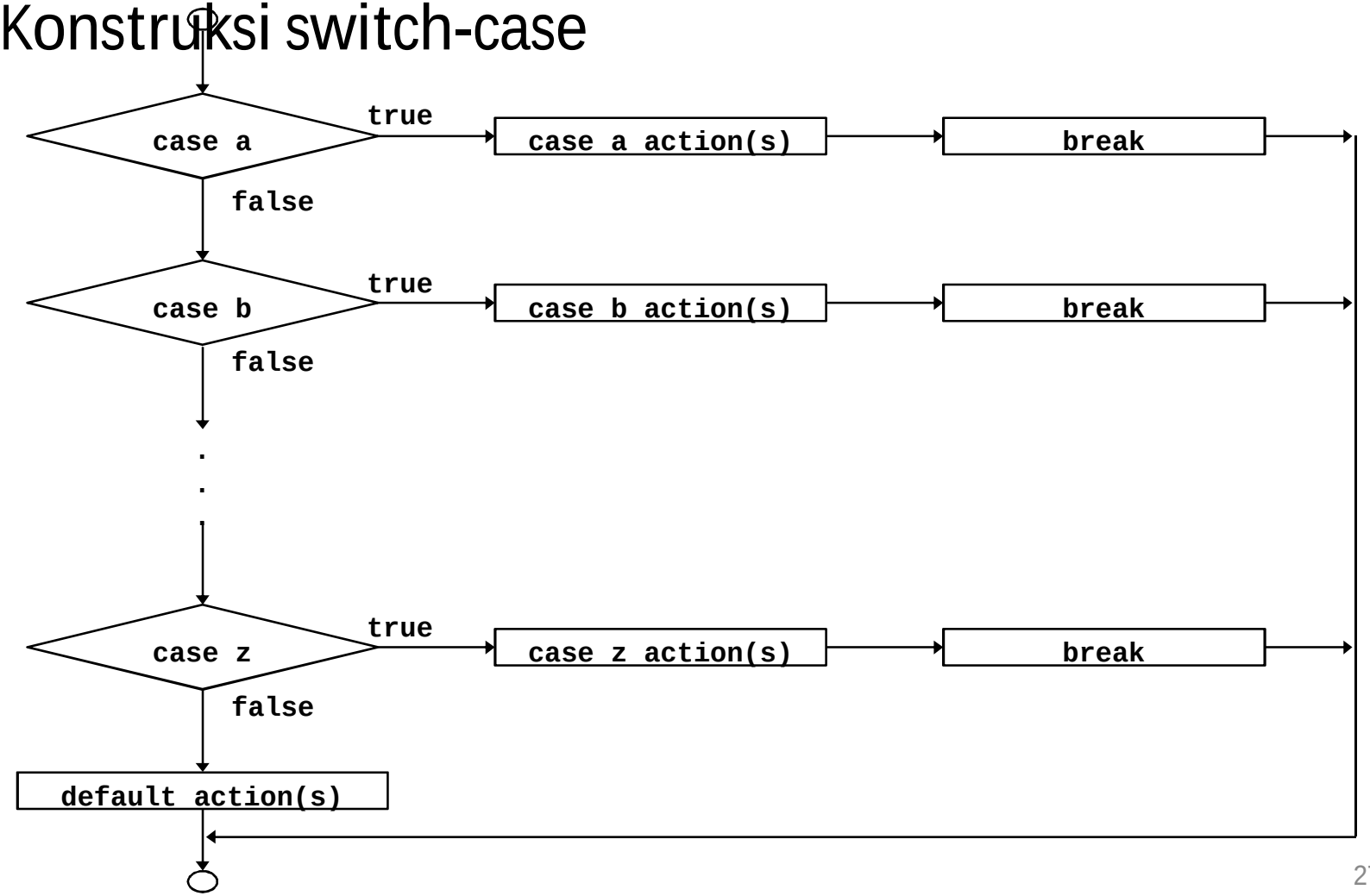
```
switch (ekspresi) {
        case constant1 : statements1; break;
        .
        .
        case constant2 : statements2; break;
        default : statements;
}
```

Operasi Seleksi : switch-case

- Pernyataan switch mengevaluasi ekspresi dan kemudian melihat isi case constant. Jika nilai ekspresi ada didalam constant list maka pernyataan dieksekusi. Jika tidak ada yang cocok, pernyataan default yang dieksekusi.
- Catatan: nilai ekspresi harus integer dan constant harus integer constant termasuk char.

Operasi Seleksi : switch-case

- Konstruksi switch-case



Operasi Seleksi : switch-case

- Contoh :

```
#include <stdio.h>
int main()
 {
       float bil1, bil2; char op;
       while(1) {
             printf("\n Ketik bil1 operator bil2 \n");
             scanf("%f %c %f", &bil1, &op, &bil2);
             switch(op){
                   case('+'): printf(" = %f", bil1 + bil2); break;
                   case('-') : printf(" = %f", bil1 - bil2); break;
                   case('*') : printf(" = %f", bil1 * bil2); break;
                   case('/') : printf(" = %f", bil1 / bil2); break;
                   default :   printf("operator TAK DIKENAL");
             }
       }
       return(0);
}
```

Penulisan: case ('+') bisa ditulis case '+'

Operasi Seleksi

- Contoh Program Menghitung Nilai Ujian
- Tabel Nilai

Nilai Akhir	Bobot	Nilai Huruf
80 – 100	4	A : Sangat Baik
65 – 79	3	B : Baik
55 – 65	2	C : Cukup
50 - 54	1	D : Kurang
0 - 49	0	E : Gagal

- Mata kuliah Dasar Pemrograman dilengkapi dengan praktikum di Lab. Multimedia, maka nilai akhir dihitung dari :

Operasi Seleksi

– Nilai Teori =

50%(Ujian Akhir Semester) +

30%(Ujian Mid Semester) +

20%(TM Teori)

- Nilai Praktikum =

40%(Ujian Akhir Praktikum) +

30%(Ujian Mid Praktikum) +

30%(TM Praktikum)

- Nilai Akhir = 0.8*NilaiTeori + 0.2*NilaiPraktikum

Operasi Seleksi

- Contoh :

```
/*------------------------
Program Nilai_Ujian
 ------------------------*/
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int TugasT, TugasP;
int MidT, MidP, Nilai_akhir;
int FinalT, FinalP;
float NilaiT, NilaiP;
char Jawab;
```

Operasi Seleksi

```
int main()
{
      clrscr();
      printf("Teruskan [Y/T] ? "); scanf("%c",&Jawab);
      while (toupper(Jawab) == 'Y') {
            printf("TM Teori (0 -100) : ");  scanf("%d",&TugasT);
            printf("TM Praktikum (0 -100) : ");  scanf("%d",&TugasP);
            printf("UTS (0 -100) : ");  scanf("%d",&MidT);
            printf("UTP (0 -100) : ");  scanf("%d",&MidP);
            printf("UAS (0 -100) : ");  scanf("%d",&FinalT);
            printf("UAP (0 -100) : ");  scanf("%d",&FinalP);
            NilaiT = 0.2*TugasT+0.3*MidT + 0.5*FinalT;
            NilaiP = 0.3*TugasP + 0.3*MidP + 0.4*FinalP;
            Nilai_akhir = ceil(0.8*NilaiT + 0.2*NilaiP);
            if(Nilai_akhir >=85) printf("Nilai_akhir = A");
              else if(Nilai_akhir >=75) printf("Nilai_akhir = B");
                 else if(Nilai_akhir >=65) printf("Nilai_akhir = C");
                    else if(Nilai_akhir >=50) printf("Nilai_akhir = D");
                       else printf("Nilai_akhir = E");
            printf("\n");
            printf("Teruskan [Y/T] ? ");  scanf("%c",&Jawab);
     }
}
```

Operasi Seleksi

- Contoh program diatas menggunakan fungsi:
  - clrscr()
    - Untuk membersihkan layar
    - Header file: <conio.h>
  - toupper()
    - Sintak: int toupper(int c);
    - Mengkonversi karakter c ke huruf besar
    - Header file: <ctype.h> dan <stdlib.h>

Goto dan Label

- Bahasa C masih menyediakan statement goto ke suatu label.
- Sintak:

```
        goto label;
              ......
    label :
              ......
```

- Penulisan label diakhiri dengan titik dua
- Hindari menggunakan goto karena bisa menyebabkan program menjadi tidak terstruktur.

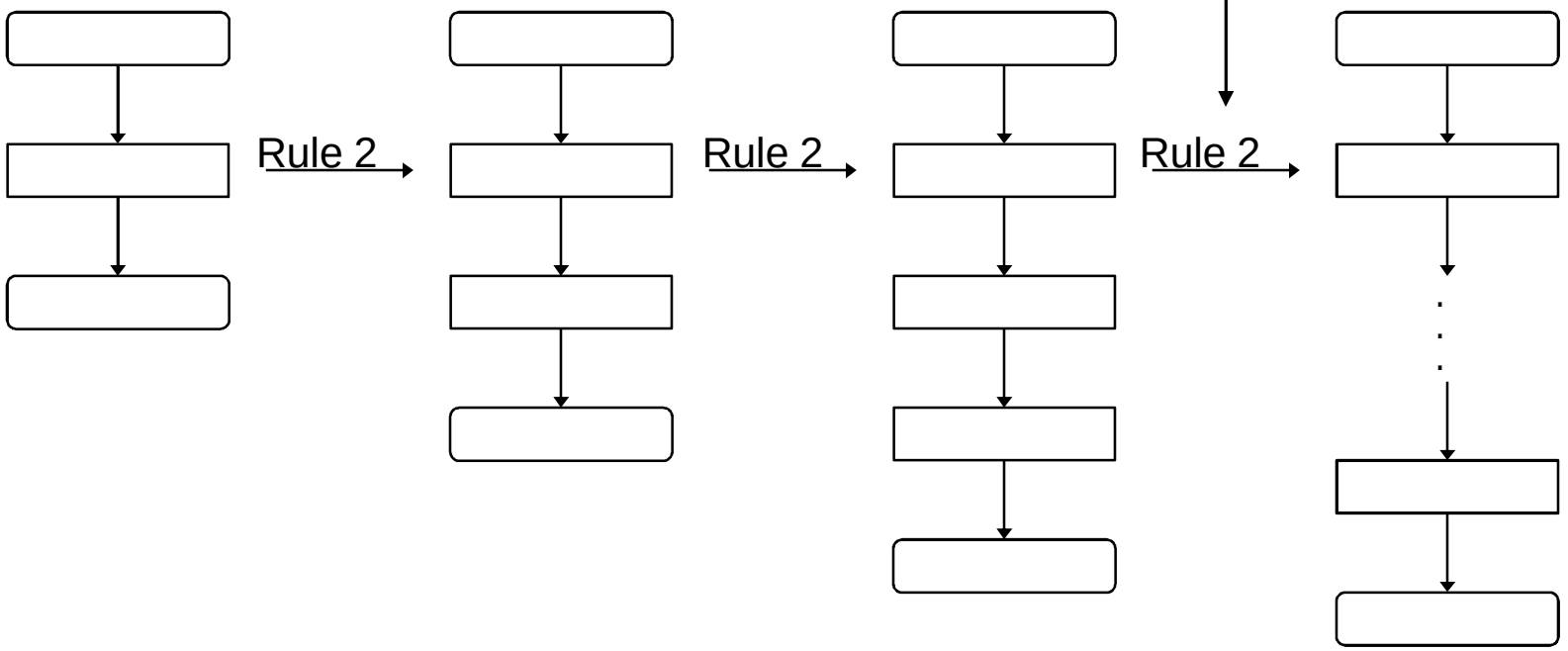
Structured-Programming

- Rules for structured programming
  - Rules developed by programming community
  - Only single-entry/single-exit control structures are used
  - Rules:
    1. Begin with the “simplest flowchart”
    2. Any rectangle (action) can be replaced by two rectangles (actions) in sequence
    3. Any rectangle (action) can be replaced by any control structure (sequence, **`if`**, **`if/else`**, **`switch`**, **`while`**, **`do/while`** or **`for`**)
    4. Rules 2 and 3 can be applied in any order and multiple times

Structured-Programming

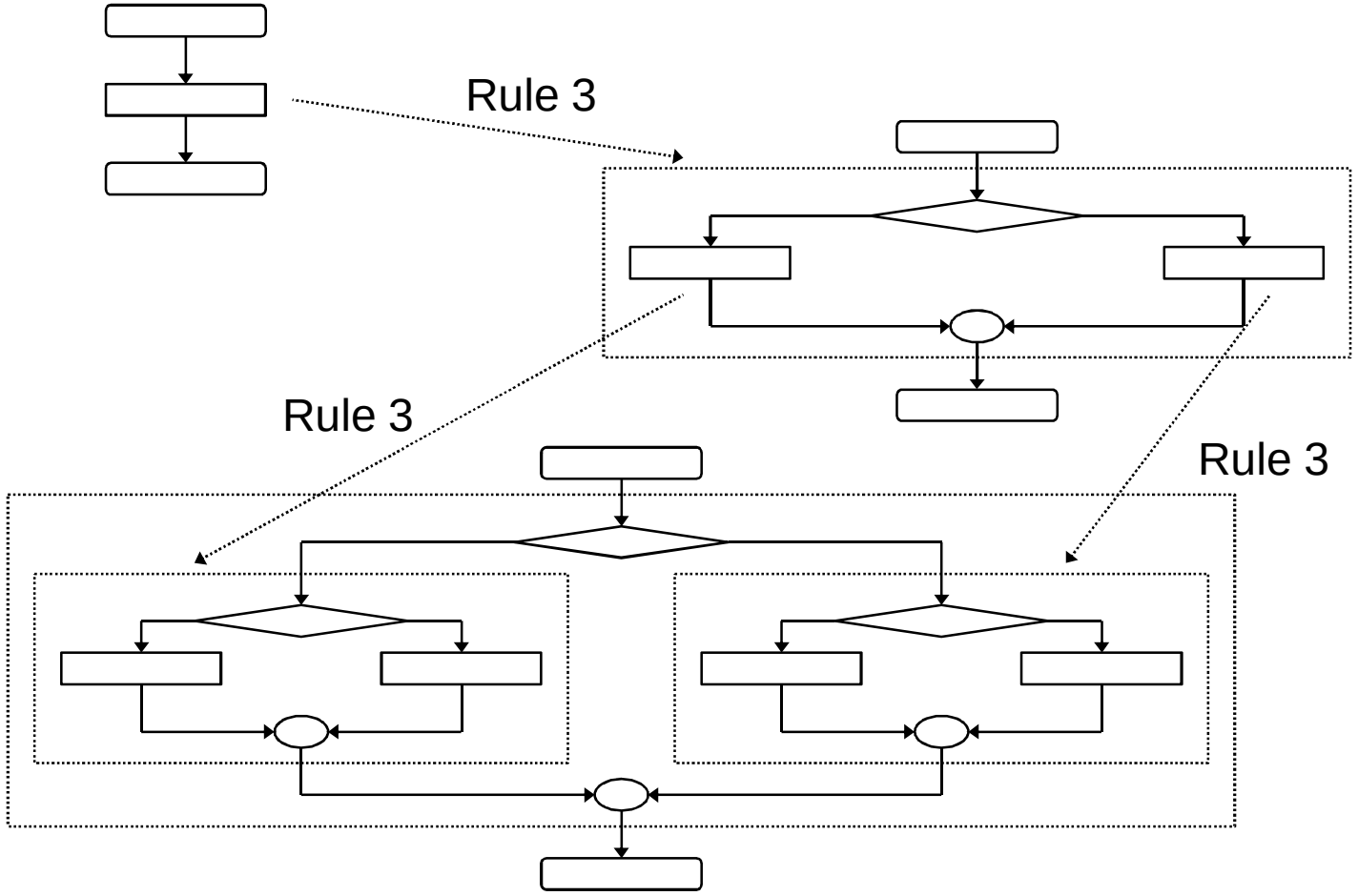
Rule 1 - Begin with the simplest flowchart

Rule 2 - Any rectangle can be replaced by two rectangles in sequence



Structured-Programming

Rule 3 - Replace any rectangle with a control structure



Structured-Programming

- All programs can be broken down into 3 controls
  - Sequence – handled automatically by compiler
  - Selection – **`if`**, **`if`**/**`else`** or **`switch`**
  - Repetition – **`while`**, **`do`**/**`while`** or **`for`**

Jenis Error

- Compile-Time Error
  - Error saat dikompilasi yang disebabkan oleh sintax error (penulisan program yg salah)
- Link-Time Error
  - Saat dikompilasi tidak masalah, tetapi pada saat di link error
  - Biasanya disebabkan karena object code tidak ada saat proses link
- Run-Time Error
  - Saat dikompilasi dan link tidak masalah, tetapi saat di run hasilnya error. Error ini biasanya disebabkan karena operasi bilangan spt, overflow, floating point underflow, division by zero
- Logical Error
  - Error karena kesalahan logika atau algoritma

Jenis Error

- Diantara jenis error diatas yang paling sulit mencari kesalahannya adalah Logical Error.
- Contoh program dengan Compile-Time Error

```
#include <stdio.h>

int main()
{
    printf("Selamat Datang di UBINUS\n);
    return(0);
}
```

Compililer Dev-C akan memberikan pesan sbb: In function main missing terminating ” character, sintax error berfore return

Jenis Error

- Banyak compiler C yang menggabung proses kompilasi dengan link, sehingga agak susah membedakan jenis error antara Compile-Time Error dengan Link-Time Error
- Contoh program dengan Link-Time Error (Visual C++)

```
int main()
{
    extern int x;
    x=2345;
    return(0);
}
```

```
Compiling...
coba.cpp

coba.obj - 0 error(s), 0 warning(s)
```

```
Linking...
coba.obj : error LNK2001: unresolved external symbol "int x" (?x@@3HA)
Debug/coba.exe : fatal error LNK1120: 1 unresolved externals
Error executing link.exe.

coba.exe - 2 error(s), 0 warning(s)
```

Jenis Error

- Contoh program dengan Run-Time Error

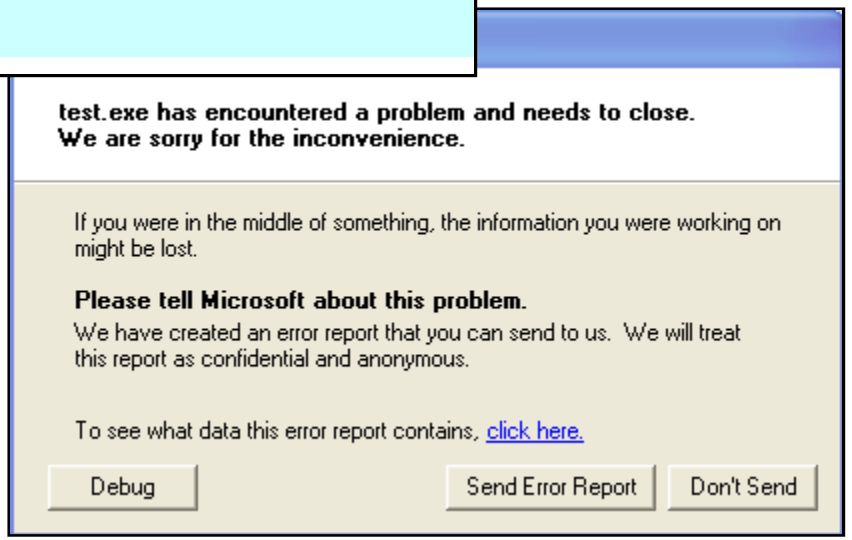
```
#include <stdio.h>
int main()
{
    char n=10;
    n *= n;
    printf("n=%d\n",n);
    n *= n;
    printf("n=%d\n",n);
    return(0);
}
```

- Saat di kompilasi dan link tidak error, tetapi saat RUN hasilnya salah, karena overflow (tipe char range max hanya 127)

Jenis Error

- Contoh program dengan Run-Time Error

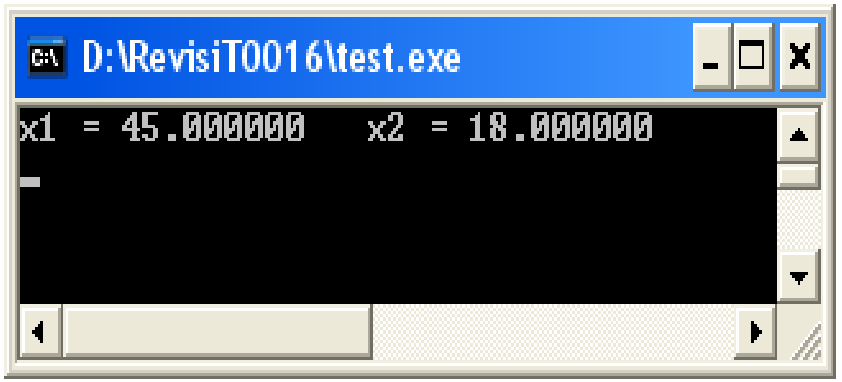
```c
#include <stdio.h>
int main()
{
    int n=10;
    n = n % 2;
    printf("25/n=%d\n",25/n);
    getch();
    return(0);
}
```



Error karena Division by Zero

- Contoh program dengan Logical Error

```c
#include <stdio.h>
#include <math.h>
int main()
{
    double a,b,c,d,x1,x2;
    a=3; b=-21; c=30;
    d = b*b -4*a*c;
    x1 = (-b + sqrt(d))/2*a;
    x2 = (-b - sqrt(d))/2*a;
    printf("x1 = %lf   x2 = %lf\n" ,x1,x2);
    getch();
    return(0);
}
```



Seharusnya nilai x1 = 5.00 dan x2 = 2.00

Dimanakah letak kesalahan ??

Latihan

```
#include <stdio.h>
int main() {
        int n;
        for(; ;) {
            printf("\n Enter integer : ");  scanf("%d ", &n);
            if((n%2) == 0) continue;
            else if((n%3) == 0) break;
            printf("\n\t Bottom of loop.");
        }
       print("\n\t Outside of loop.");
}
```

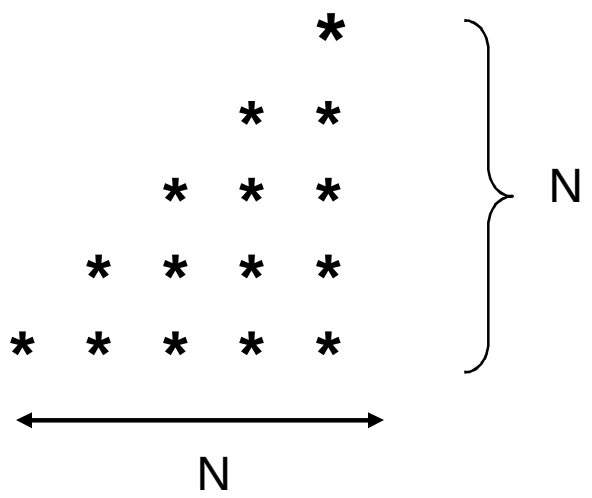
Apa output program jika diinput nilai 7, 4 dan 9 berturut-turut?

Latihan

- Buat program untuk menginput nilai IPK mahasiswa dan berikan penilaian :
  - 3.5 - 4.0 Sangat Memuaskan
  - 3.0 - 3.4 Memuaskan
  - 2.5 – 2.9 Baik Sekali
  - 2.0 – 2.4 Baik
  - Dibawah 2.0 Kurang
- Gunakan perintah if / if-else !

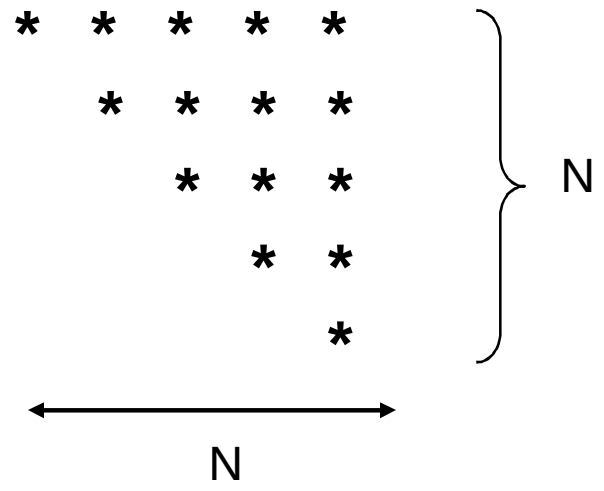
Latihan

- Buatlah program untuk menampilkan gambar seperti contoh dibawah, dimana N variabel integer yg nilainya di-input dari keyboard (dengan for, while atau do-while loop).



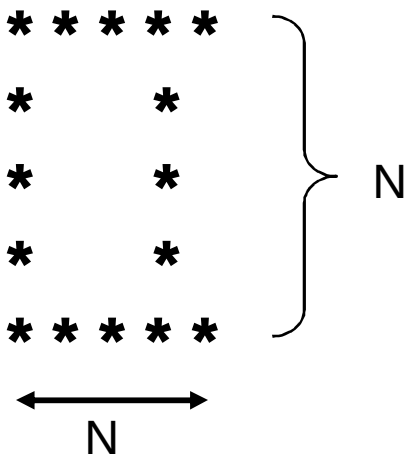
Latihan

- Buatlah program untuk menampilkan gambar seperti contoh dibawah, dimana N variabel integer yg nilainya di-input dari keyboard (dengan for, while atau do-while loop).



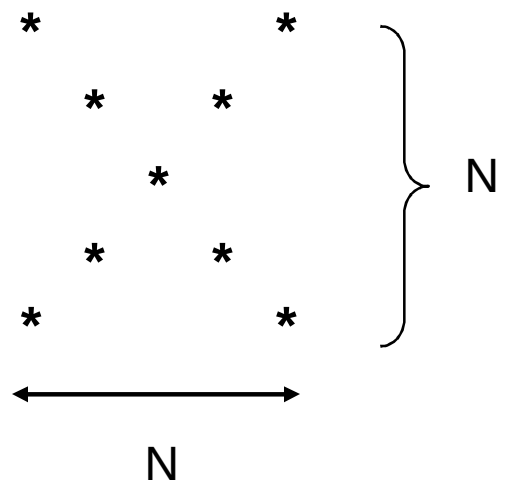
Latihan

- Buatlah program untuk menampilkan gambar seperti contoh dibawah, dimana N variabel integer yg nilainya di-input dari keyboard (dengan for, while atau do-while loop).



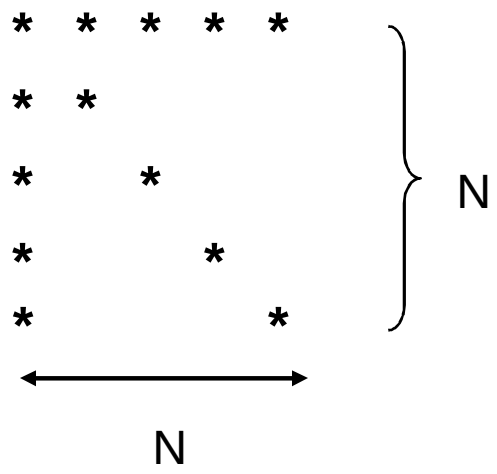
Latihan

- Buatlah program untuk menampilkan gambar seperti contoh dibawah, dimana N variabel integer yg nilainya di-input dari keyboard (dengan for, while atau do-while loop).



Latihan

- Buatlah program untuk menampilkan gambar seperti contoh dibawah, dimana N variabel integer yg nilainya di-input dari keyboard (dengan for, while atau do-while loop).



Latihan

- Perhatikan potongan program dibawah ini:

```
if(n > 0)
      if(a > b)
            z = a;
      else
            z = b;
```

- **Jelaskan keyword else berpasangan dengan if yang mana ?**
- **Perbaiki cara penulisan potongan program diatas, agar menjadi lebih jelas dibaca algoritmanya !**

UTS

Pertemuan 8

Pertemuan 9

Pointer dan Array

Learning Outcomes

Pada akhir pertemuan ini, diharapkan mahasiswa akan mampu :

- Menerapkan konsep tipe data array untuk data majemuk homogen

Outline Materi

- Pointer
- Definisi dan karakteristik array
- Array dimensi satu
- Inisialisasi array
- Array sebagai parameter
- Array berdimensi dua dan tiga
- String

Pointer

- Pointer adalah variabel yang menyimpan alamat dari variabel yang lainnya.
- Deklarasi pointer : <*type*> **ptr_name;*
- *Dua operator yang sering digunakan pada pointer : * (content of) dan & (address of).*
- *Contoh*

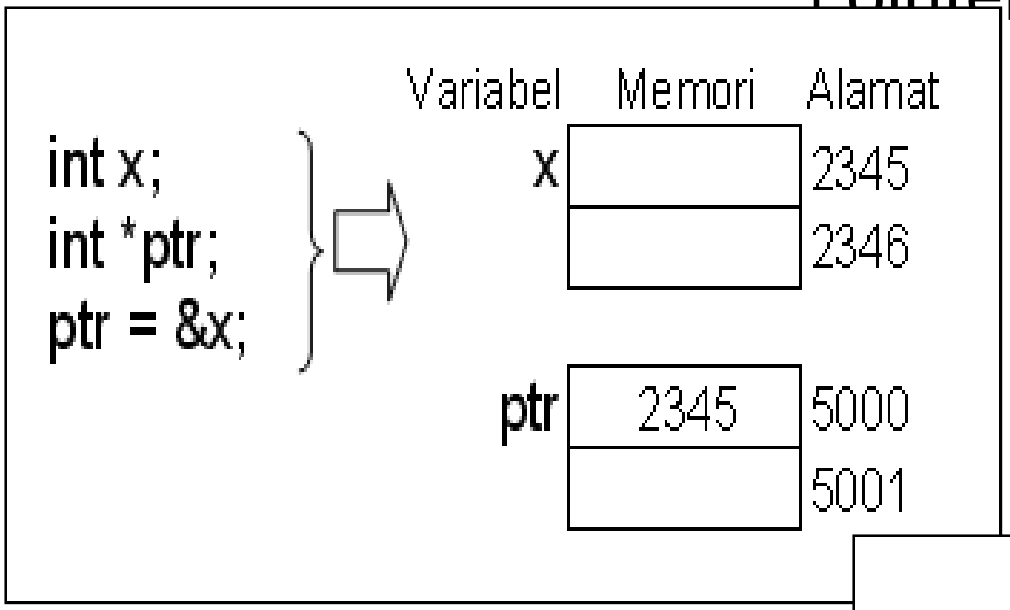
Inisialisasi sebuah integer pointer ke data variable:

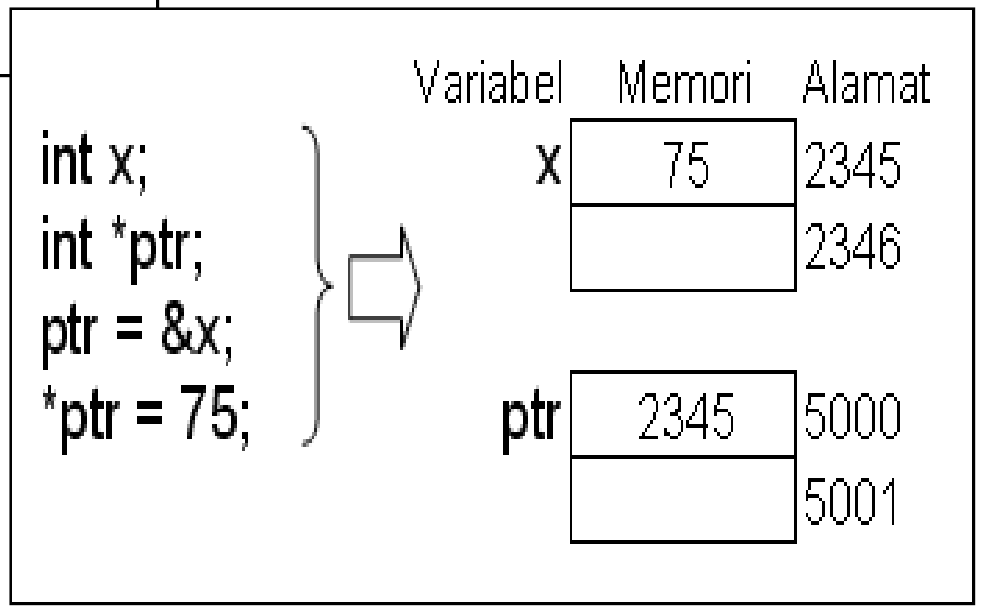
```
int i, *ptr;
ptr = &i;
```

Untuk merubah isi/nilai yg ditunjuk oleh pointer:

```
*ptr = 5;  /* sama artinya dgn i=5 */
```

Pointer


Variabel
Memori
Alamat
int x;
int *ptr;
ptr = &x;
x
2345
2346
ptr
2345
5000
5001


Variabel
Memori
Alamat
int x;
int *ptr;
ptr = &x;
*ptr = 75;
x
75
2345
2346
ptr
2345
5000
5001

Pointer to Pointer

- Pointer to pointer adalah variabel yang menyimpan alamat dari pointer yang lainnya.
- Deklarasi pointer : <*type*> ***ptr_ptr ;*
- *Contoh*

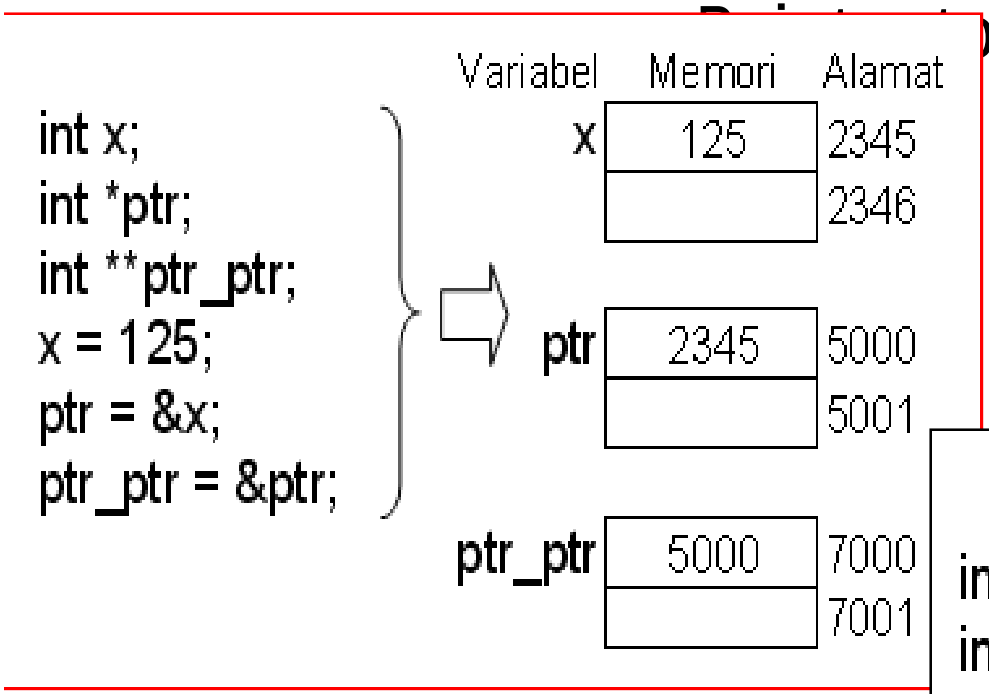
```
int i, *ptr, **ptr_ptr ;
ptr = &i;
ptr_ptr = &ptr;
```

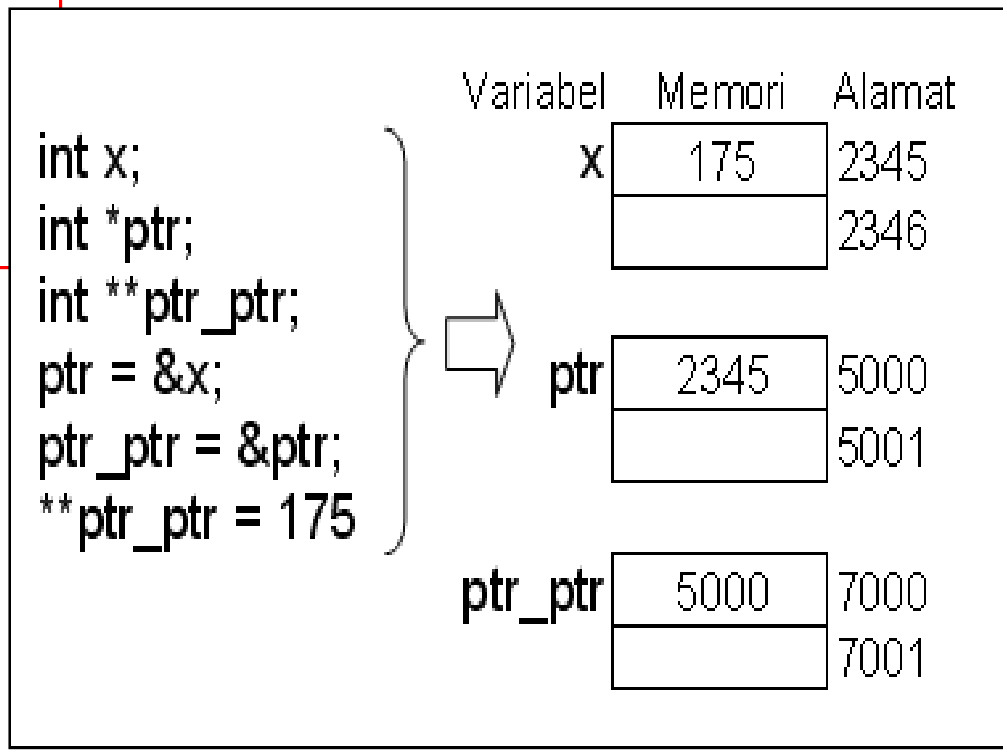
Untuk merubah isi/nilai variabel i bisa melalui sbb:

***ptr = 5; //** sama artinya dgn i=5 ;*

ptr_ptr = 9; //sama artinya dgn i=9; atau *ptr=9;

Pointer to Pointer





Array

- Data disimpan dalam suatu struktur, sedemikian rupa sehingga elemen-elemen di dalam struktur tadi dapat diolah secara kelompok ataupun secara individu.

- **Sifat - sifat Array**
  - Homogen

    Seluruh elemen di dalam struktur array mempunyai tipe data yang sama.
  - Random Access

    Setiap elemen di dalam struktur array dapat dicapai secara individual, langsung ke lokasi elemen yang diinginkan, tidak harus melalui elemen pertama.

Array Dimensi Satu

- Sintaks deklarasi array dimensi satu

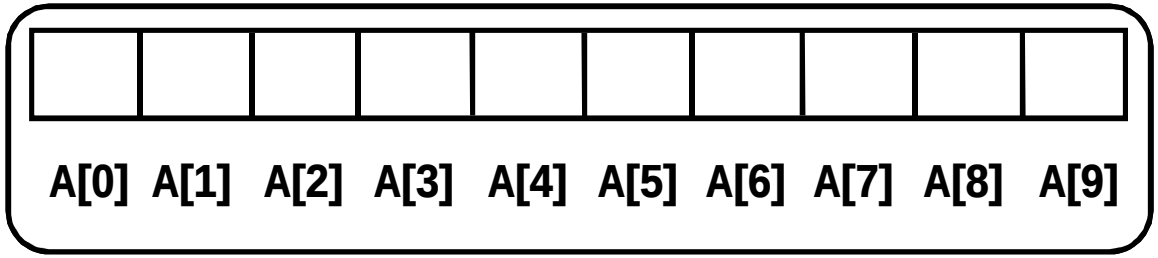
  type nama_arrray[Nilai_dim]**;**

  Contoh : int A[10];

- Definisi dari sebuah array terdiri dari 4 komponen yaitu :
  - Type specifier
  - Identifier (nama array)
  - Operator index ([ ])
  - Nilai dimensi dalam operator [ ]

Visualisasi Array

- Dengan menggunakan contoh deklarasi sebelumnya dapat digambarkan alokasi untuk variabel A
- Elemen-elemen suatu Array diindeks (subscript) mulai dari 0.



Cara Akses Array

- Dua cara yang ekuivalen untuk mengakses unsur ke-i dari suatu array. Misal untuk i=2;

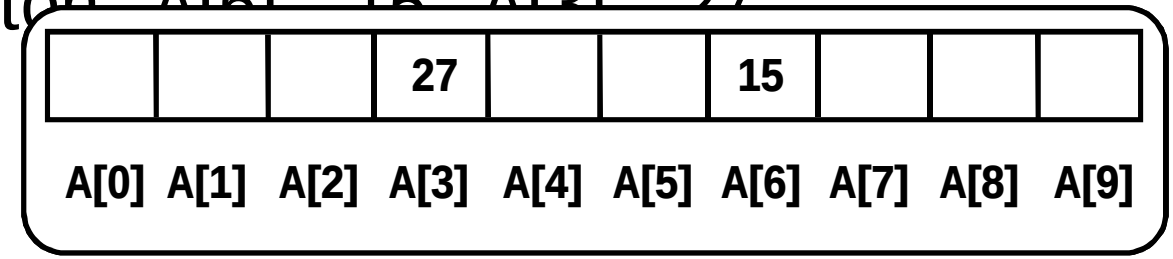
  ***(A+2) atau A[2]**
- A ekuivalen dengan &A[0] atau pointer constant ke elemen-pertama dari array tersebut.
- Bila elemen A[2] hendak ditampilkan di layar monitor, gunakan statemen sbb:

  printf("%d",A[2]) atau

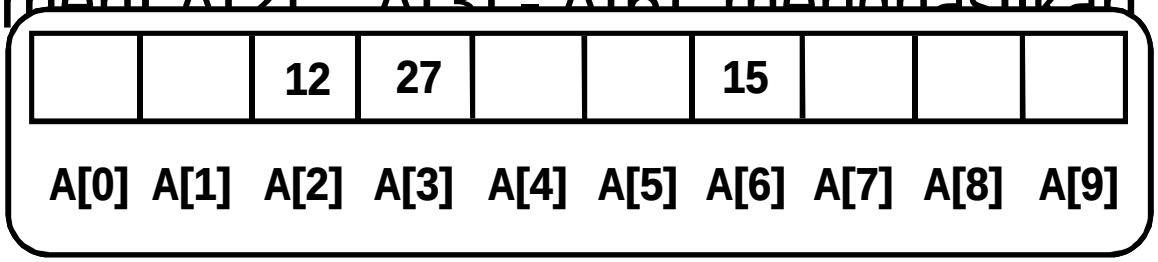
  printf("%d\n",(A+2));*

Mengisi Data Array

- Mengisi data ke elemen array dilakukan dengan menggunakan assignment operator.
- Contoh : A[6] = 15; A[3] = 27;

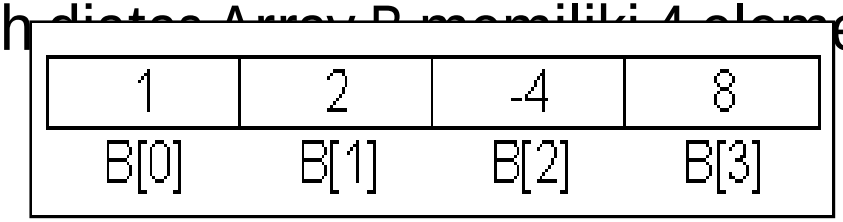


- Statement A[2] = A[3] - A[6] menghasilkan :

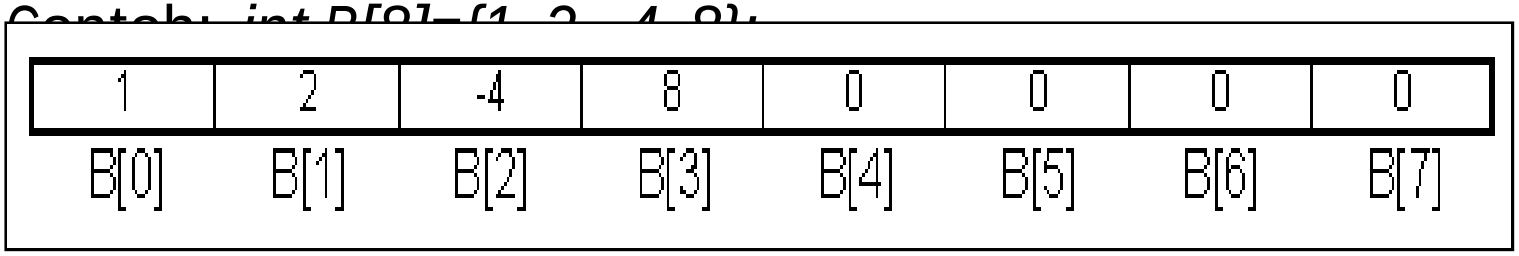


Inisialisasi Array

- Array dapat diinisialisasi secara eksplisit pada saat didefinisikan dan bisa tidak diberikan nilai dimensinya.
  - *Contoh: int B[ ]={1, 2, -4, 8};*
  - Pada contoh diatas Array B memiliki 4 element



  - *Contoh: int B[8]={1, 2, -4, 8};*



Inisialisasi Array

  - Contoh: *int B[4] = { 1, 2, -4, 8, 9 }; //error*
  - Contoh diatas error karena nilai dimensi lebih kecil dari jumlah element.

- Contoh inisialisasi array setelah didefinisikan :

```
int A[5];
(for i=0; i<5;i++) A[i]=0;

int B[5];
B[5]={0,0,0,0,0};
```

Error, mengapa ?

Pointer Constant vs Pointer Variable

- Pointer variabel : adalah pointer yang isinya bisa dirubah-rubah pada saat run time
- Pointer constant : adalah pointer yang isinya tdk bisa dirubah pada saat run time
- Array tipenya adalah Pointer Constant ke element pertama dari array tersebut, oleh karena itu Array bisa mengisi pointer variabel.
- Contoh:
  - int x=10, y=20;
  - int *ptr; //ptr adalah pointer variable
  - ptr = &x;
  - ptr = &y;

} Isi ptr bisa dirubah-rubah saat run time

Pointer Constant vs Pointer Variable

- Contoh:
  - int x=10, y=20;
  - int B[4]; //B adalah Array => pointer constant
  - int *ptr; //ptr adalah pointer variable
  - ptr = &x; //ok
  - ptr = B; //ok
  - ptr++; //ok
  - B = ptr; //error
  - B++; //error
  - B = &y; //error
- ptr = B; sama artinya dengan ptr = &B[0]; karena B tipenya pointer konstan ke elemen pertama dari array

Pointer Constant vs Pointer Variable

- Pointer konstan hanya bisa di-inisialisasi pada saat didefinisikan.
- Contoh:

```
int Arr1[10];
Arr1[10] = {1, 2, 3, 4, 5};   //error
Arr1 = {1, 2, 3, 4, 5};    //error
Arr1[10] = 12;  //error krn max dimensi adl 9
Arr1[0] = 23;  //ok

int Arr2[10] = {1, 2, 3, 4, 5}; //ok
```

Akses Array

- Akses array dengan pointer

```
int arr[10];
int *ptr_arr;
ptr_arr = arr; // sama artinya dengan
          // ptr_arr = &arr[0];
```

- Untuk mengakses elemen ke-i dapat dilakukan dengan cara berikut:

```
ptr_arr[i];
arr[i];
*(ptr_arr + i);
*(arr + i);
ptr_arr = ptr_arr + i; *ptr_arr;
```

Array

```
#include <stdio.h>
void main()
{
        int i;
        int list_int[10];
        for (i=0; i<10; i++){
            list_int[i] = i + 1;
              printf( "list_int[%d] diinisialisasi dengan %d.\n", i, list_int[i]);
       }
}
```

Array Dimensi Satu

- Bahasa C tidak membatasi jumlah dimensi array yang bisa digunakan. Hal ini semata-mata dibatasi jumlah memori komputer yang tersedia.
- Contoh Array dimensi 1:

```
#include<stdio.h>

int SIZE = 5;

void main() {
        int i, j;
        int n[SIZE] = {15, 9, 1, 7, 5};
        for( i=0 ; i<= SIZE ; i++) {
                printf("%5d ", n[i]);
                for ( j=1; j<=n[i] ; j++) printf("%c","*");
                printf("\n");
        }
}
```

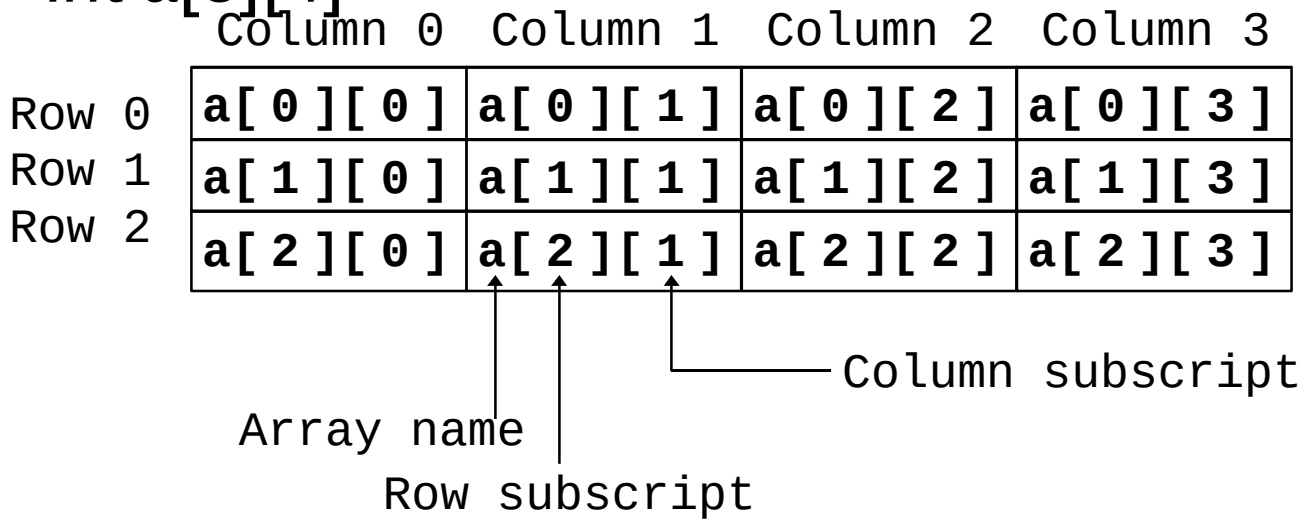
Array Dimensi Dua

- Sintaks deklarasi array dimensi dua

  type *nama_arrray[baris][kolom];*

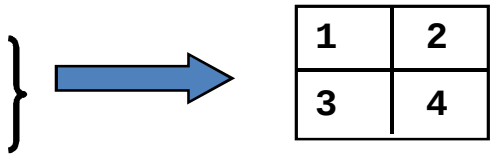
- Contoh

  int a[3][4]



Array Dimensi Dua

- Inisialisasi: menggunakan aturan rmo (row major order).
- Contoh:
  - int b[2][2] = {1, 2, 3, 4 };
  - int b[2][2] = { { 1, 2 }, { 3, 4 } };



1	2
3	4

  - int b[2][2] = { { 1 }, { 3, 4 } };

1	0
3	4

  - int x[3][4] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12};
  - int x[3][4] = { {1, 2, 3, 4},
    {5, 6, 7, 8},
    {9, 10, 11, 12}};

Array Dimensi Dua

- Program :

```
/* Mencetak array 2-D */
#include <stdio.h>
void main() {
        int two_dim[3][5] = {1, 2, 3, 4, 5,
                             10, 20, 30, 40, 50,
                            100, 200, 300, 400, 500};
        int i, j;

        for (i=0; i<3; i++){
            for (j=0; j<5; j++) printf("%6d", two_dim[i][j]);
           printf("\n");
        }
}
```

```
1 2 3 4 5
10 20 30 40 50
100 200 300 400 500
```

Array Dimensi Tiga

- Sintaks deklarasi array dimensi tiga :

  type *nama_arrray[baris][kolom][depth]**;***

- Contoh:

```
int x[3][2][4] = {{{1,2,3,4}, {5,6,7,8}},
                  {{11,12,13,14}, {15,16,17,18}},
                  {{21,22,23,24}, {25,26,27,28}}};

void main() {
  x[4][3][5] = {{{1, 2, 3}, {0, 4, 3, 4}, {1, 2}},
                {{9, 7, 5}, {5, 7, 2}, {9}},
                {{3, 3, 5}, {2, 8, 9, 9}, {1, 2, 1}},
                {{0}, {1}, {0, 1, 9}}};
  printf("%5d", x[2][1][3]);
}
```

Array Of Pointer

- Sebuah array yang isinya adalah pointer
- Sintak :

  type *nama_array [nilai_dim];

- Contoh:

```
int i;
int *ptr[4];
int x=1, y=2,  z=3, w=5;
ptr[0]=&x, ptr[1]=&y; ptr[2]=&z; ptr[3]=&w;
for(i=0;i<4;i++) printf("%d ",*ptr[i]);
```

Output : 1 2 3 5

Array of character

- Array yang isinya character
- Sintak:

  char nama_array[nilai_dim];

- Contoh:

```
char nama[40];
char ss[20]={'I','T','P'};  //20 elemen
char ss[ ]= {'I','T','P'};    //5 elemen
```

String

- String adalah ***Array of character*** yang diakhiri dengan ***null character*** ('\0' atau ASCII nya = 0)

- String constant atau string literal adalah beberapa character yang diapit oleh tanda petik dua.
  - Contoh: **"Selamat datang"**

- Tipe dari string constant adalah pointer constant, sehingga bisa di-assigned ke *array of character* sbb :
  - Contoh :

```
char nama[40] = "Sutan";  //ok
nama = "Sutan";   //error krn nama adalah pointer konstan
Nama[40]= "Sutan";  //error
```

String

- String constant dapat digabung pada saat dikompilasi:

  ”Hello,” ” world”

  Sama artinya dengan :

  ”Hello, world”

- Contoh inisialisasi string:

  char s[ ] = ”iTp”;

  Sama artinya dgn :

  char s[ ] = {’i’,’T’,’p’,’\0’};

- *String* bukan tipe data di Bahasa C.

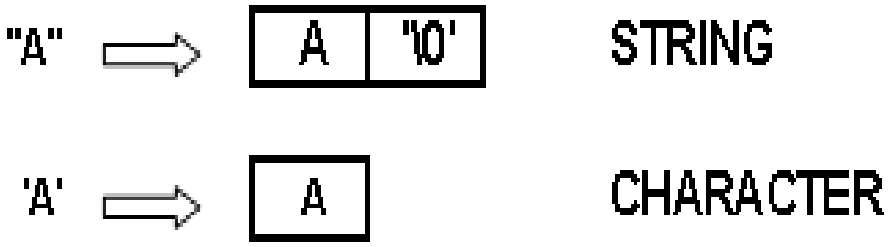
Karakter vs String

- Karakter dalam bahasa C diapit oleh *single quote*. Tiap karakter menempati satu byte memori.
- Contoh:

```
char ch='A';
char ch=65;     //Ascii
char ch=0x41; //Ascii
```

Sama

- String diapit oeh *double quote*.



Manipulasi string

- Pada Standard Library Function (header file **string.h**) disediakan fungsi-fungsi untuk memanipulasi string antara lain :
  - **strlen()**
    Menghasilkan panjang string, tidak menghitung karakter null.
  - **strcpy(s1,s2)**
    Copy s2 ke s1.
  - **strncpy(s1,s2,n)**
    Copy n karakter pertama s2 ke s1.
  - **strcat(s1,s2)**
    Menambahkan string s2 ke akhir dari string s1.
  - **strncat(s1,s2,n)**
    Menambahkan n karakter string s2 ke akhir dari string s1.
  - **strcmp(s1,s2)**
    Membandingkan isi string s1 dan s2, jika isinya sama maka nilainya 0.
  - dll masih banyak lagi

Manipulasi string

- Contoh :

```
char s1[ ] = “abcdef”;
char s2[ ] = “xyz”;

strlen(“nana”);                 // 4
strcmp(“nana”, “nana”)          // bernilai 0
strcpy(s1,s2);                  // s1 = “xyz”, s2 = “xyz”
strncpy(s1,s2,2);               // s1 = “xyabcdef”, s2 = “xyz”
strncpy(s1,s2,4);               // jika n>=strlen(s2) efek sama
                                // dengan strcpy()   s1 = “xyz”
strcat(s1,s2);                  // s1=“abcdefxyz”, s2=“xyz”
strncat(s1,s2,2);               // s1=“abcdefxy”, s2=“xyz”

s1 = “Happy”
s2 = “New Year”

strcat( s1, s2 )                // s1= “Happy New Year”
strncat( s3, s1, 6 )    // s1= “Happy”
strcat( s3, s1 )                // s1= “Happy Happy New Year”
```

Manipulasi string

- Contoh :

```
/* Copy string */
#include <stdio.h>
#include <string.h>
void main() {
        char str1[] = "Copy a string.";
        char str2[15];
        char str3[15];
        int  i;

        strcpy(str2, str1);             // dengan strcpy()

        for (i=0; str1[i]; i++)         // tanpa strcpy()
                str3[i] = str1[i];
        str3[i] = `\0';
                                /* menampilkan str2 dan str3 */
        printf("The content of str2: %s\n", str2);
        printf("The content of str3: %s\n", str3);
}
```

ASCII Character Codes Chart 1

Ctrl	Dec	Hex	Char	Code
^@	0	00		NUL
^A	1	01	☺	SOH
^B	2	02	☻	STX
^C	3	03	♥	ETX
^D	4	04	♦	EOT
^E	5	05	♣	ENQ
^F	6	06	♠	ACK
^G	7	07	•	BEL
^H	8	08	◘	BS
^I	9	09	○	HT
^J	10	0A	◙	LF
^K	11	0B	♂	VT
^L	12	0C	♀	FF
^M	13	0D	♪	CR
^N	14	0E	♫	SO
^O	15	0F	☼	SI
^P	16	10	►	DLE
^Q	17	11	◄	DC1
^R	18	12	↕	DC2
^S	19	13	‼	DC3
^T	20	14	¶	DC4
^U	21	15	§	NAK
^V	22	16	▬	SYN
^W	23	17	↨	ETB
^X	24	18	↑	CAN
^Y	25	19	↓	EM
^Z	26	1A	→	SUB
^[	27	1B	←	ESC
^\	28	1C	∟	FS
^]	29	1D	↔	GS
^^	30	1E	▲	RS
^_	31	1F	▼	US

Dec	Hex	Char
32	20	sp
33	21	!
34	22	"
35	23	#
36	24	$
37	25	%
38	26	&
39	27	'
40	28	(
41	29	)
42	2A	*
43	2B	+
44	2C	,
45	2D	-
46	2E	.
47	2F	/
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	:
59	3B	;
60	3C	<
61	3D	=
62	3E	>
63	3F	?

Dec	Hex	Char
64	40	@
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z
91	5B	[
92	5C	\
93	5D	]
94	5E	^
95	5F	_

Dec	Hex	Char
96	60	`
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z
123	7B	{
124	7C	|
125	7D	}
126	7E	~
127	7F	⌂

Latihan

- Buatlah program untuk mengambil 10 bilangan integer dari keyboard dan disimpan dalam array, kemudian
  - Cari bilangan terbesar dalam array tersebut
  - Cari bilangan terkecil dalam array tersebut
  - Hitung nilai rata-rata dari isi Array tersebut
  - Tampilkan hasilnya di layar monitor

- Buat program untuk:
  - mengambil string dari keyboard
  - membalik susunan karakter dalam string tsb
  - menampilkan string yg sudah dibalik ke layar monitor

  Contoh: string yg di-input : KASUR
  dibalik : RUSAK

Latihan

- Buat program untuk menjumlahkan 2 matrik dibawah ini. Gunakan Array dimensi 2

3	4	5	5
5	6	7	9
1	7	11	10
2	8	9	4

+

1	2	3	4
8	7	6	5
9	10	11	2
6	5	4	3

- Buat program untuk mengalikan 2 matrik dibawah ini. Gunakan Array dimensi 2

3	4	5	5
5	6	7	9
1	7	11	10
2	8	9	4

×

1	2	3	4
8	7	6	5
9	10	11	2
6	5	4	3

Latihan

Perhatikan definisi Array dibawah ini sbb:

```
int A[3][4]={1, 3, 2, 4, 5, 7, 6, 8, 9,11, 12 };
int B[3][3]={{1, 2}, {3, 4, 5} ,{ 7 }};
```

Berapa isi data dari :

1. A[1][1] = ?
2. B[2][2] = ?
3. A[2][3] = ?
4. B[0][1] = ?
5. A[0][2] = ?

Latihan

1. Perhatikan dua statement dibawah:

   ```
   char str[] = ”Selamat Datang di Binus”;
   char *str = ”Selamat Datang di Binus”;
   ```

   Jelaskan perbedaan kedua identifier str tersebut diatas !

2. Perhatikan dua statement dibawah:

   ```
   char *name[] = {”Ali”,”Ani”,”Tono”};
   char name[][10] = {”Ali”,”Ani”,”Tono”};
   ```

   Jelaskan perbedaan kedua identifier name tersebut diatas !

Latihan

- Jelaskan dengan memberikan contoh program sederhana, fungsi / cara kerja dari library function yang ada di <string.h> sbb:
  - strchr(char *s, int c);
  - strrchr(char *s, int c);
  - strstr(char *s, char *src);

Latihan

- Jelaskan beberapa fungsi yang ada di <ctype.h>, sbb:
  - **isalpha(int c);**
  - **isupper(int c);**
  - **islower(int c);**
  - **isdigit(int c);**
  - **isalnum(int c);**
  - **isspace(int c);**
  - **toupper(int c);**
  - **tolower(int c);**

Pertemuan 10

Fungsi

Learning Outcomes

Pada akhir pertemuan ini, diharapkan mahasiswa akan mampu :

- Mendemonstrasikan penggunaan fungsi serta pengiriman parameter

Outline Materi

Fungsi

- Pemrograman Modular
- Library Function vs user-defined function
- Prototipe fungsi
- Jangkauan identifier
- Pengiriman parameter
- Iterasi vs rekursif

Pemrograman Modular

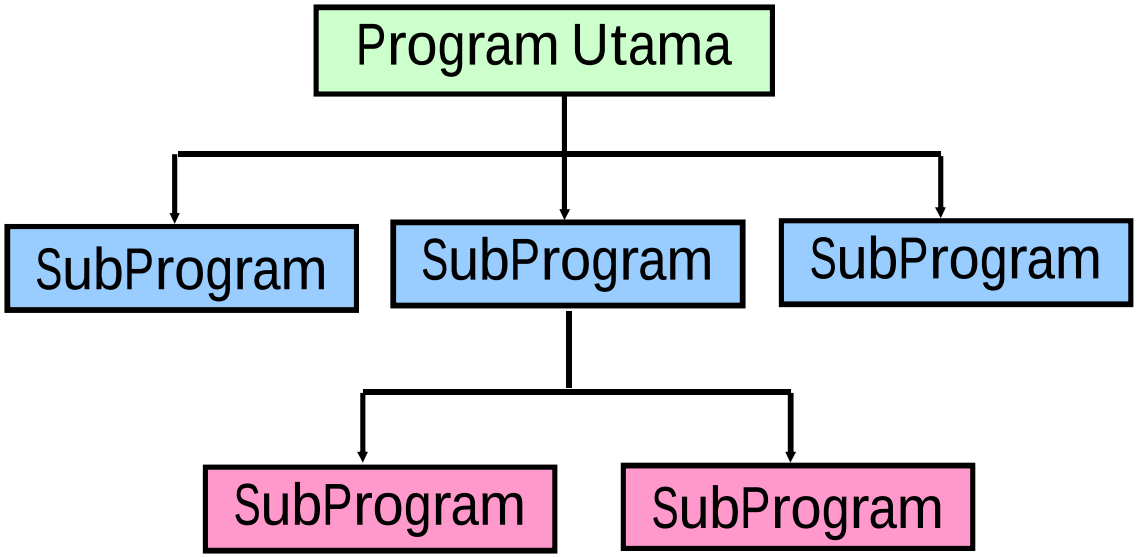
- Program dibagi-bagi menjadi Modul-Modul
- Modul dalam bahasa C di-implementasikan dengan Fungsi
- Fungsi dibentuk dengan mengelompokkan sejumlah perintah untuk menyelesaikan tugas tertentu.
- Modul diperlukan jika kelompok perintah tersebut kerap kali digunakan di tempat lain dalam program
- Modul sering disebut juga dengan Sub-Program

Pemrograman Modular

- **Keuntungan menggunakan modul :**
  1. Rancangan Top - down dengan teknik Sub goal, program besar dapat dibagi menjadi modul-modul yang lebih kecil.
  2. Dapat dikerjakan oleh lebih dari satu orang dengan koordinasi yang relatif mudah.
  3. Mencari kesalahan relatif lebih mudah karena alur logika lebih jelas, dan kesalahan dapat dilokalisir dalam satu modul.
  4. Modifikasi dapat dilakukan, tanpa menggangu program secara keseluruhan
  5. Mempermudah dokumentasi

Pemrograman Modular

- Bahasa C melengkapi fasilitas modular dengan menggunakan ***fungsi*** pada setiap SubProgram.
- Contoh pembagian program menjadi beberapa subprogram.



- **Sifat-sifat modul yang baik adalah :**
  - **Fan-In** yang tinggi, yaitu makin sering suatu modul dipanggil oleh pengguna, makin tinggi nilai fan-in.
  - **Fan-Out** yang rendah, makin sedikit tugas yang dilakukan oleh suatu modul makin rendah nilai fan-out. Dengan demikian, makin spesifik tugas yang dikerjakan oleh modul tersebut.
  - **Self-Contained**, atau memenuhi kebutu-hannya sendiri.

Library vs User-Defined Function

- Fungsi dalama bahasa C terbagi dalam dua jenis :
  - Library function
  - User-defined function

- Library function, adalah fungsi-fungsi standard yang sudah disediakan oleh bahasa C. Fungsi-fungsi tersebut dideklarasikan dalam file header (.h), contohnya clrscr() ada di file conio.h, sqrt() dalam math.h, printf() dalam stdio.h

- User-define function, adalah fungsi yang didefinisikan sendiri oleh pemrogram.

Library vs User-Defined Function

```
#include<stdio.h>
#include<math.h>
int main() {
    int i;
    for(i=0; i<6; i++)  printf("%d %f",i,sqrt(i));
   return 0;
}
```

Contoh Program yang menggunakan Standard Library Function : printf dan sqrt

Konstruksi Fungsi

- Konstruksi fungsi

```
return-value-type  function-name( parameter-list )
{
    statements;
}
```

- return-value-type: tipe data yang dikembalikan oleh fungsi
- Jika tidak diisi maka dianggap tipenya integer (default int)
- Jika return-value-type diganti void maka fungsi tidak mengembalikan nilai
- Parameter-list: berisi daftar nilai yang dikirimkan dari fungsi pemanggil

Konstruksi Fungsi

- Contoh :

formal parameter

```
int maksimum (int x, int y){
        int maks = x;
        if ( y > maks) maks = y;
        return maks
}
```

Fungsi

Pemanggil

```
void main () {
      int a,b;
      printf("Input 2 bilangan bulat : ");
      scanf("%d %d", &a, &b);
      printf("Bilangan yg lebih besar : %d\n",maksimum(a,b));
}
```

Actual parameter

Prototipe Fungsi

- Penulisan fungsi pada bahasa C pada dasarnya diletakkan diatas pemanggil (blok *main,* atau blok fungsi lainnya*)*. Namun adakalanya blok fungsi diletakkan setelah blok pemanggil. Pada kondisi tersebut perlu digunakan prototipe fungsi.
- Tujuan dari prototipe fungsi :
  - Meyakinkan sebuah fungsi dikenal oleh pemanggilnya
  - *Compiler* akan memvalidasi parameter
- Sintaks

  return-value-type function-name(parameter-list);

Prototipe Fungsi

- Contoh :

```
#include<stdio.h>

int maksimum (int x, int y){
     int maks = x;
     if ( y > maks)       maks = y;
     return maks
}

void main () {
    int a,b;
    printf("Input 2 bilangan bulat : ");
    scanf("%d %d", &a, &b);
    printf("Bilangan yg lebih besar : %d\n",maksimum(a,b));
}
```

Karena fungsi maksimum diletakkan di atas pemanggilnya (main program), maka tidak perlu prototipe fungsi

Prototipe Fungsi

- Contoh :

```
#include<stdio.h>

int maksimum(int, int);

void main () {
    int a,b;
    printf("Input 2 bilangan bulat : ");
    scanf("%d %d", &a, &b);
    printf("Bilangan yg lebih besar : %d\n",maksimum(a,b));
}

int maksimum (int x, int y){
     int maks = x;
     if ( y > maks)      maks = y;
     return maks
}
```

Prototipe Fungsi

Karena fungsi maksimum diletakkan di bawah pemanggilnya (main), maka perlu diletakkan prototipe fungsi diatas,supaya dikenal oleh pemanggilnya

Prototipe Fungsi

- Penulisan Prototipe Fungsi seperti diatas bisa ditambah nama parameternya sbb :

  int maksimum(int a, int b);

- Yang dipentingkan dalam prototipe fungsi adalah tipe parameter, jumlah parameter dan urutannya.

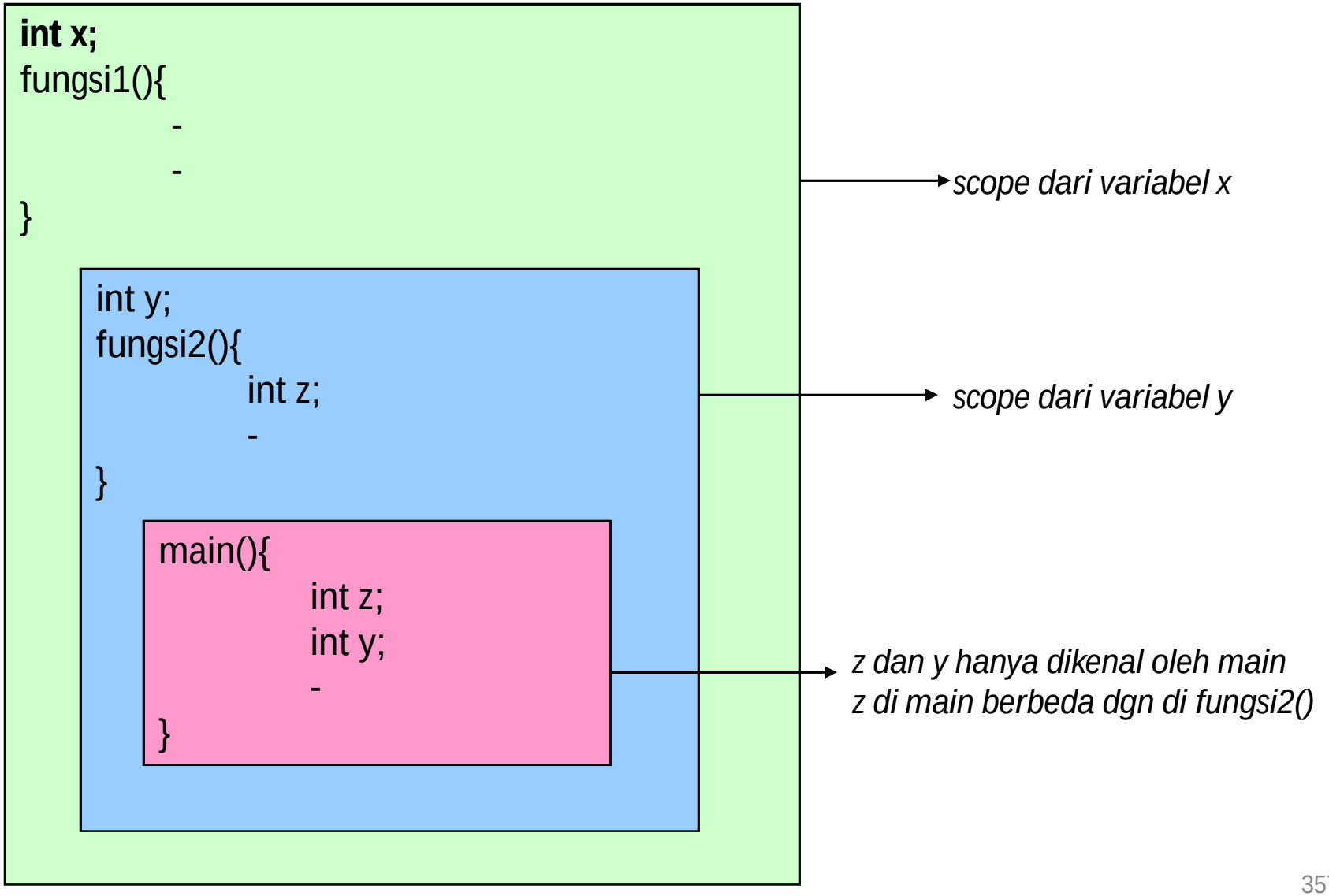
Jangkauan Identifier

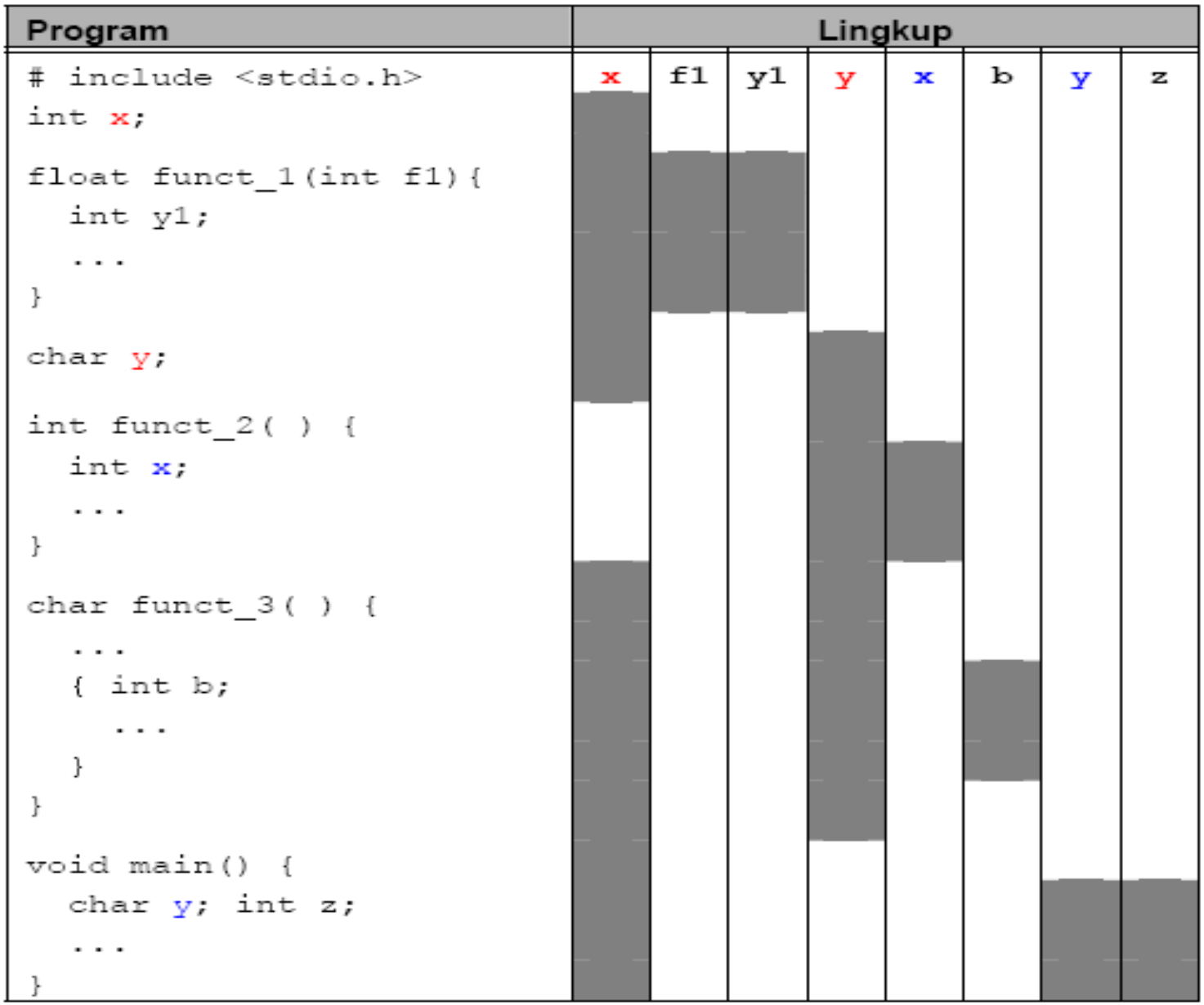
- Jangkauan identifier meliputi bagian-bagian program dimana sebuah identifier masih bisa diakses.
- Jangkauan identifier meliputi :
  - Local
  - Global
- Local identifier
  - Identifier yang dideklarasikan di dalam fungsi, termasuk daftar parameter.
  - Jangkauan terbatas pada function itu sendiri.

Jangkauan Identifier

- Global identifier
  - Identifier yang dideklarasikan di luar fungsi dan ditempatkan di atas semua fungsi dalam suatu program
  - Jangkauan meliputi seluruh program
  - Identifier yang dideklarasikan secara global, dapat dideklarasikan kembali (redeclared) di subprogram
  - Disarankan tidak terlalu banyak menggunakan global identifier karena:
    - Jika program semakin besar, maka semakin besar pula kecenderungan terjadinya error.
    - Sulit melacak kesalahan.
    - Data tidak terjaga dengan baik, setiap subprogram dapat mengubah isi variabel tanpa sepengetahuan subprogram lainnya.

Jangkauan Identifier




Program
Lingkup
x
f1
y1
y
x
b
y
z
include <stdio.h>
int x;
float funct_1(int f1){
int y1;
...
}
char y;
int funct_2( ) {
int x;
...
}
char funct_3( ) {
...
{ int b;
...
}
}
void main() {
char y; int z;
...
}

Pengiriman Parameter

- Bila modul tidak dapat mencukupi kebutuhan sendiri, data ditransfer lewat daftar parameter dan nilai yang dihasilkan juga ditransfer balik melalui daftar parameter.
- Daftar parameter merupakan ‘interface’ suatu modul dengan modul lain.
- Pengiriman Parameter
  - **By-Value**

    Yang dikirim ke modul lain adalah nilainya.
  - **By Location / by reference**

    Yang ditransfer ke modul lain adalah alamatnya.

Pengiriman Parameter

- Contoh : Pengiriman parameter by value

```
#include <stdio.h>

void Garis (char x ) {          /* x disebut Parameter Formal */
{
        int i;                          / *i, x adalah Local Variabel */
        for (i = 1; i<=10; i++) printf("%c",x);
}

/*Program Utama*/
void main()
{
                char A = '-';
                Garis(A);       /* A disebut Parameter Aktual */
}
```

Pengiriman Parameter

- Contoh : Pengiriman parameter by location

```
#include <stdio.h>
void Hitung (int X, int Y, int *P, int *Q)
{
        *P = X + Y;
        *Q = X * Y;
}

void main()
 {
        int X, Y, P, Q;      /*local variabel*/
        printf(" X="); scanf("%d",&X);
        printf(" Y="); scanf("%d",&Y);
        Hitung(X,Y,&P,&Q);
        printf("X + Y = %d\n", P);
        printf("X * Y = %d\n", Q);
}
```

Array Dimensi-1 Sebagai Parameter

- Jika array digunakan sebagai parameter dalam suatu fungsi, maka passing parameter harus by location.
- Contoh:

```
#include <stdio.h>
void cetak_array(int index, int *A) {
        printf("A[%d]=%d\n",index, A[index]);
}

void main() {
        int A[ ]={1,6,2,8,12};
        cetak_array(2, A);
}
```

- Contoh diatas: A pada fungsi main adalah pointer constant, sedangkan A pada fungsi cetak_array adalah pointer variable.

Array Dimensi-2 Sebagai Parameter

Deklarasi fungsinya dapat berupa:

```
void isimatriks(int a[10][10], int b, int k)
```

atau

```
void isimatriks(int a[][10], int b, int k)
```

tetapi **TIDAK** bisa berupa:

```
void isimatriks(int a[10][], int b, int k)
```

atau

```
void isimatriks(int a[][], int b, int k)
```

Array Dimensi-2 Sebagai Parameter

```
#include <stdio.h>
void cetak(int A[3][4])
{
     int row,col;
     for(row=0; row<3; row++){
        for(col=0; col<4; col++)  printf("X[%d][%d]=%d   ",row,col,A[row][col]);
        printf("\n");
     }
}

int main()
{
  int x[3][4]={{1,2,3,4},
               {8,7,6,5},
               {9,10,11,12}};
  cetak(x);
  return(0);
}
```

Pengiriman Parameter

```
int main()
{
   char ss[20]="KASUR";
   balik(ss);
   printf("%s\n",ss);
   getch();
   return(0);
}
```

Untuk string pada formal parameter bisa :

char[ ] atau char *

```
void balik( char ss[ ] )
{
   int c,i,j;
   for(i=0, j=strlen(ss)-1; i<j; i++, j--){
      c=ss[i];
      ss[i]=ss[j];
      ss[j]=c;
   }
}
```

```
void balik( char *ss )
{
   int c,i,j;
   for(i=0, j=strlen(ss)-1; i<j; i++, j--){
      c=ss[i];
      ss[i]=ss[j];
      ss[j]=c;
   }
}
```

Fungsi Rekursif

- Fungsi rekursi adalah fungsi yang didalam function body-nya ada statement yang memanggil dirinya sendiri.
- Fungsi rekursif, sangat berguna dalam pemecahan masalah jika masalah tadi dapat didefinisikan secara rekursif pula.
- Contoh :

  Faktorial (n) atau n! didefinisikan sebagai berikut :

  n! = 1, untuk n = 0;

  *n! = n * (n-1)!, untuk n > 0*

  4! = 4 * 3!

  3! = 3 * 2!

  2! = 2 * 1!

  1! = 1* 0!

  0! = 1

  Bila ditelusur mundur : 4! = 1*2*3*4 = 24

Fungsi Rekursif

- Contoh pe

```
5!
(5 * 4!)
(5 * (4 *3!))
(5 * (4 * (3 * 2!)))
(5 * (4 * (3 * (2 * 1!))))
(5 * (4 * (3 * (2 * (1 * 0!)))))
(5 * (4 * (3 * (2 * (1 * 1)))))
(5 * (4 * (3 * (2 *  1))))
(5 * (4 * (3 * 2)))
(5 * (4 * 6 ))
(5 * 24)
120
```

Fungsi Rekursif

- Fungsi rekursif mempunyai dua komponen yaitu:
  - **Base case**: mengembalikan nilai tanpa melakukan pemanggilan rekursi berikutnya.
  - **Reduction step**: menghubungkan fungsi di suatu nilai input ke fungsi yang dievaluasi di nilai input yang lain. Sekuen nilai input harus konvergen ke base case.
- Fungsi faktorial
  - Base case : n = 0
  - Reduction step: f(n) = n * f(n-1)

Fungsi Iterasi vs Rekursif

- Contoh

- ***Faktorial - Rekursif***

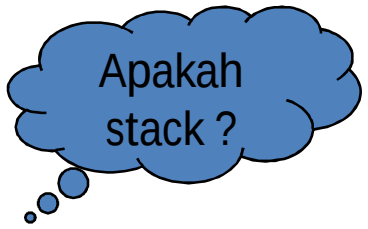
```
long faktor (int n)
{
      if(n==0) return (1);
     else return(n * faktor(n-1));
}
```

- ***Faktorial - Iteratif***

```
long faktor(int n) {
      long i, fak = 1;
      for(i=1; i<=n; i++)  fak *= i;
      return (fak);
}
```

Kekurangan Rekursif

- Meskipun penulisan program dengan cara rekursif bisa lebih pendek, namun procedure atau function rekursif memerlukan :
  - Memori yang lebih banyak, karena perlu tambahan untuk mengaktifkan Stack.
  - Waktu lebih lama, karena perlu menjejaki setiap pemanggilan rekursif melalui stack.



Kapan Menggunakan Rekursif

- Secara umum, gunakan penyelesaian secara rekursif, hanya jika :
  - Penyelesaian sulit dilaksanakan secara iteratif
  - Efisiensi dengan cara rekursif sudah memadai
  - Efisiensi bukan masalah dibandingkan dengan kejelasan logika program
  - Tidak mempertimbangkan faktor penghematan memori dan kecepatan eksekusi program
- Pertimbangan antara aspek kecepatan dan penghematan menggunakan iteratif, dibanding perancangan logika yang baik menggunakan rekursif

Bilangan Fibonacci

- Urutan bilangan 0, 1, 1, 2, 3, 5, 8, 13 ... disebut bilangan fibonacci. Hubungan antara satu angka dengan angka berikutnya didefinisikan secara rekursi sebagai berikut :
  - Fib(N) = N jika N = 0 atau 1
  - Fib(N) = Fib(N-2) + Fib(N-1) jika N >= 2

- Contoh :

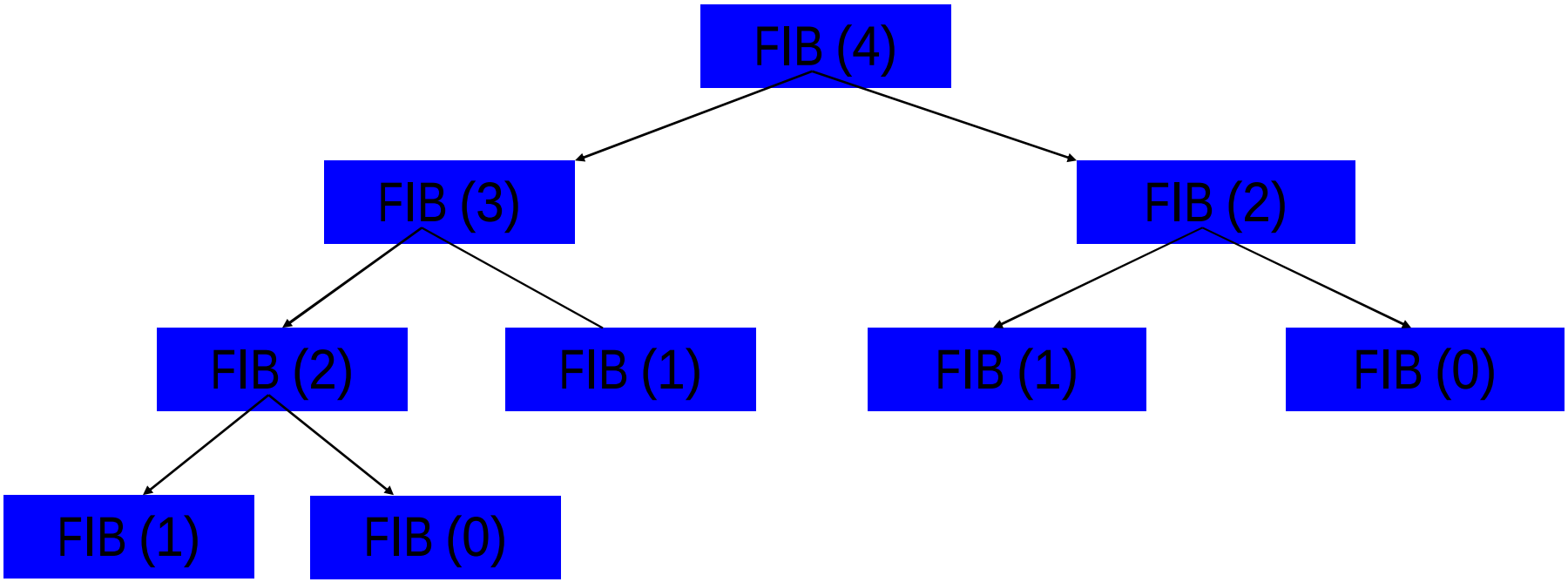
Bilangan Fibonacci

```
int Fib(int n) {
   int f;
   if(n==0) f = 0;
      else if(n==1) f = 1;
         else f = Fib(n-2) + Fib(n-1);
   return f;
}
```

Fungsi fib() di-samping ditulis secara rekursi dan disebut sebagai slow_Fib() tulislah fast_Fib() menggunakan iterasi.

Bilangan Fibonacci

- Contoh : Skema fibonacci jika N=4



Function Parameter Declaration

- Classic Function Parameter declaration
- Contoh:

```
int fungsi1(a)
int a;
{
    a++;
    return a;
}

int fungsi2(b)
int b;
{
    b = b * b;
    return b;
}
```

```
#include <stdio.h>

int main()
{
    int x;
    x=fungsi1(3);
    printf("x=%d\n",x);
    x=fungsi2(13);
    printf("x=%d\n",x);
    return(0);
}
```

Function Parameter Declaration

- Modern Function Parameter declaration
- Contoh:

```
int fungsi1(int a)
{
    a++;
    return a;
}

int fungsi2(int b)
{
    b = b * b;
    return b;
}
```

```
#include <stdio.h>

int main()
{
    int x;
    x=fungsi1(3);
    printf("x=%d\n",x);
    x=fungsi2(13);
    printf("x=%d\n",x);
    return(0);
}
```

Latihan

- Buatlah program dengan fungsi sbb:
  - Fungsi untuk meng-input 10 bilangan ke dalam array
  - Fungsi untuk mencari bilangan terbesar dalam array tersebut
  - Fungsi untuk mencari bilangan terkecil dalam array tersebut
  - Fungsi untuk menampilkan :
    - 10 bilangan tersebut
    - Bilangan terbesar dan terkecil

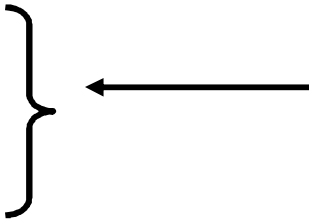
Latihan

- Perbaiki program berikut sehingga bisa digunakan untuk menukar 2 buah karakter

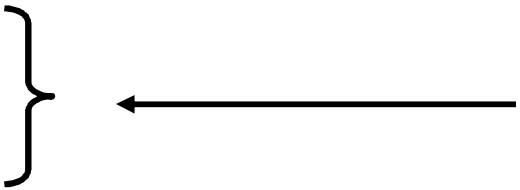
```
void Tukar(char A, char B )
{
      char C ;
      C = A;   A = B,   B = C;
}

void main()
{
            char X, Y ;
            X = 'S';    Y = 'D';
            Tukar(X, Y);
            printf("X = %c  Y= %c", X, Y);
}
```

Latihan

```
#include <stdio.h>
void bagi(float x, int y, float *z) {
    if(y==0) return;
    *z=x/y;
}

float div(float x, int y){
     if(y!=0) return(x/y);
}

void main()
{
   float f,a=12.75;    int b=5;
   bagi(a,b,&f);
   printf("%f dibagi %d = %f\n",a,b,f);
   b=3;
   f=div(a,b);
   printf("%f dibagi %d = %f\n",a,b,f);
}
```



Fungsi tidak mengembalikan nilai



Fungsi yang mengembalikan nilai

Jelaskan apa perbedaan keyword return yang ada pada fungsi bagi dengan return yang ada pada fungsi div ?

Latihan

```
#include <stdio.h>
void bagi(float x, int y, float *z) {
    if(y==0) return;
    *z=x/y;
}

float div(float x, int y){
     if(y!=0) return(x/y);
}

void main()
{
   float f,a=12.75;    int b=5;
   bagi(a,b,&f);
   printf("%f dibagi %d = %f\n",a,b,f);
   b=3;
   f=div(a,b);
   printf("%f dibagi %d = %f\n",a,b,f);
}
```

1. Bolehkah pada fungsi bagi tidak menggunakan keyword return, jika boleh silahkan programnya dirubah ?
2. Bolehkan pada fungsi div tidak menggunakan keyword return ?

Latihan

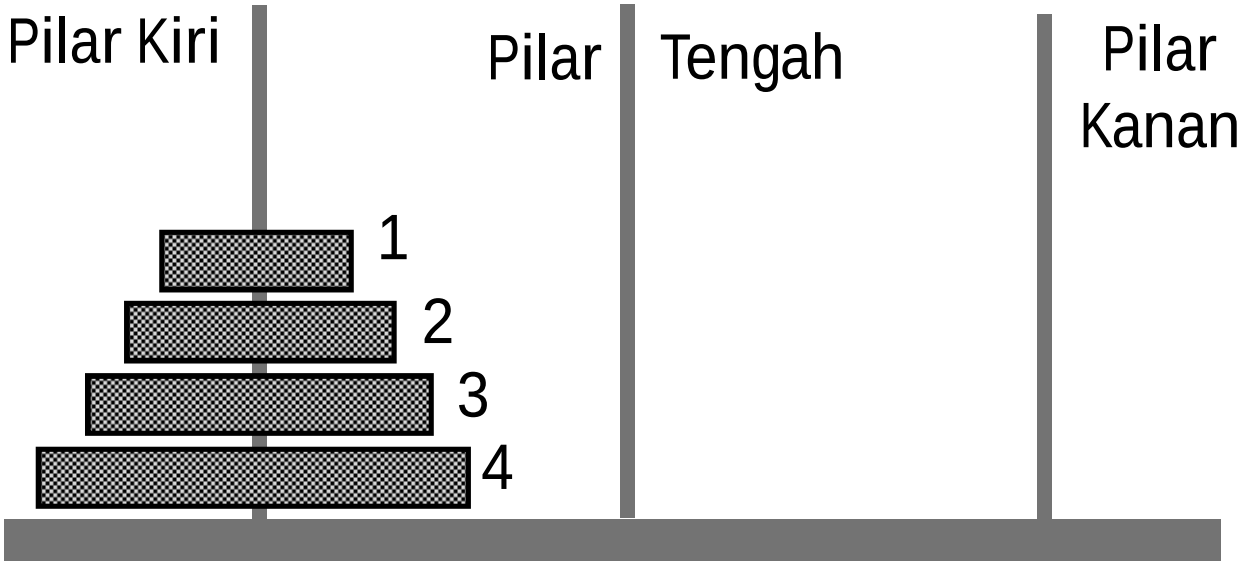
```
#include <stdio.h>
int main(){
    int x,y;
    for(x=1; x<=3; x++) {
        int x=5;
        printf("x=%d ",x++);
        for(y=0; y<x; y++){
          int x=20;
          printf("x=%d ",x++);
        }
        printf("\n");
    }
    return 0;
}
```

Perhatikan Jangkauan variabel x pada program disamping.

Apa output dari program disamping ?

Latihan

- Menara Hanoi



Latihan

- Pindahkan n-piringan dari pilar-kiri ke pilar-kanan dengan pilar-tengah sebagai antara. Piringan yang berada dipilar kiri tersusun sedemikian rupa sehingga menyerupai menara, yaitu piringan yang lebih kecil selalu berada diatas piringan yang lebih besar. Pada proses pemindahan piringan-piringan tersebut, pola susunan menara harus selalu dijaga.
- Alur pemecahan secara rekursif :
  1. Pindahkan (n-1) piringan-piringan atas ke pilar antara.
  2. Pindahkan piringan terakhir ke pilar tujuan.
  3. Ulangi 2 dan 3, hingga selesai.

Latihan

- Simulasikan pemindahan dengan :
  - 3 piringan
  - 4 piringan
  - 5 piringan
- Buat programnya secara rekursif

Pertemuan 11

Structure dan Union

Learning Outcomes

Pada akhir pertemuan ini, diharapkan mahasiswa akan mampu :

- Menerapkan konsep tipe data structure untuk data majemuk non homogen

Outline Materi

Structure

- Definisi dan deklarasi structure
- Nested structure
- Inisialisasi structure
- Akses anggota structure
- Array of structure
- Array vs structure
- Bit field
- Union

Definisi dan Deklarasi Structure

- **Structure**: tipe data yang digunakan untuk menampung sekelompok data yang berbeda tipe, tetapi berkaitan.
- Komponen struktur disebut anggota atau field atau elemen.
- Bersifat heterogen (karena tipe data dari setiap field bisa berbeda).
- Structure di bahasa pemrograman lain, sering disebut dengan **record**

Definisi dan Deklarasi Structure

- Sintaks

```
struct nama_structure {
    tipedata1 nama_field1;
    tipedata2 nama_field2;
    ...
};
```

- variabel structure bisa didefinisikan pada saat deklarasi structure.

```
struct nama_structure {
  tipedata1 nama_field1;
  tipedata2 nama_field2;
...
} nama_variable_structure ;
```

- **Deklarasi variabel structure**

```
struct nama_structure nama_variable_structure;
```

Definisi dan Deklarasi Structure

- Contoh :

```
struct rekening {
    int noRek;
    char tipeRek;
    char nama[31];
    long saldo;
};

struct rekening  nasabah1, nasabah2;
```

ATAU

```
struct rekening {
    int noRek;
    char tipeRek;
    char nama[31];
    long saldo;
} nasabah1, nasabah2;
```

Definisi dan Deklarasi Structure

- Contoh :

```
struct automobile {
    int year;
    char model[8];
    int engine_power;
    float weight;
};
```

```
struct Mahasiswa{
    int Nim;
    char Namal[20];
    float IPK;
    char JenisKelamin;
};
```

Deklarasi Structure

```
# include <stdio.h>
# include <string.h>

struct tmhs {     // tipe data structure
  char nim[11];
  char nama[26];
  float ipk;
};

void function1 ( ) {
  struct tmhs akad_mhs; // variabel structure
  ...
}

int main (){
  ...
  struct tmhs ali, tono;  // variabel structure
  ...
}
```

Deklarasi Structure

```
# include <stdio.h>
# include <string.h>

struct tmhs {     // tipe data structure
  char nim[11];
  char nama[26];
  float ipk;
} akad_mhs, ali, tono; // global variabel structure

void function1 ( ) {
   ...
}

int main (){
  ...
}
```

Deklarasi Structure

- **Structure bisa tanpa nama, dan variabel structure langsung dibuat setelah deklarasi structure.**

```
# include <stdio.h>
```

```
struct {
  char nim[11];
  char nama[26];
  float ipk;
} akad_mhs, tono, ali; // global variabel structure

void function1 ( ) {
    ...
}

int main (){
  ...
}
```

AKSES STRUCTURE

Elemen (field) structure diakses dengan operator titik dari sebuah variabel structure.

```
# include <stdio.h>
# include <string.h>

struct mhs {
  char nim[9];
  char nama[26];
  float ipk;
};
```

```
int main (){
  struct mhs lia;
  float wipk;

  scanf("%s", &lia.nim);
  fflush(stdin);
  gets(lia.nama);
  scanf("%f", &wipk);
  lia.ipk = wipk;
  printf("%s %s %.2f",
  lia.nim, lia.nama, lia.ipk);
  return 1;
}
```

DEKLARASI STRUCTURE LOKAL

```
# include <stdio.h>
# include <math.h>

void main() {
  struct {
    int x, y;
  } tA, tB;
  float dist;

  printf("Titik A : \n   ");
  printf("posisi x dan y ? ");
  scanf("%d %d", &tA.x, &tA.y);
  printf("\nTitik B : \n   ");
  printf("posisi x dan y ? ");
  scanf("%d %d", &tB.x, &tB.y);
  dist = sqrt(pow((tA.x - tB.x), 2) +
         pow((tA.y - tB.y), 2));
  printf("\nJarak A dan B = %.2f unit", dist);
}
```

```
Titik A :
   posisi x dan y ? 5 10

Titik B :
   posisi x dan y ? 15 15

Jarak A dan B = 11.18 unit
```

Nested Structure

- Struktur yang salah satu anggotanya adalah struktur lain.
- Deklarasi struktur lain dilakukan sebelum deklarasi struktur yang memuatnya.
- Contoh :
  - Struct Mhs berisi Nim, Nama, **Alamat**, **Tanggal Lahir**
  - Alamat merupakan struct yang terdiri dari Nama Jalan, Nomor Rumah, Kota, Provinsi
  - Tanggal Lahir merupakan struct yang berisi Tanggal, Bulan dan Tahun

Nested Structure

- Contoh :

```
struct Alamat {
    char Jalan[40];
    int NoRumah;
    char Kota[20];
    char Propinsi[20];
};

struct TanggalLahir{
    int Tangga, Bulan, Tahun;
};

struct Mhs{
    int   Nim;
    char  Nama[20];
    struct Alamat   addr;
    struct TanggalLahir   tgllahir;
};
```

Inisialisasi Structure

- Sintaks
  - Struct *name* variabel = {nilai_1, ..., nilai_m};
- Contoh
  - Struct rekening nasabah1 = {1984, ‘a’, “frenzy”, 200000, 19};

Inisialisasi Structure

- Contoh :

```
#include <stdio.h>

struct employee {
   int id;
   char name[32];
};

void main(void)
{
   struct employee info = {1,"B. Smith"};
   printf("Nama Karyawan: %s\n", info.name);
   printf("ID Karyawan: %04d\n\n", info.id);
}
```

Inisialisasi Structure

- Variabel structure bisa di isi dengan variabel structure yang lain yang nama structure nya sama.

```
#include <stdio.h>
struct employee {
   int id;
   char name[32];
};

void main(void)
{
   struct employee info = {1,"B. Smith"};
   struct employee amir = info;

   printf("Nama Karyawan: %s\n", amir.name);
   printf("ID Karyawan: %04d\n\n", amir.id);
}
```

Array of Structure

- Tipe data struct dalam kenyataannya hanya bisa menampung satu record saja, sedangkan dalam aplikasi biasanya dibutuhkan record lebih dari satu.
- Maka, dalam penggunaanya tipe data struct biasanya digabung dengan array.

Array of Structure

- Contoh :

```
struct tanggal {
      int tgl, bln, thn;
};

struct rekening {
      int noRek;
      char tipeRek;
      char nama[31];
      long saldo;
      struct tanggal transAkhir;
};

//Array of structure
struct rekening nasabah[100];
```

Nilai Awal Array of Structure

```
struct tanggal {
   char nama[31];
   int tgl, bln, thn;
};

struct tanggal ultah[ ] = {
  {“Tata”, 9, 7, 1984},
  {“Titi”, 7, 9, 1986},
  {“Tutu”, 9, 9, 1990}
 };
```

Array of Structure

```
# include <stdio.h>
# include <string.h>

struct tmhs {
  char nim[9];
  char nama[26];
  float ipk;
};
```

```
int main (){
  struct tmhs arr[50], x;
  ...

  scanf("%s", &arr[0].nim);
  ...
  x = arr[0];
  arr[0] = arr[1];
  arr[1] = x;

  for (i = 0; i < 50; i++) {
  printf("%s %s %.2f",
    arr[i].nim, arr[i].nama,
    arr[i].ipk);
  }
  return 1;
}
```

typedef

- typedef : adalah alias (nama lain)
- Digunakan untuk mempersingkat penulisan, khususnya untuk nama/identifer yang panjang.
- typedef sering digunakan pada structure
- Contoh:

```
typedef struct MahasiswaBinus{
    char Nama[20];
    int  Nim;
    float Ipk;
}Mhs;
```

- Pada contoh diatas Mhs adalah nama lain (alias) dari struct MahasiswaBinus, dan berfungsi sebagai tipedata baru.
- Untuk mendefinisikan variabel struct dengan cara sbb

```
Mhs ali, tono;
```

typedef

```
#include <stdio.h>
#include <conio.h>

typedef struct employee {
   int id;
   char name[32];
}EMP;

int main(void)
{
   EMP info = {1,"B. Smith"};
   printf("Nama Karyawan: %s\n", info.name);
   printf("ID Karyawan: %04d\n\n", info.id);

   getch();
   return 0;
}
```

Bit-Field

- Sebuah struct dimana setiap field jumlah bit-nya tertentu.

- Sintak:

```
struct name{
    tipe  field1: jumlah_bit;
    ......
};
```

- Tipe : hanya boleh **unsigned int**, **signed int**, atau **int**.

Bit-Field

- Contoh :

```
struct {
    unsigned short icon : 8;
    unsigned short color : 4;
    unsigned short underline : 1;
    unsigned short blink : 1;
}screen[25][80];
```

- Pada contoh diatas, array screen berisi 2000 element, dan setiap element terdiri dari 2-byte. (14-bit memerlukan 2 byte).

UNION

- **Union** digunakan untuk kongsi memory. Dengan menggunakan union suatu lokasi memori dapat ditempati oleh dua atau beberapa variabel dengan masing-masing tipe data yang berbeda.

- Jumlah memori yang digunakan oleh variabel union adalah sama dengan memori terbesar diantara elemen union.

UNION

Deklarasi tipe data union

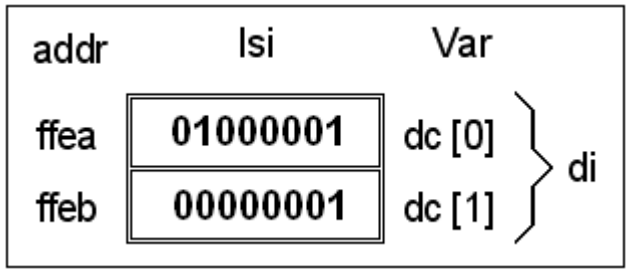
```
union  nama_union{
    tipedata1  nama_var1 ;
    tipedata2   nama_var2;
    ……
} nama_var_union;
```

Deklarasi variabel union

```
union  nama_union   nama_var_union;
```

UNION

```
# include <stdio.h>

void main() {
  union tbil{
    unsigned int di;
    unsigned char dc[2];
  } bil_x;

  bil_x.di = 321;
  printf("di    = %d \n", bil_x.di);
  printf("dc[0] = %d \n", bil_x.dc[0]);
  printf("dc[1] = %d\n",  bil_x.dc[1]);
}
//asumsi size untuk tipe int = 2 byte
```




```
di    = 321
dc[0] = 65
dc[1] = 1
```

321= 101000001

UNION

```
#include <stdio.h>

struct biner{
      unsigned bit0:1; unsigned bit1:1;
      unsigned bit2:1; unsigned bit3:1;
      unsigned bit4:1; unsigned bit5:1;
      unsigned bit6:1; unsigned bit7:1;
};

int main()
{
   unsigned char ch;
   union byte x;
   printf("masukkan bilangan (0-255): ");
   scanf("%d",&ch);
   x.ch=ch;
   printf("%d binernya = %d%d%d%d%d%d%d%d\n",
   ch,x.bit.bit7,x.bit.bit6,x.bit.bit5,x.bit.bit4,
   x.bit.bit3,x.bit.bit2,x.bit.bit1,x.bit.bit0);
   getch();
   return 0;
}
```

```
union byte{
    unsigned char ch;
    struct biner bit;
};
```

Contoh program untuk mengkonversi 1 byte bilangan desimal (0-255) ke bilangan biner dengan menggunakan UNION dan BIT-FIELD

ENUMERATION

- **Enumaration** adalah suatu tipe data yang jumlah datanya sudah ditentukan. Jumlah data yang terbatas ini diberi nama dengan tujuan untuk memperjelas program.

- **Deklarasi tipe data enumeration**

```
enum nama_tipe {
      konstanta1, konstanta2,… konstanta_n
    }nama_var;
```

- **Deklarasi variabel enumeration**

```
enum nama_tipe nama_var;
```

ENUMERATION

```
# include <stdio.h>
enum boolean {false, true};

enum boolean ujigenap(int n) {
  enum boolean hasiluji;
  if (n % 2  == 0) hasiluji = true;
  else hasiluji = false;
  Return hasiluji;
}

int main () {
  int bil;
  enum boolean hasil;

  scanf("%d", &bil);
  hasil = ujigenap(bil);
  if (hasil == true) printf("genap");
  else printf("ganjil");
  return 1;
}
```

```
100
genap
```

```
37
ganjil
```

ENUMERATION

```
# include <stdio.h>
# include <conio.h>

enum tipehari {minggu, senin, selasa, rabu,
   kamis, jumat, sabtu};

void main() {
  enum tipehari hari;

  clrscr();
  for (hari = minggu; hari <= sabtu; hari++)
    printf("%d ",  hari);
  ...
}
```

ENUMERATION

```
#include <stdio.h>
typedef enum boolen {TRUE, FALSE}  BOOL;

int main()
{
   BOOL b;
   b=TRUE;
   if(b==TRUE) printf("b=TRUE\n");
   getch();
   return 0;
}
```

Untuk mempersingkat penulisan enumeration bisa digunakan typedef seperti contoh diatas.

Latihan

- Buatlah struct sebagai berikut :
  - Struct Mhs berisi Nim, Nama, **Alamat**, Tempat, **TanggalLahir**
  - Alamat merupakan struct yang terdiri dari NamaJalan, NomorRumah, Kota, Provinsi
  - TanggalLahir merupakan struct yang berisi Tanggal, Bulan dan Tahun

Latihan

- Berdasarkan soal sebelumnya, buatlah program untuk menginput data sebanyak 5 mahasiswa (menggunakan array of structure)

Latihan

- Berdasarkan struct berikut :

```
struct automobile {
    int year;
    char model[8];
    int engine_power;
    float weight;
    };
```

- Buatlah aplikasi menggunakan array of structure untuk menginput 5 jenis mobil, kemudian tampilkan dalam format yang layak.

Latihan

- Menggunakan struct berikut :

```
struct ipkmhs {
    char nim[11];
    char nama[30];
    float ipk;
};
```

- Buatlah program untuk menginput data 5 mahasiswa, dan tampilkan data mahasiswa ipk >= 3.0 dan ipk < 3.0
- Contoh :

```
Mhs ipk >=3.0 :
    Andi
    Budi
    Candra
Mhs ipk < 3.0 :
    Dadu
    Emin
```

Latihan

- Buatlah sebuah struct :

```
struct nilaimhs {
        char Nim[11];
        char Nama [30];
        char KodeMtk [5];
        int sks;
        char grade;
};
```

- Buat program (tanpa menggunakan array) untuk menginput struct tersebut, kemudian tampilkan nim, nama, kodemtk, sks, grade.

Latihan

- Berdasarkan soal sebelumnya, dengan mempertimbangkan bobot grade dan sks :

Grade	BobotGrade
A	4
B	3
C	2
D	1
E	0

- Buatlah program menggunakan array of struct untuk menginput 5 nilai matakuliah pada semester 1, kemudian tampilkan IP mahasiswa.

Latihan

- Buat program untuk mengkonversi 4 byte bilangan unsigned integer ke bilangan heksadesimal dengan menggunakan UNION dan BIT-FIELD

PERTEMUAN 12

ALOKASI MEMORI DINAMIK

Learning Outcomes

Pada akhir pertemuan ini, diharapkan mahasiswa akan mampu :

- Mendemonstrasikan program aplikasi yang menggunakan alokasi memori dinamik

Outline Materi

- Variabel static
- Register Variabel
- Extern variabel
- Tipe data void *
- Argument pada command line
- Alokasi memori dinamik
- C Preprocessor Directive

Static keyword

- Keyword static bisa digunakan untuk tipe variabel, atau tipe return value dari suatu fungsi
- Variabel static:
  - Dialokasikan saat program start dan di dealokasikan jika program sudah berakhir
  - Default nilai awal=0
  - Scope variable static adalah file dimana dia didefinisikan
  - Sintak:

    static tipe nama_variabel;
  - Contoh:

    static int x;

Static keyword

```
// Example of the static keyword

static int i; //Variable accessible only from this file
static void func(); // Function accessible only from this
                    // file
int max_so_far( int curr ){
    static int biggest; // Variable whose value is
                        // retained between each function
                        // call
    if( curr > biggest ) biggest = curr;
    return biggest;
}
```

Variabel Static

```
#include <stdio.h>

void cetak()
{
    static int count=0;
    printf("count=%d\n",count++);
}

int main(void)
{
   int i;
   for(i=0; i<5; i++) cetak();
   for(i=0; i<3; i++) cetak();
   return 0;
}
```

Output :

```
count = 0
count = 1
count = 2
count = 3
count = 4
count = 5
count = 7
```

Register variabel

- Tujuan nya : menambah kecepatan
- Untuk mendefinisikan agar sebuah variabel disimpan di sebuah register (jika memungkinkan / belum tentu diterima karena jumlah register terbatas). Jika tidak diterima maka akan diperlakukan sebagai variabel otomatis
- Sintak:

  register *tipe_data nama_variabel;*
- *Contoh :*

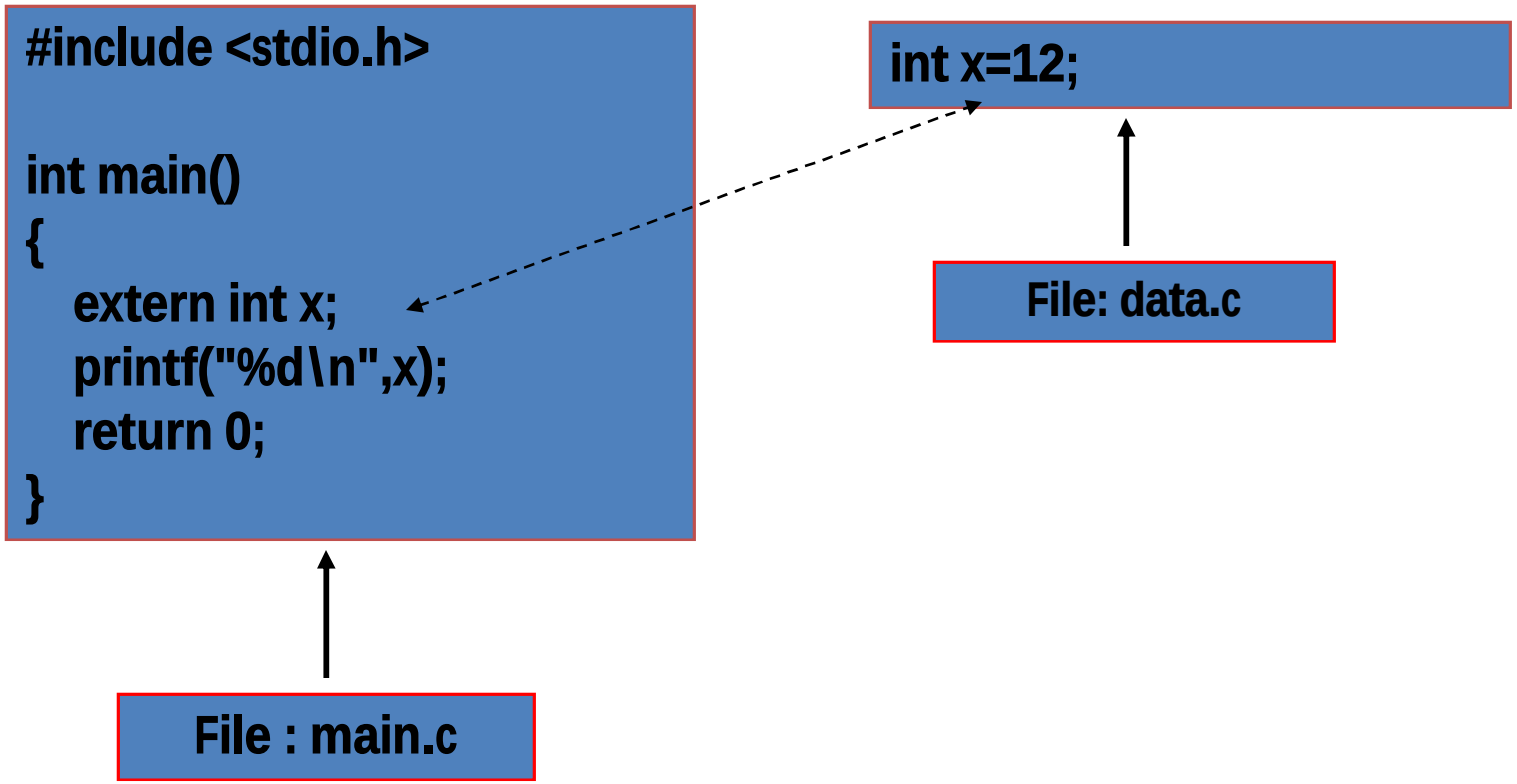
  register int x;

external variabel

- Pada program modular, program dibagi bagi menjadi modul-modul, dan modul dalam bahasa C di implementasikan dengan fungsi
- Satu modul bisa terdiri dari beberapa fungsi, dan disimpan dalam satu nama file
- Jika suatu fungsi di file tertentu ingin mengakses variabel di file yang lainnya, maka digunakan keyword **extern**
- Contoh **: extern int x;**

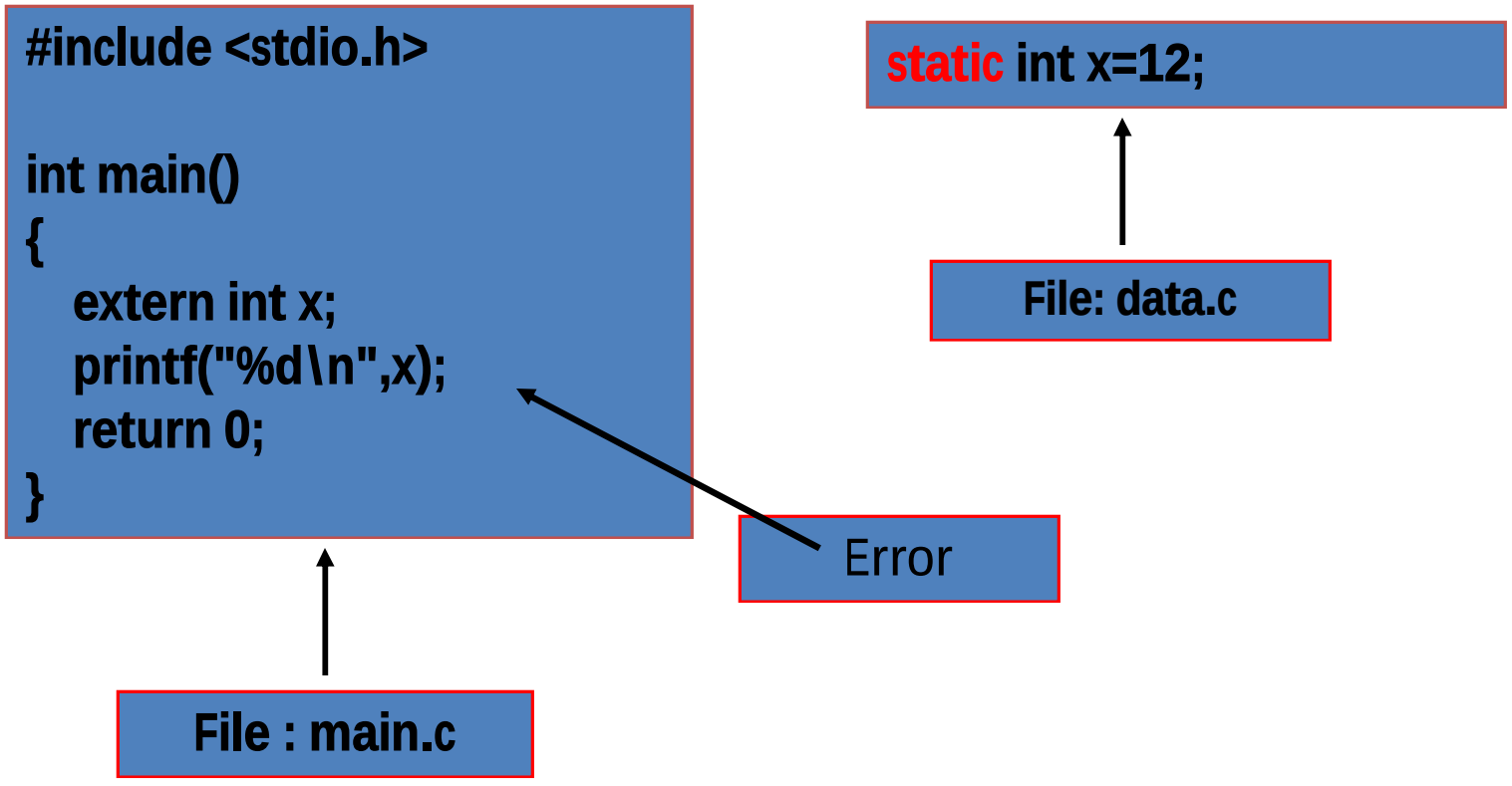
external variabel

- Contoh



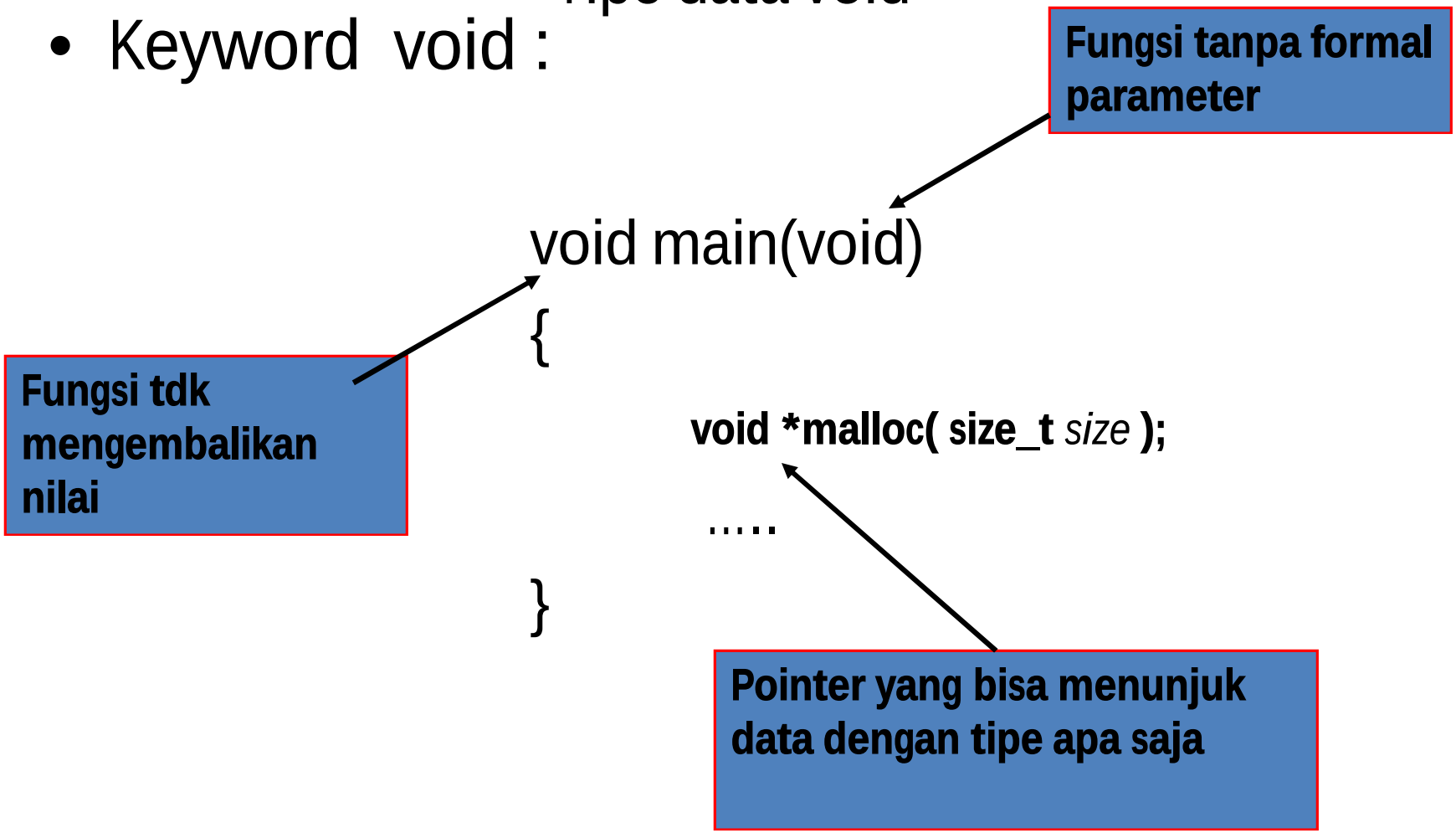
extern variabel

- Contoh



Tipe data void *

- Keyword void :



Tipe data void *

Tipe void * selalu menggunakan Cast

Contoh:

```
char *pstr;
int *pint;
float *pfloat;
pstr=(char *)malloc(20);
pint=(int *)malloc(20);
pfloat=(float *)malloc(20);
```

const char *

- Apakah beda antara char * dengan const char * pada sintak strcpy dibawah ini :

```
char *strcpy( char *strDestination, const char *strSource );
```

- **Jawab:**
  - **char * : harus sebuah pointer (memiliki address)**
  - **const char * : bisa string constant atau pointer**

- **Contoh:**

```
char ss[20];    char str[20] = "Selamat Pagi";
strcpy(ss,"Selamat Pagi");
strcpy(ss, str);
strcpy("Hello", ss);  //error
```

Command-line Arguments

- Beberapa aplikasi seringkali harus mengambil parameter dari command lain, seperti commad pada DOS, Linux untuk a.l:
  - copy file1 file2
  - del file3
- Bagaimana caranya mengambil parameter file1, file2 dan file3 diatas ?
  - Jawab: Dengan menambah parameter pada main program
  - int main(int argc, char *argv[ ])

Command-line Arguments

```
#include <stdio.h>
int  main( int argc,  char *argv[ ])
{
    int i;
    printf("Jumlah argument=%d\n",argc);
    for(i=0; i<argc; i++) {
        printf("Argument ke-%d = %s\n",i,argv[i]);
    }
    return 0;
}
```

Jika program dijalankan dari DOS Command atau Console Linux sbb:

```
C>Prog  argc1  argc2  argc3
Argument ke-0 = Prog
Argument ke-1 = argc1
Argument ke-2 = argc2
Argument ke-3 = argc3
```

Command Execution

- Untuk menjalankan command DOS atau Linux, bisa digunakan library function system(char *)
- Contoh:

```
#include <stdio.h>
int main(){
    char str[]="Selamat Datang di Binus\n";
    printf(str);
    system("date");
    return 0;
}
```

Output :

```
Selamat Datang di Binus
The current date is: 08/07/2007
Enter the new date: (dd-mm-yy)
```

ALOKASI MEMORI SECARA DINAMIK

Alokasi memori: meminta sejumlah memori (RAM) yang dikelola sistem operasi untuk digunakan

Disalokasi memori: mengembalikan sejumlah memori (RAM) yang tidak diperlukan lagi kepada sistem operasi

```
int main() {
  long nim;
  float ip;
  ...
}
```

nim adalah variabel, **nim** memerlukan memori untuk menampung data

ip adalah variabel, **ip** memerlukan memori untuk menampung ada

ALOKASI MEMORI SECARA DINAMIK

Alokasi memori untuk digunakan menampung data:

1. Alokasi memori secara **statik**
   - memori yang dialokasi dapat diberi nama ▸ **variabel**
   - dialokasi pada saat compile time
   - dialokasi pada local stack memory
   - tidak berubah selama program berlangsung
   - di disalokasi saat program selesai
2. Alokasi memori secara **dinamik**
   - memori yang dialokasi dapat diberi nama
   - dialokasi pada saat run-time
   - dialokasi pada heap memory
   - dapat didisalokasi kapan saja

STATIC MEMORY ALLOCATION

```
struct tdata {
  char nim[9];
  char nama[26];
  float ipk;
} ;

int main () {
  struct tdata data[100];
  int i;
  ...
}
```

Jika sebuah *integer* memerlukan memori 4 *byte* dan sebuah *float* 8 *byte* maka deklarasi variabel data dan i di atas memerlukan memori sebesar 100 x (9 + 26 + 8) + 4 = 4504 *byte*

DYNAMIC MEMORY ALLOCATION

***Function* malloc()** digunakan untuk mengalokasi satu blok memori secara dinamis dari *heap memory*. Argumen (*actual parameter*) yang dikirim kepada *function* ini berupa nilai yang menyatakan jumlah memori (dalam ukuran *byte*) yang ingin dialokasi

Sintak : `void * malloc (size_t size);`

size menyatakan jumlah *byte* yang minta dialokasi.
Jika alokasi berhasil dilakukan maka akan dikembalikan alamat awal blok memori tersebut. Jika alokasi tidak berhasil (misalnya ukuran memori yang minta dialokasi lebih besar daripada jumlah *heap memory* yang tersedia) akan dikembalikan Null. Contoh:

```
int *a;
a = (int *) malloc(sizeof(int));
```

DYNAMIC MEMORY ALLOCATION

***Function* calloc()** digunakan untuk mengalokasi beberapa blok atau elemen memori secara dinamis dari *heap memory* dan setiap element berukuran *size*. Fungsi calloc(), secara otomatis akan meng-inisialisasi semua elemen dengan NOL.

Sintak :

```
void *calloc(size_t memblock, size_t size);
```

size menyatakan jumlah *byte* perblock / per element dan memblock menyatakan jumlah blok atau element.
Jika alokasi berhasil dilakukan maka akan dikembalikan alamat awal blok memori tersebut. Jika alokasi tidak berhasil (misalnya ukuran memori yang minta dialokasi lebih besar daripada jumlah *heap memory* yang tersedia) akan dikembalikan Null. Contoh:

```
int *a;
a = (int *) calloc(20,sizeof(int));
```

DYNAMIC MEMORY ALLOCATION

***Function* realloc()** digunakan untuk merealokasikan kembali beberapa blok atau elemen memori secara dinamis dari *heap memory* dan setiap element berukuran *size*.

Sintak :

```
void *realloc(void *ptr, size_t size);
```

size menyatakan jumlah *byte* perblock / per element dan ptr adalah pointer ke suatu alokasi memori yang sizenya akan dirubah.
Jika alokasi berhasil dilakukan maka akan dikembalikan alamat awal blok memori tersebut. Jika alokasi tidak berhasil (misalnya ukuran memori yang minta dialokasi lebih besar daripada jumlah *heap memory* yang tersedia) akan dikembalikan Null. Contoh:

```
int *a, *ptr;
a = (int *) realloc(ptr,sizeof(int));
```

DYNAMIC MEMORY ALLOCATION

***Function* free()** digunakan untuk mengembalikan memori (yang dialokasi secara dinamik) kepada *heap memory*. Biasanya free() dilakukan setelah proses pemakaian memori tersebut selesai.

```
Sintak : void free (void  *block);
```

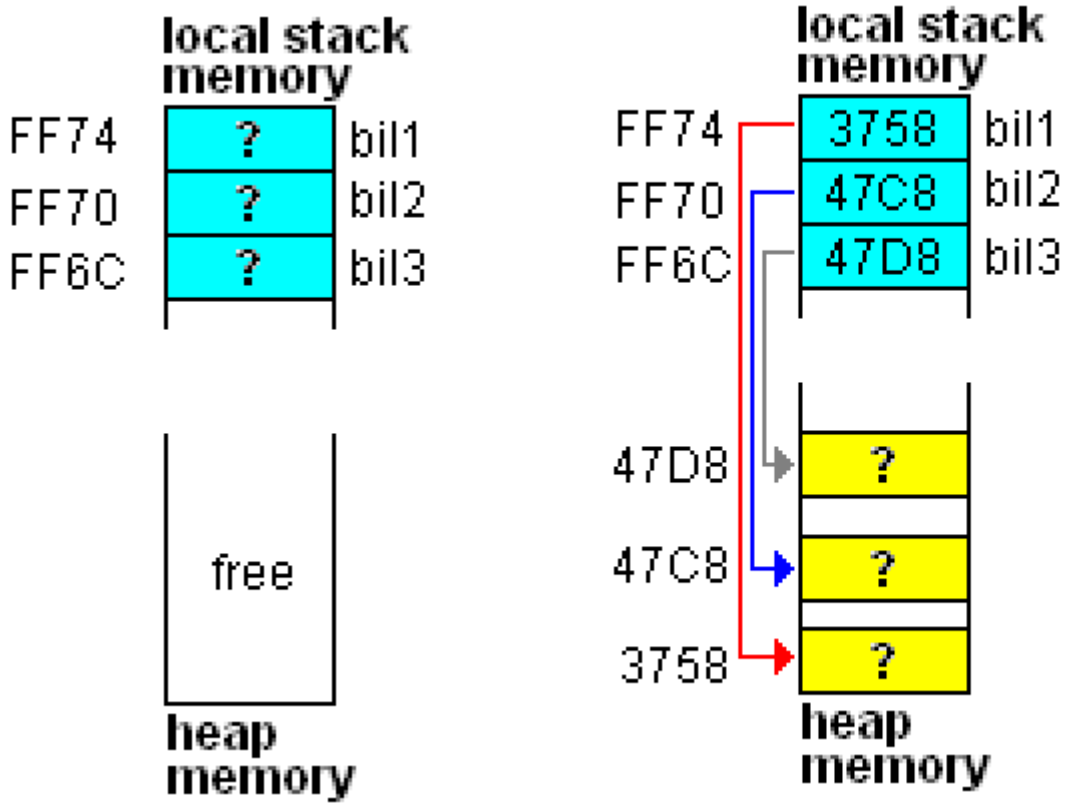
block berupa variabel *pointer* yang digunakan untuk mengacu memori yang dialokasi secara dinamik.

INTEGER ALOKASI DINAMIK

```
# include <stdio.h>
# include <stdlib.h>

int main() {
  int *bil1, *bil2, *bil3;
  bil1 = (int *) malloc (sizeof(int));
  bil2 = (int *) malloc (sizeof(int));
  bil3 = (int *) malloc (sizeof(int));
  scanf("%d %d", bil1, bil2);
  *bil3 = *bil1 * *bil2;
  printf("%d x %d = %d\n", *bil1, *bil2, *bil3);
  free(bil1); free(bil2); free(bil3);
  return 1;
}
```

INTEGER ALOKASI DINAMIK

```
int *bil1, *bil2, *bil3;
bil1 = (int *) malloc (sizeof(int));
bil2 = (int *) malloc (sizeof(int));
bil3 = (int *) malloc (sizeof(int));
```



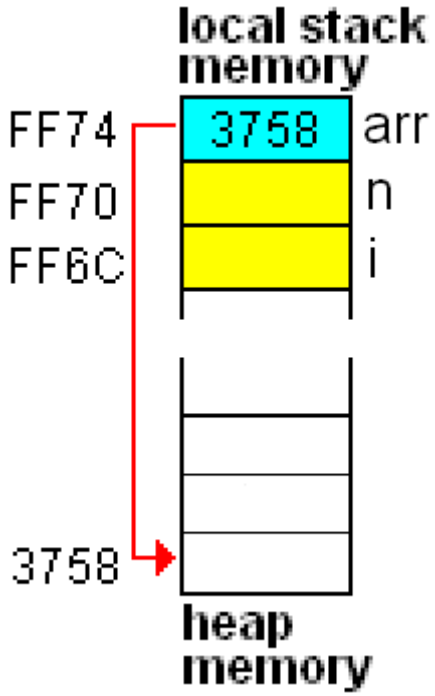
ARRAY DINAMIK

```
# include <stdio.h>
# include <stdlib.h>
int main() {
  int *arr, n, i;
  do {
    fflush(stdin);
    printf("jumlah elemen ? ");
    scanf("%d", &n);
    if (n == 0) break;
    arr = (int *) malloc (n * sizeof(int));
    printf("masukkan %d bilangan: ", n);
    for (i = 0; i < n; i++) scanf(“%d", &arr[i]);
    printf("dibalik: ");
    for (i = n - 1; i >= 0; i--) printf("%d ", arr[i]);
    printf("\n\n");
    free(arr);
  }while (1);
  return 1;
}
```

ARRAY DINAMIK

```
int *arr;
printf("jumlah elemen ? ");
scanf("%d", &n);
arr = (int *) malloc (n * sizeof(int));
```

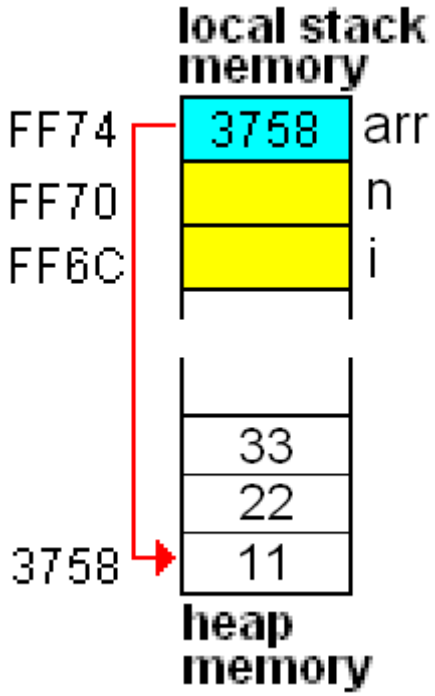
```
jumlah elemen ? 3
```



ARRAY DINAMIK

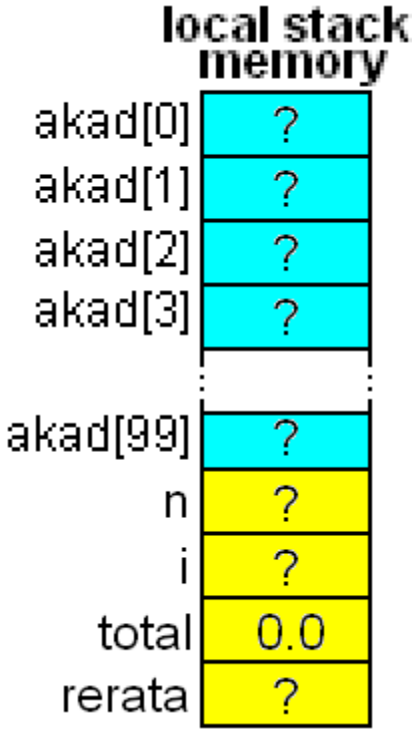
```
printf("masukkan %d bilangan: ", n);
for (i = 0; i < n; i++) scanf("%d", &arr[i]);
printf("dibalik: ");
for (i = n - 1; i >= 0; i--) printf("%d ", arr[i]);
```

```
jumlah elemen ? 3
masukkan 3 bilangan: 11 22 33
dibalik: 33 22 11
```

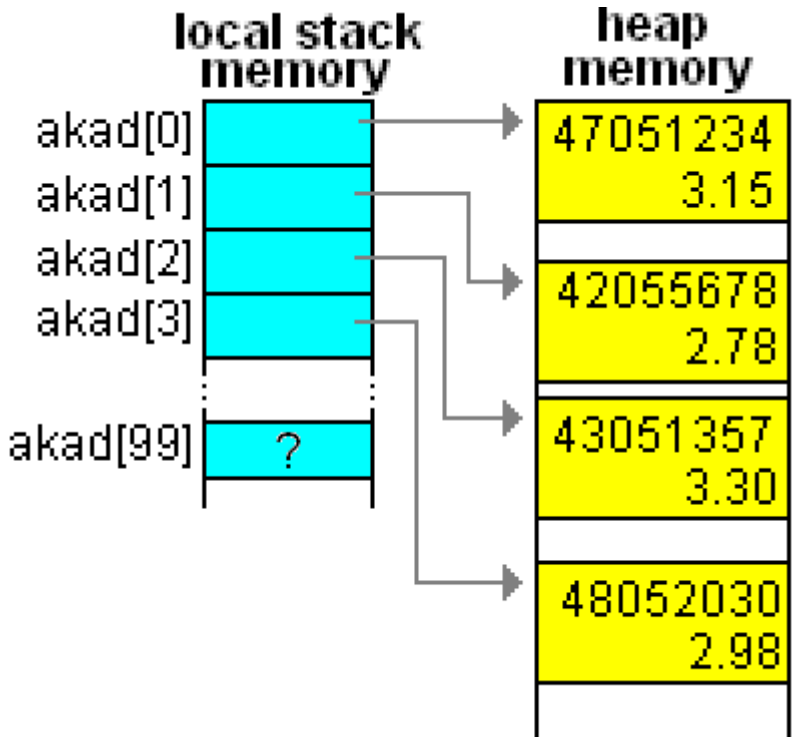


ARRAY OF POINTER OF DYNAMIC STRUCTURE

```
# include <stdio.h>
# include <stdlib.h>

struct takad {
  char nim[9];
  float ipk;
} ;

int main() {
  struct takad *akad[100];
  int n, i;
  float total = 0, rerata;
  ...
}
```



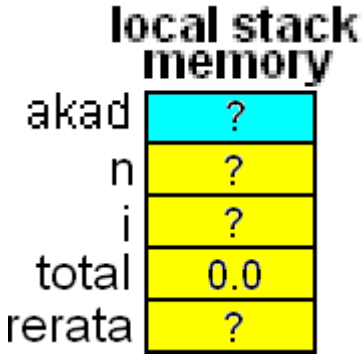
ARRAY OF POINTER OF DYNAMIC STRUCTURE




```
akad[i]= (struct takad*) malloc(sizeof(struct takad));
```

POINTER OF ARRAY OF DYNAMIC STRUCTURE

```
# include <stdio.h>
# include <stdlib.h>

struct takad {
  char nim[9];
  float ipk;
} ;

int main() {
  struct takad *akad;
  int n, i;
  float total = 0, rerata;
```



MAKRO

- DIDEFINISIKAN MENGGUNAKAN #define.
- UMUMNYA DILETAKKAN DI AWAL PROGRAM
- LINGKUPNYA MULAI DARI POSISI IA DIDEFINISI-KAN SAMPAI AKHIR PROGRAM.
- MAKRO YANG DIDEFINISIKAN DI SUATU PROGRAM TIDAK DIKENAL DI PROGRAM LAIN

MAKRO

```
#include <stdio.h>

#define ERROR printf("Error..\n")

int main() {
    int x=0;
    int y=35;
    if(x == 0) ERROR;
    else y=y/x;
    return 1;
}
```

MAKRO (denganParameter)

```
#include <stdio.h>

#define SUM(x, y)(x+y)

int main() {
    int x=10;
    int y=35;
    printf("%d\n",SUM(x,y));
    return 1;
}
```

MAKRO (dengan Multi line)

```
#include <stdio.h>
#define Segitiga(n) for(brs=1; brs<=n; brs++){          \
                for(i=1; i<=brs; i++) printf("%s","* ");      \
                printf("\n");                                          \
              }

int main() {
    int i, brs;
    Segitiga(4);
    return 1;
}
```

Pointer to Functions

- Pointer to function berisi alamat dari sebuah fungsi di dalam memori
- Sintak:

  return_type (* nama_pointer)(parameter);
- Contoh:

  int (*compare)(int a, int b);

  Artinya: compare adalah nama pointer to function, yang menunjuk sebuah fungsi yang mengembalikan nilai tipe integer, dan fungsi tersebut mempunyai 2 parameter bertipe integer
- Perhatikan bedanya dengan deklarasi dibawah:

  int compare (int a, int b);

  Artinya: compare adalah nama fungsi yg mengembalikan tipe integer dan mempunyai 2 parameter bertipe integer

Pointer to Functions

```
#include <stdio.h>
int fungsi1(int a);
int fungsi2(int b);

int main()
{
    int x;
    int(*f)(int);
    f=fungsi1;
    x=(*f)(3);
    printf("x=%d\n",x);
    f=fungsi2;
    x=(*f)(13);
    printf("x=%d\n",x);
    getch();
    return(0);
}
```

```
int fungsi1(int a)
{
    a++;
    return a;
}

int fungsi2(int b)
{
    b = b * b;
    return b;
}
```

C Preprocessor Directive

- Pemrosesan dilakukan sebelum program dikompilasi.
- Preporcessor directive diawali dengan tanda #
- Contoh :

#include <namafile>
#include ”namafile”
#define identifer replacement-text
Contoh :
#define PHI 3.14

Sudah dibahas pada pertemuan 3-4

Preprocessor #define juga bisa digunakan untuk membuat Macro

C Preprocessor Directive

- Conditional Compilation

```
#if expresi
    ….
#endif

Atau

#if expresi
   …..
#else
   ……
#endif
```

Preprocessor operator **defined** :
Syntax :

defined(*identifier* **)**
defined *identifier*

Akan mengembalikan nilai true (nonzero) jika *identifier* didefinisikan dan false (0), jika *identifier* tidak didefinisiakn

C Preprocessor Directive

```
#include <stdio.h>
#include <stdlib.h>
#define DEVC
int main()
{
  int i;  clrscr();
  for(i=1; i<79; i++){
      gotoxy(i,10); printf(" =>");
      #if defined TURBOC
      delay(60);
      #else
      sleep(60);
      #endif
  }
  getch();
  return 0;
}
```

Pada Dev-C tidak ada fungsi delay(), seperti pada Turbo C, tetapi ada fungsi sleep() yang artinya sama.

Jika ingin membuat program yang bisa dikompilasi pada Turbo C dan Dev-C maka bisa digunakan Preprocessor Conditional Compilation seperti disamping.

Program ini untuk dikompilasi di Dev-C

C Preprocessor Directive

```
#include <stdio.h>
#include <stdlib.h>
#define TURBOC
int main()
{
  int i;  clrscr();
  for(i=1; i<79; i++){
      gotoxy(i,10); printf(" =>");
      #if defined TURBOC
      delay(60);
      #else
      sleep(60);
      #endif
  }
  getch();
  return 0;
}
```

Program ini untuk dikompilasi di Turbo C

Latihan-1

- Pada <malloc.h> ada beberapa fungsi seperti dibawah. Jelaskan kegunaan fungsi tersebut !

  - **void *malloc(size_t** *size* **);**
  - **void *calloc(size_t** *num*, **size_t** *size* **);**
  - **void *realloc(void **memblock*, size_t *size*);**
  - **void free(void **memblock*);**

Latihan-2

```
void (*ptr[3])(int)={function1. function2, function3 };
```

Jelaskan apa maksud/arti dari statement diatas !

Latihan-3

- Jelaskan arti dari preprocessor directive sbb:

```
1. #ifdef
     ….
   #endif
2. #error
3. #pragma
4. #line
5. #ifndef
     ……
   #endif
```

Latihan-4

- Bahasa C menyediakan predefined symbolic constant sbb:

  __LINE__

  __FILE__

  __DATE__

  __TIME__

Jelaskan maksud dari predefined symbolic constant diatas !

Latihan-5

- Jelaskan untung rugi Macro dengan Fungsi !

Pertemuan 13

File

Learning Outcomes

Pada akhir pertemuan ini, diharapkan mahasiswa akan mampu :

- Mendemonstrasikan program yang berisi aplikasi untuk membaca, menulis (read, write) data ke file text atau biner

Outline Materi

- Stream
- Definisi file
- Tipe file
- File teks dan biner
- Fungsi Input Output file

STREAM

- Supaya data yang di-key-in melalui keyboard tidak hilang (saat komputer dimatikan) maka perlu disimpan ke dalam secondary storage device sebagai file data

- **Stream** ialah kumpulan karakter yang disusun dalam baris-baris yang berpindah dari satu media ke media lain pada sistem komputer. Semua data masukan dan keluaran berupa *stream*. Bahasa C memperlakukan file sebagai stream.

STREAM

Saat program C dijalankan maka tiga buah stream standar **OTOMATIS** diaktifkan:

1. **Standard Input Stream**,
   mengatur aliran data masukan melalui ketikan keyboard
2. **Standard Output Stream**,
   mengatur aliran data keluaran ke layar monitor
3. **Standard Error Stream**,
   mengatur tampilan pesan kesalahan ke layar monitor

Masing-masing stream berasosiasi dengan sebuah file.

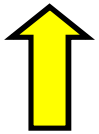
File

- **Definisi File**
  - File adalah kumpulan record
  - Record adalah kumpulan field
  - Field kumpulan byte
  - Byte dalah kumpulan bit

FILE

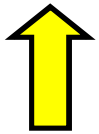
Operasi membuka (mengaktifkan) sebuah file mengakibatkan sebuah **POINTER** dikembalikan kepada instruksi pemanggil. Pointer ini menunjuk ke sebuah struktur data bertipe **FILE** yang sudah didefinisikan di stdio.h

Standard input stream
Standard output stream
Standard error stream



stream

- **stdin**
- **stdout**
- **stderr**



File pointer

FILE

```
typedef struct  {
  int           level;          // fill/empty level of buffer
  unsigned  flags;          // File status flags
  char        fd;             // File descriptor
  unsigned  char  hold;    // Ungetc char if no buffer
  int           bsize;          // Buffer size
  unsigned char   *buffer;   // Data transfer buffer
  unsigned char   *curp;     // Current active pointer
  unsigned  istemp;       // Temporary file indicator
  short        token;       //Used for validity checking
} FILE;
```

- **FILE *filepointer;**

JENIS FILE

TEXT FILE ialah berkas yang disimpan dalam format teks**.**
Biasanya diistilahkan dengan **FILE ASCII**.

- besarnya storage penyimpanan disesuaikan dengan jumlah angka bilangan: 10000 perlu 5 byte
- isi file dapat ‘dibaca’ dengan menggunakan editor teks
- isi file dapat ‘dibaca’ dengan c:>TYPE namafile

BINARY FILE menyimpan data numerik dalam format yang tetap sesuai ketentuan *micro-processor* (misanya dengan format *sign-magnitude 2’s complement*), tidak bergantung pada jumlah digit bilangan.

Buffer Area

- Buffer area adalah bagian dari memori yang digunakan sebagai tempat penampungan sementara sebelum data dipindahkan ke file.
- Sintaks pembentukan buffer area

  FILE *fp;

  Dimana **fp** adalah variabel file pointer yang digunakan untuk menunjuk awal buffer area.
- Dikenal juga dengan nama stream pointer.

Membuka File

- **Untuk membuka file digunakan fungsi fopen(), sintak sbb:**

  FILE *fopen(const char *_filename_, const char *_mode_);

- **Fungsi fopen() didefinisikan di <stdio.h>**

- **Fungsi fopen() mengembalikan pointer ke awal buffer area. Nilai null dikembalikan jika berkas tidak dapat dibuka.**

Membuka File

- Nilai mode operasi yang mungkin adalah

Mode	**Makna**
"**r**"	Membuka file utuk dibaca.
"**w**"	Membentuk file baru untuk ditulisi.
"**a**"	Membuka file untuk tambah data.
"**r+**"	Membuka file untuk dibaca/ditulisi.
"**w+**"	Membentuk file untuk dibaca/ditulisi.
"**a+**"	Membuka file untuk dibaca dan ditambah isinya
"rb"	Membuka file biner untuk dibaca
"wb"	Membentuk file biner untuk ditulid

Menutup File

- File ditutup dengan menggunakan fungsi fclose(), sintak :

  int fclose(FILE ****stream* **);

- Fungsi fclose() didefinisikan di <stdio.h>
- Fungsi fclose() akan mengembalikan nilai 0 jika sukses, dan EOF jika error
- EOF (End Of File) nilainya -1
- Fungsi fclose() akan membebaskan buffer agar dapat digunakan oleh file lain dan mengirim data yang masih tertinggal di buffer agar segera dikirim ke file.

Menutup File

- Fungsi fcloseall() dengan sintak :

int fcloseall (void);

✓ Function ini menutup seluruh stream yang aktif **KECUALI** stdin, stdout, stdprn, stderr, dan stdaux.

✓ Apabila proses berhasil maka dikembalikan bilangan yang menyatakan jumlah stream yang berhasil ditutup. Apabila terjadi kesalahan maka dikembalikan EOF.

✓ Header file (stdio.h>

Fungsi Input Output File

- **fgetc**
  - **Membaca satu karakter dari file**
  - **fgetc(stdin) ekuivalen dengan getchar()**
  - **Sintak : int fgetc(FILE **stream*);**
  - **Mengembalikan karakter yang dibaca jika sukses, dan mengembalikan EOF jika error**

- **fputc**
  - **Menulis satu karakter ke file**
  - **fputc('a', stdout) ekuivalen dengan putchar('a')**
  - **Sintak: int fputc(int *c*, FILE **stream*);**
  - **Mengembalikan karakter yang ditulis jika sukses, dan mengembalikan EOF jika error**

Fungsi Input Output File

- **fgets**
  - **Sintak: char *fgets(char *s*tring*, int *n*, FILE *s*tream*);**
  - **Membaca satu baris dari file yang diakhiri dengan new line, atau maximum n-1 karakter**
  - **Mengembalikan string jika sukses dan mengembalikan NULL jika error**

- **fputs**
  - **Menulis satu baris ke file**
  - **Sintak: int fputs(const char *s*tring*, FILE *s*tream*);**
  - **Mengembalikan nilai non-negatif jika sukses, dan mengembalikan EOF jika error**

Fungsi Input Output File

- **fscanf**
  - **Sintak: int fscanf(FILE **stream*, const char **format* [, *argument*]...);**
  - **Membaca data dari file dengan format sesuai dengan format scanf.**
  - **Mengembalikan nilai yang menyatakan jumlah field yang sukses dibaca, dan mengembalikan EOF jika error**

- **fprintf**
  - **Sintak: int fprintf(FILE **stream*, const char **format* [, *argument*]...);**
  - **Menulis data ke file dengan format sesuai dengan format printf.**
  - **Mengembalikan nilai yang menyatakan jumlah byte yang sukses ditulis, dan mengembalikan nilai negatif jika error**

Fungsi Input Output File

- **fwrite**
  - **Sintak: size_t fwrite(const void **buffer*, size_t *size*, size_t *count*, FILE **stream*);**
  - **Menulis satu block data yg ada di buffer ke file**
  - **Mengembalikan jumlah byte data yang ditulis jika sukses, dan error jika return value nya lebih kecil dari size data yang ditulis**
- **fread**
  - **Sintak: size_t fread(void **buffer*, size_t *size*, size_t *count*, FILE **stream*);**
  - **Membaca satu block data sebesar size dari file**
- **feof**
  - **Sintak : int feof(FILE **stream*);**
  - **Untuk ngetest apakah posisi pointer sudah di end-of-file**
  - **Mengembalikan 0 jika belum end-of-file**

Fungsi Input Output File

- Contoh :

```
fwrite( &mhs, sizeof( mhs ), 1, fp );
```

  - &mhs = lokasi asal data
  - sizeof(mhs) = mengembalikan nilai ukuran dari mhs
  - 1 => 1 kali write sebesar sizeof(mhs)
  - fp = file pointer

File : Contoh

- Contoh :

```
#include <stdio.h>
void main( void ) {
    FILE *stream;
    char *p, buffer[] = "This is the line of output\n";
    int ch; ch = 0;
    stream = stdout;
    for( p = buffer; (ch != EOF) && (*p != '\0'); p++ )
        ch = putc( *p, stream );
}
```

Output:

This is the line of output

File : Contoh

- **Contoh program untuk membaca file fgetc.c**

```
#include <stdio.h>
int main(void)
{
    char ch;   FILE *fp;
    fp=fopen("fgetc.c","r");
    if(fp==NULL){
        printf("File fgetc.c tidak bisa dibuka\n");   exit(1);
    }
    while(!feof(fp)){
        ch=fgetc(fp);   printf("%c",ch);
    }
    fclose(fp);
    return 0;
}
```

File : Contoh

- Contoh menulis string ke file test.txt dengan fputc

```
#include <stdio.h>
int main(void)
{
    FILE *fp; int i;
    char ss[80]="Kalimat ini disimpan ke file test.txt dengan fputc";
    fp=fopen("test.txt","w");
    if(fp==NULL){
        printf("File test.txt tidak bisa create\n");   exit(1);
    }
    for(i=0; i<strlen(ss); i++)   fputc(ss[i], fp);
    fclose(fp);
    return 0;
}
```

- Contoh membaca file fgets.c dengan fgets

```
#include <stdio.h>
int main(void)
{
   FILE *fp;
   char ss[80];
   fp=fopen("fgets.c","r");
   if(fp==NULL){
      printf("File fgets.c tidak bisa di open\n");   exit(1);
   }
   while(fgets(ss, 80, fp)) printf("%s",ss);
   fclose(fp);
   return 0;
}
```

File : Contoh

Contoh menulis string ke file test.txt dengan fputs

```
#include <stdio.h>
int main(void)
{
      FILE *fp;
      char ss[80]="Kalimat ini ditulis ke file test.txt dengan fputs";
      fp=fopen("test.txt","w");
      if(fp==NULL){
          printf("File test.txt tidak bisa di create\n");
         exit(1);
      }
      fputs(ss, fp);
      fclose(fp);
      return 0;
}
```

File : Contoh

Contoh menulis data ke file test.txt dengan fprintf

```
#include <stdio.h>
int main(void)
{
   FILE *fp;
   fp=fopen("test.txt","w");
   if(fp==NULL){
      printf("File test.txt tidak bisa di create\n");
      exit(1);
   }
   fprintf(fp,"%d %s %f\n",1,“Amir", 3.95);
   fprintf(fp,"%d %s %f\n",2,“Tono", 3.15);
   fclose(fp);
   return 0;
}
```

Contoh membaca data dari file test.txt dengan fscanf

```
#include <stdio.h>
int main(void)
{
      FILE *fp;  int no; char nama[20]; float ipk;
      fp=fopen("test.txt","r");
      if(fp==NULL){
          printf("File test.txt tidak bisa di open\n");   exit(1);
      }
      fscanf(fp,"%d %s %f",&no,nama, &ipk);
      printf("%d %s %f\n",no,nama,ipk);
      fscanf(fp,"%d %s %f",&no,nama, &ipk);
      printf("%d %s %f\n",no,nama,ipk);
     fclose(fp);     return 0;
}
```

Contoh menulis data ke file biner test.dat dengan fwrite

```
#include <stdio.h>

int main(void)
{
   FILE *fp;
   int Arr[]={1,2,3,4,5};
   fp=fopen("test.dat","w");
   if(fp==NULL){
      printf("File test.dat tidak bisa di create\n");
      exit(1);
   }
   fwrite(Arr,sizeof(Arr),1,fp);
   fclose(fp);
   return 0;
}
```

File : Contoh

Contoh membaca data dari file biner test.dat dengan fread

```
#include <stdio.h>
int main(void)
{
   FILE *fp; int i;
   int Arr[5];
   fp=fopen("test.dat","r");
   if(fp==NULL){
      printf("File test.dat tidak bisa di open\n");
      exit(1);
   }
   fread(Arr,sizeof(Arr),1,fp);
   for(i=0; i<5; i++) printf("%d ",Arr[i]);
   fclose(fp);
   return 0;
}
```

Latihan-1

Sebuah text file berisi data tanggal lahir karyawan dengan format tgl/bln/thn. Tgl, bln dan thn masing-masing hanya 2 angka. Contoh sbb:

01/06/50
03/06/51
10/02/54
08/01/48
26/08/51
27/04/54
21/09/51
...dst....

Baca file tsb, kemudian tentukan jumlah karyawan yang umurnya sbb:

- **Diatas 51 thn**
- **Antara 44 – 51 Thn**
- **Antara 36 – 43 Thn**
- **Antara 28 – 35 Thn**
- **Dibawah 28 Thn**

(Catatan : Umur = Thn sekarang – thn lahir)

Latihan-2

```
struct Mhs{
   char nama[20];
   int nim;
   float ipk;
};
```

Buat file biner dengan menggunakan fwrite, untuk menyimpan 5 record data Mahasiswa dengan struc Mhs seperti diatas. Data nama, nim dan ipk di input dari keyboard. Nama file dibuat dengan nama Mhs.dat

Latihan-3

Bacalah file Mhs.dat pada latihan sebelumnya diatas, dengan fungsi fread, kemudian tampilkan hasilnya dilayar monitor dengan format sbb:

Nim	Nama	Ipk
------	--------	----

Latihan-4

Bukalah file **Mhs.dat** pada latihan sebelumnya diatas, kemudian append (tambahkan) 5 record lagi data mahasiswa yang diinput dari keyboard.

Latihan-5

- **Jelaskan dan berikan contoh program yang menggunakan fungsi rewind() yang sintak nya sbb:**

  void rewind(FILE *_stream_);

- **Jelaskan dan berikan contoh program yang menggunakan fungsi fseek() yang sintak nya sbb:**

  int fseek(FILE *_stream_, long _offset_, int _origin_);

Latihan-6

- **Jelaskan dan berikan contoh program yang menggunakan fungsi ftell() yang sintak nya sbb:**

  long ftell(FILE **stream*);

- **Jelaskan bagaimanakah caranya untuk mengetahui ukuran (size) dari suatu file ?**

Latihan-7

- Buat program untuk meng copy sebuah file seperti command DOS dibawah ini:

  C>copy test.c coba.c

- Buat program untuk delete sebuah file seperti command DOS dibawah ini:

  C> del test.c

Pertemuan 14

Sorting

Learning Outcomes

Pada akhir pertemuan ini, diharapkan mahasiswa akan mampu :

- Mendemonstrasikan simulasi algoritma sorting

Outline Materi

Searching dan Sorting

- Bubble sort
- Selection sort
- Insertion sort
- Quick sort

Sorting

- Sorting / pengurutan diperlukan untuk mempercepat pencarian suatu target dalam suatu daftar (list)

- **Jenis Pengurutan :**
  - **Ascending**
    Pengurutan dilakukan mulai dari nilai terkecil menuju nilai terbesar
  - **Descending**
    Pengurutan dilakukan mulai dari nilai terbesar menuju nilai terkecil.

- Algoritma sorting:
  1. Internal sorting
     seluruh data yang akan diurutkan dimuat ke dalam RAM

  2. External sorting
     Proses sorting memerlukan bantuan secondary storage

Teknik Sorting

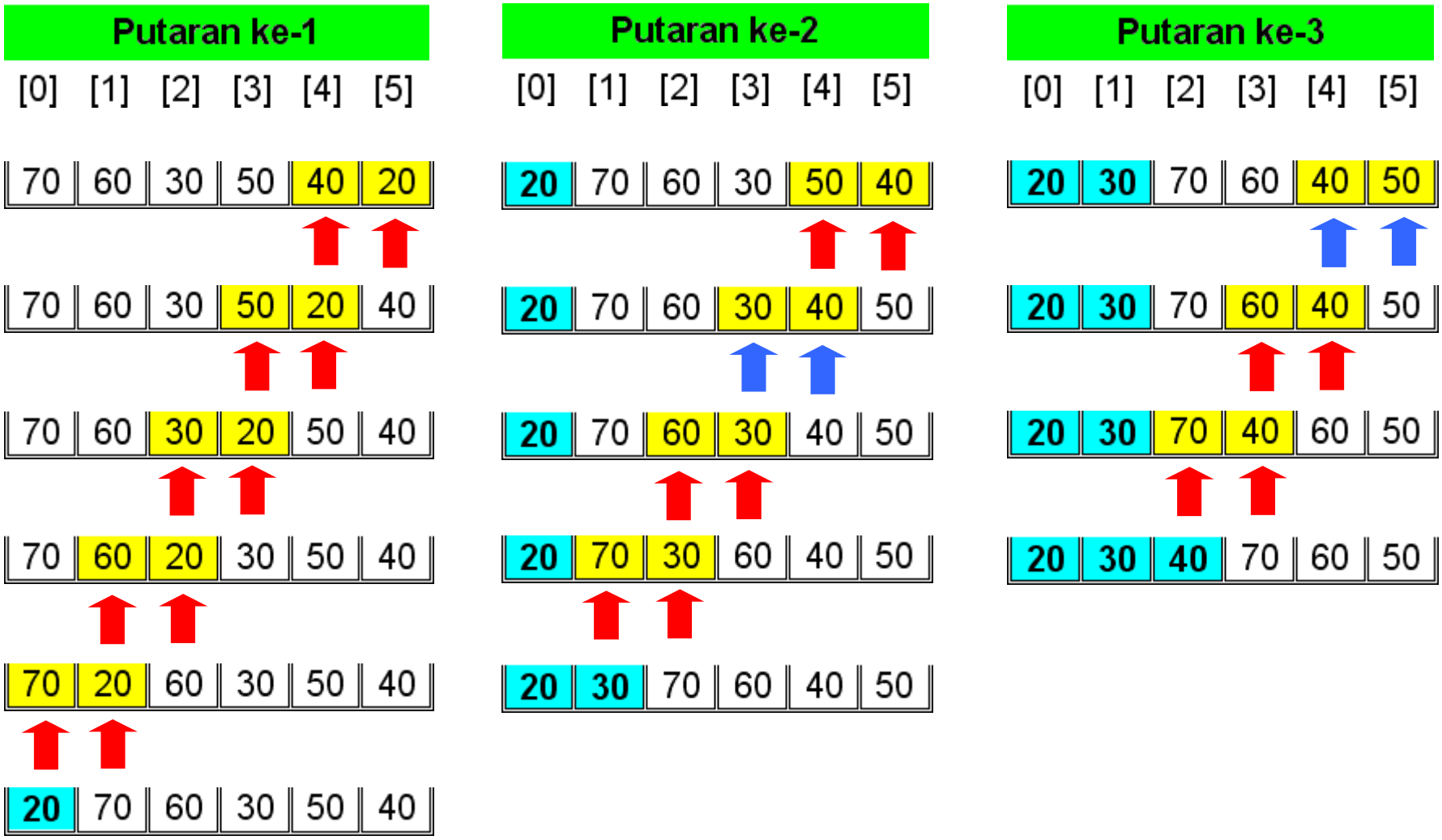
- Sederhana :
  - Bubble sort
  - Selection sort
  - Insertion sort
- Lanjut :
  - Quick Sort
  - Merge Sort

Bubble Sort

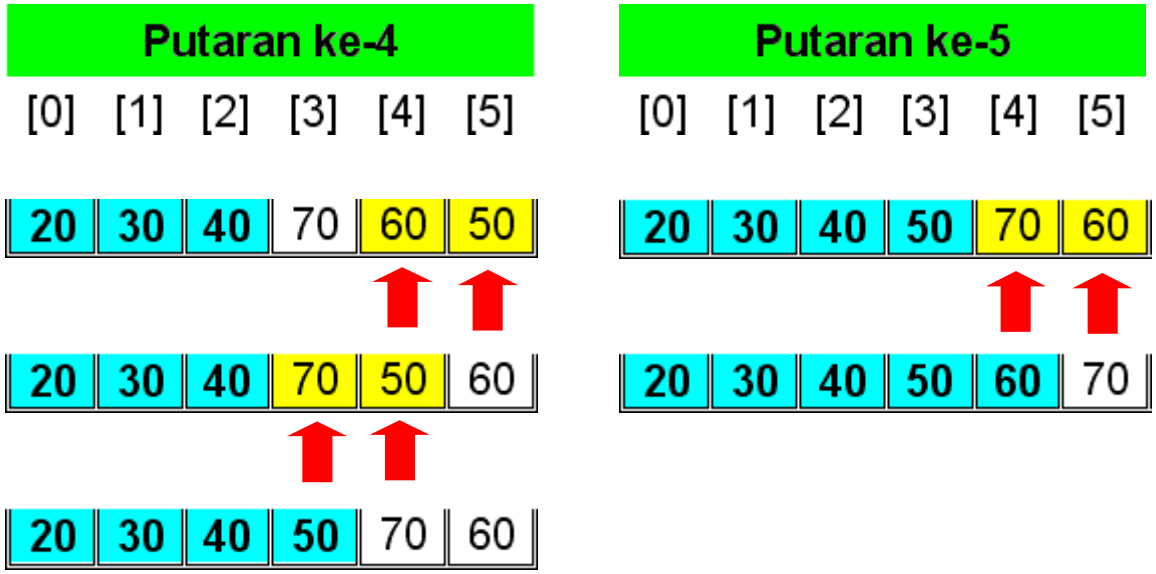
- Membandingkan dua data yang bersebelahan.
- Bandingkan dan tukar (jika perlu)
- Disebut juga exchange sort

```
void Bubble(int *DataArr, int n)
{
        int i, j;
        for(i=1; i<n; i++)
          for(j=n-1; j>=i; j--)
              if(DataArr[j-1] > DataArr[j])
                  Tukar(&DataArr[j-1],&DataArr[j]);
}
```

BUBBLE SORT


Putaran ke-1
[0] [1] [2] [3] [4] [5]
70 60 30 50 40 20
70 60 30 50 20 40
70 60 30 20 50 40
70 60 20 30 50 40
70 20 60 30 50 40
20 70 60 30 50 40
Putaran ke-2
[0] [1] [2] [3] [4] [5]
20 70 60 30 50 40
20 70 60 30 40 50
20 70 60 30 40 50
20 70 30 60 40 50
20 30 70 60 40 50
Putaran ke-3
[0] [1] [2] [3] [4] [5]
20 30 70 60 40 50
20 30 70 60 40 50
20 30 70 40 60 50
20 30 40 70 60 50

BUBBLE SORT


Putaran ke-4
[0] [1] [2] [3] [4] [5]
20 30 40 70 60 50
20 30 40 70 50 60
20 30 40 50 70 60
Putaran ke-5
[0] [1] [2] [3] [4] [5]
20 30 40 50 70 60
20 30 40 50 60 70

Bubble Sort

- Algoritma Bubble sort yang sudah dibahas sebelum, mempunyai kelemahan yaitu jika data sudah terurut maka perbandingan masih dilakukan.

- Perbaikan secara sederhana, yaitu menambah : ‘flag’ yang memberitahu, bila pada suatu putaran data telah terurut. Sering disebut sebagai Bubble-flag.

Bubble Sort

- Program

```
void Bubble_Flag(int *Arr, int n)
{
        int i, j;
        int urut;               /* Flag */
        urut = 0;  i = 1;
        while((i < n) && (!urut)){
          urut = 1;
          for(j=n-1; j>=i; j--){
            if(Arr[j-1] > Arr[j]){
              Tukar(&Arr[j-1], &Arr[j]);
              urut = 0;
            }
          }
          i = i + 1;
        }
}
```

RECURSIVE BUBBLE SORT

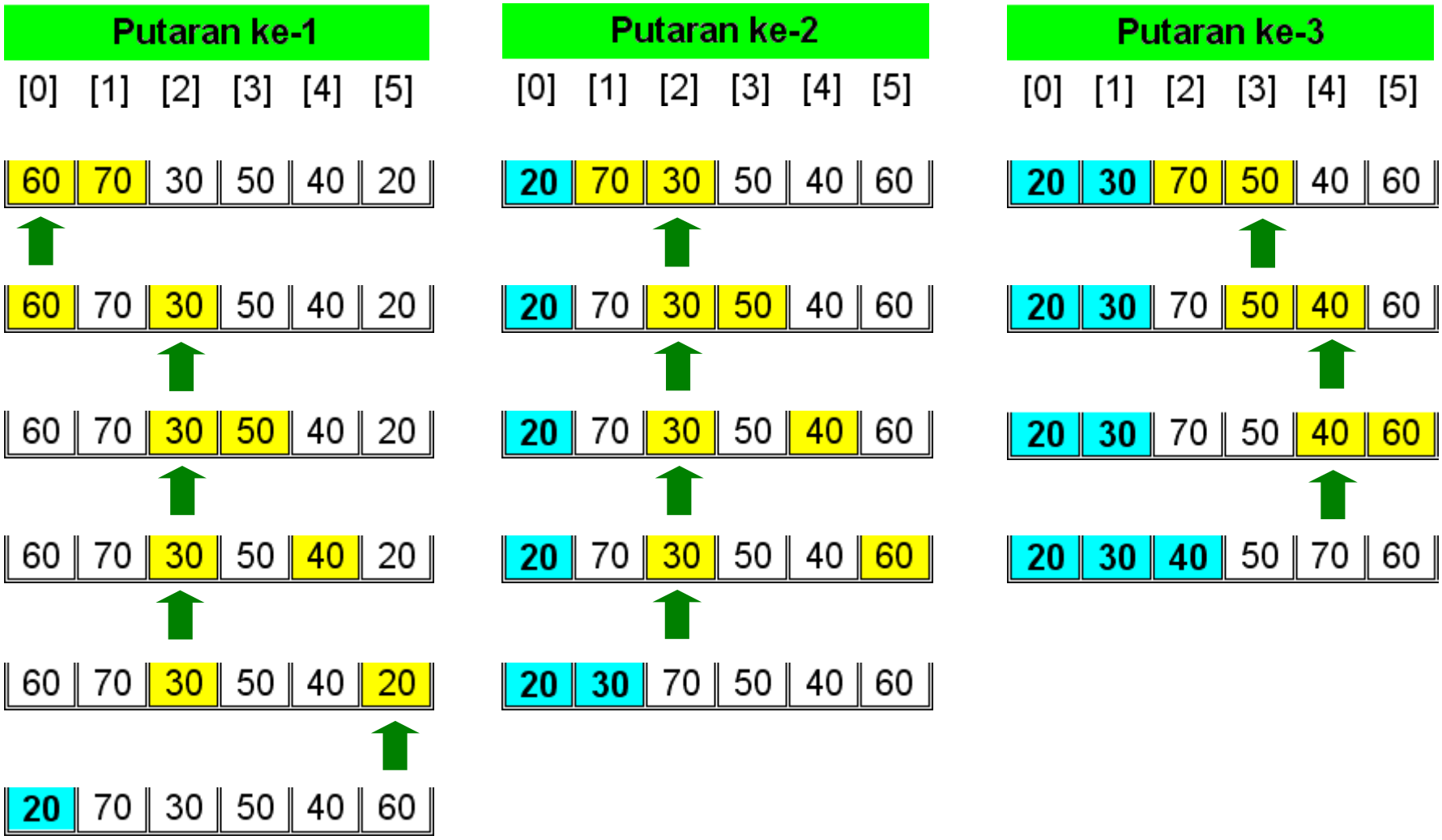
```
void banding_tukar(int arr[], int a, int n) {
  int temp;
  if (a < n) {
    if (arr[n-1] > arr[n]) {
      temp = arr[n-1];
      arr[n-1] = arr[n];
      arr[n] = temp;
    }
    banding_tukar(arr, a, n-1);
  }
}

void bubble_sort(int arr[], int a, int n) {
  if (a < n) {
    banding_tukar(arr, a, n);
    bubble_sort(arr, a+1, n);
  }
}
```

Selection Sort

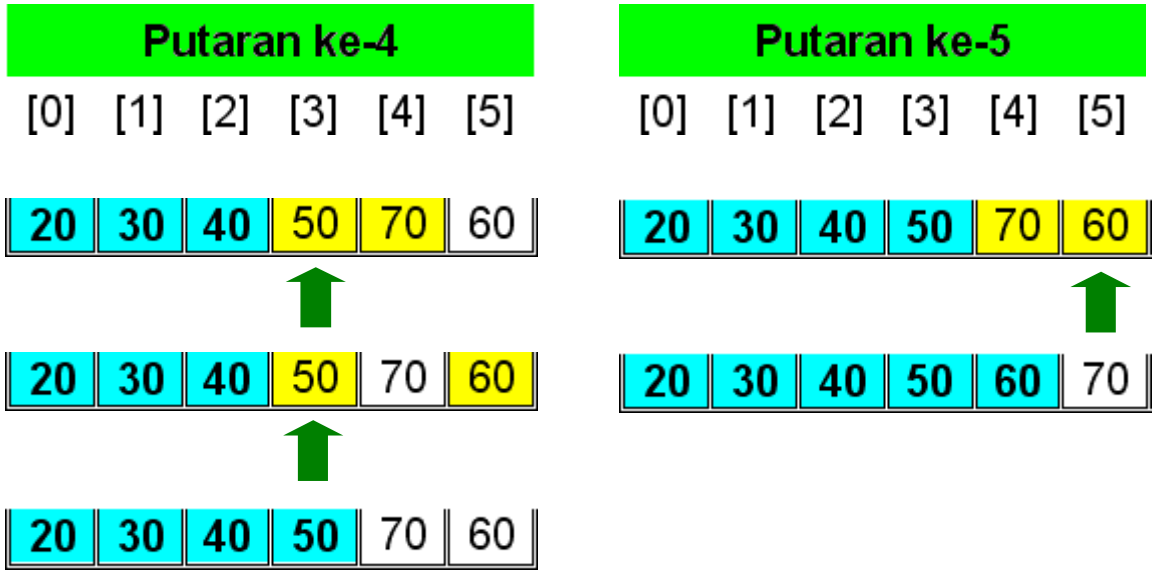
Algoritma :

```
for(i=0; i<=N-2; i++){      /* N=Banyak data dlm daftar */
    for(j=i; j<=N-1; j++){
        Tentukan index dari data terkecil antara A[j] s/d A[N-1],
        dan simpan di variabel k.
        Kemudian tukar A[i] dengan A[k].
    }
}
```

SELECTION SORT



SELECTION SORT



SELECTION SORT

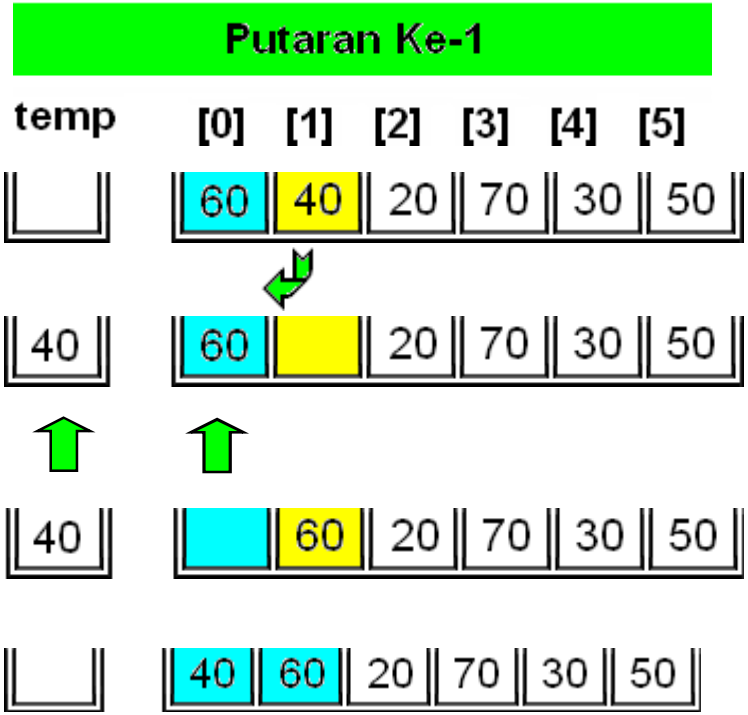
```
void SelectionSort(int data[], int n)
{
    int i, j, idx_kecil;
    for (i = 0;  i < n – 1;  i++) {
        idx_kecil = i;
        for (j = i + 1; j < n; j++) if (data[idx_kecil] > data[j]) idx_kecil = j;
            if (idx_kecil > i) tukar(&data[i], &data[idx_kecil]);
    }
}
```

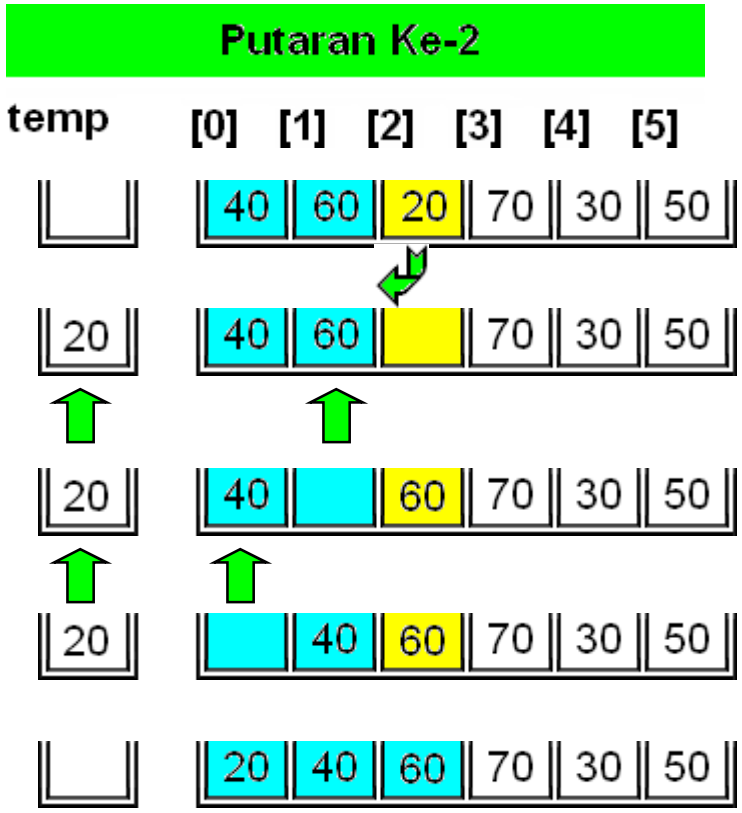
Insertion Sort

- Algoritma :

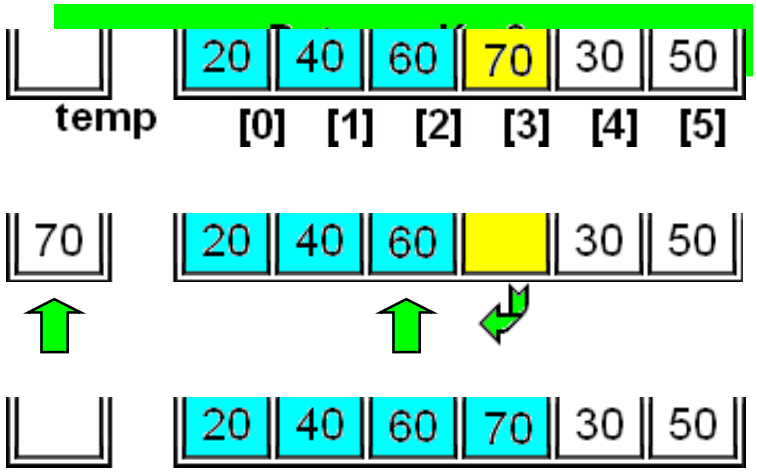
```
for(i=1; i<n; i++) {
    x = A[i], sisipkan x pada tempatnya yang
    sesuai antara A[0] s/d A[i-1].
}
```

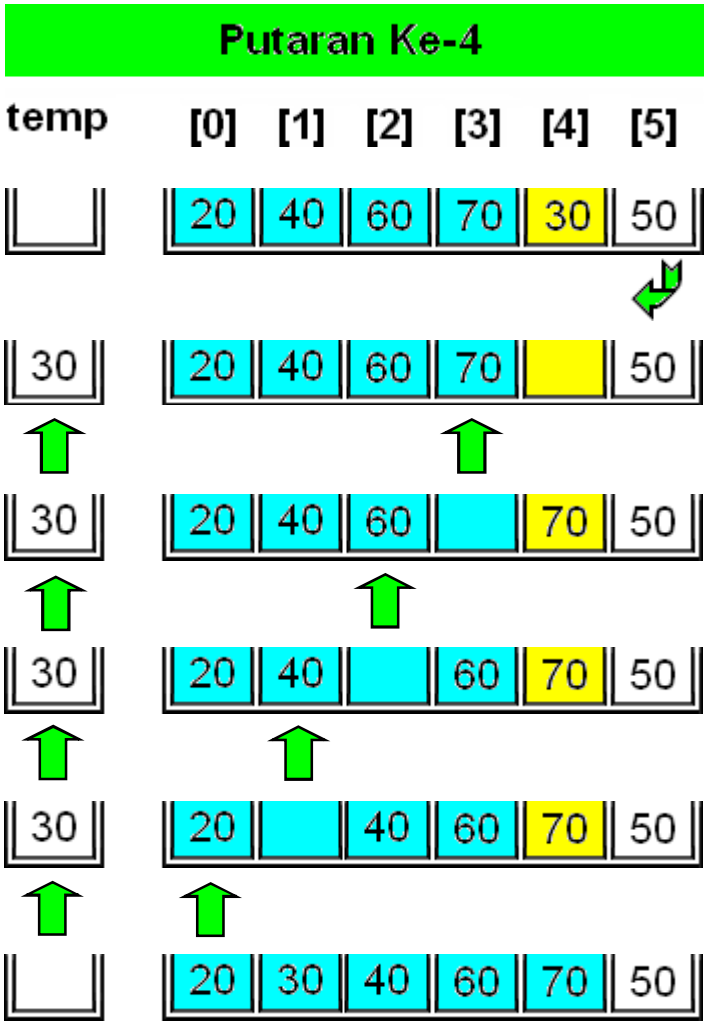
INSERTION SORT



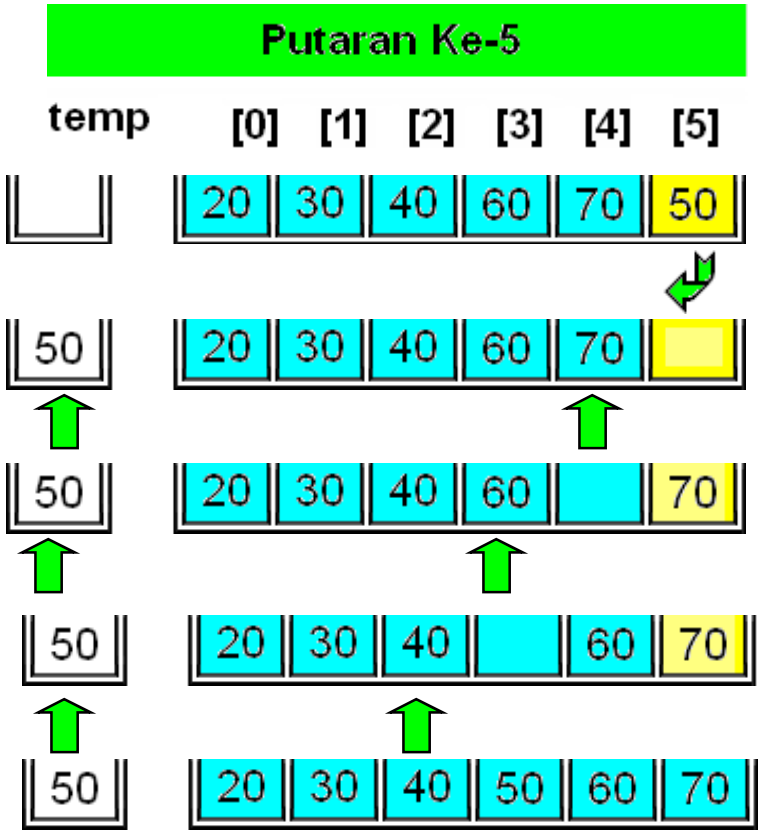


INSERTION SORT





INSERTION SORT



Insertion Sort

- Program

```
void Insertion(int *Arr, int n){
        int i, k, y;
        for(k=1; k < n; k++) {
                y = Arr[k];
                for(i=k-1; i >= 0 && y < Arr[i]; i--)
                        Arr[i+1] = Arr[i];
                Arr[i+1] = y;
        }
}
```

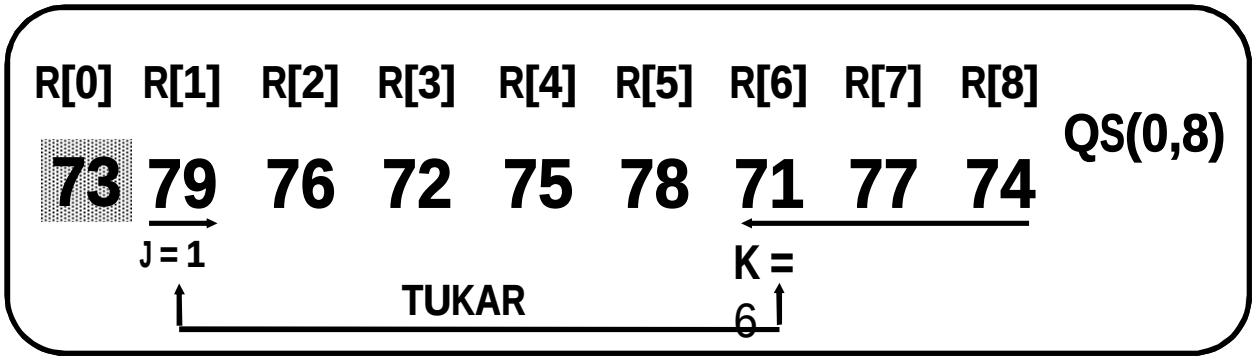
Quick Sort

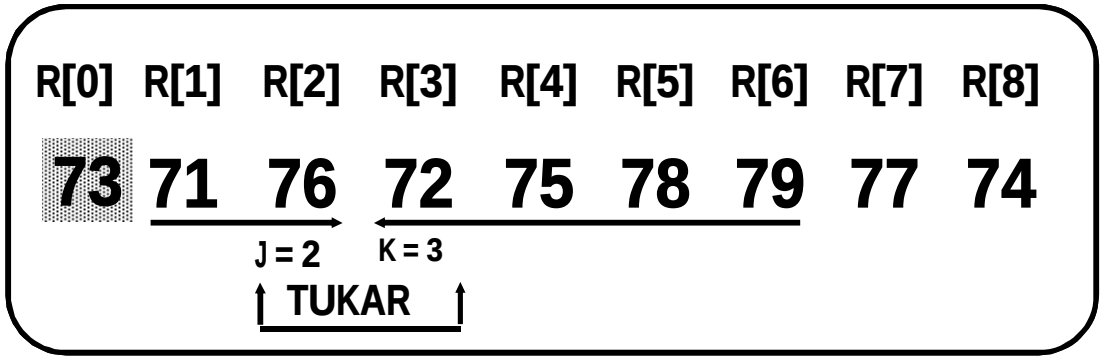
Algoritma :

```
void QuickSort(int Kiri, int Kanan)
{
    if(Kiri < Kanan){
        //Atur elemen-elemen  R[Kiri],...,R[Kanan] sehingga
        //menghasilkan urutan baru sbb.:
        R[Kiri],...,R[J-1] < R[J] dan  R[J+1],...,R[Kanan] > R[J].
        QuickSort(Kiri, J-1);
        QuickSort(J+1, Kanan);
    }
}
```

Quick Sort

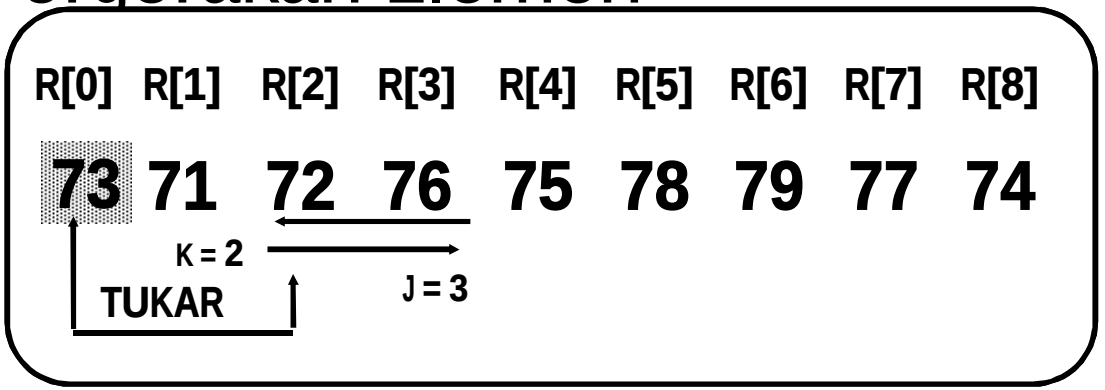
- Pola Pergerakan Elemen

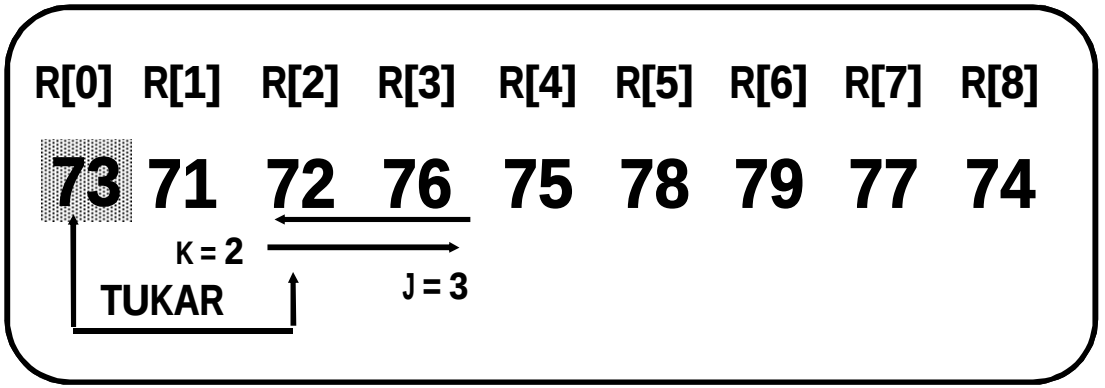




Quick Sort

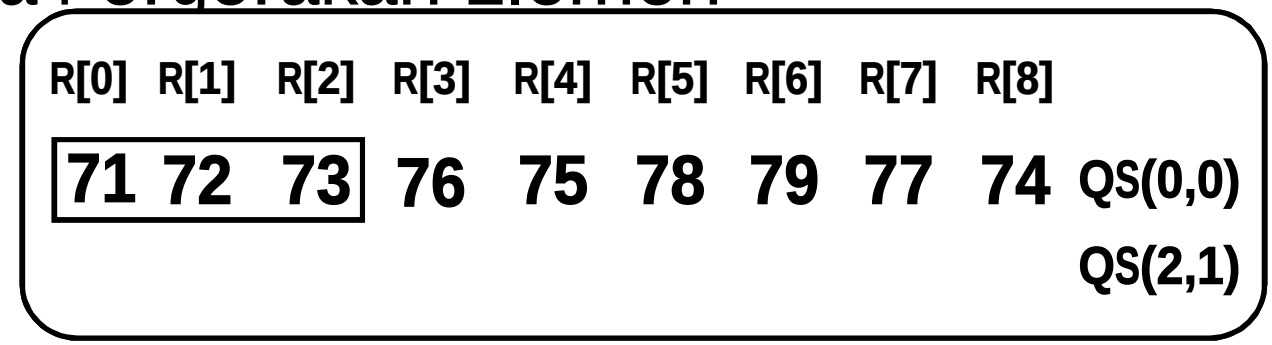
- Pola Pergerakan Elemen

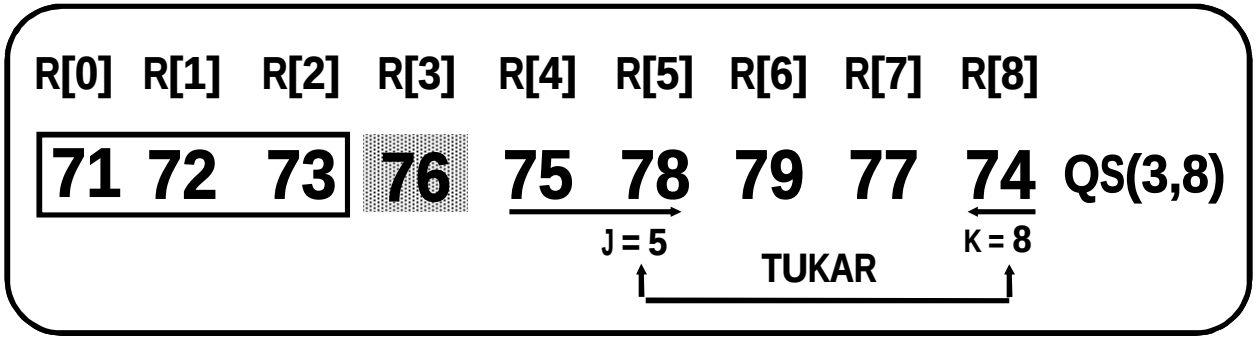




Quick Sort

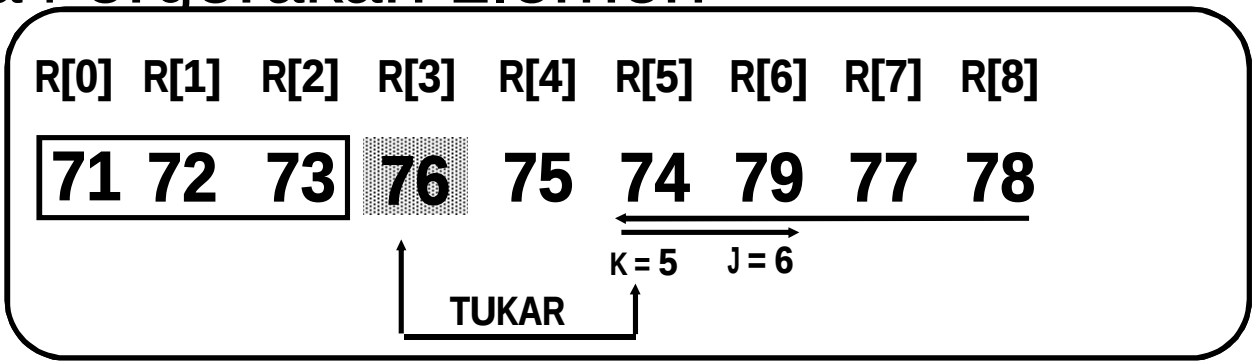
- Pola Pergerakan Elemen

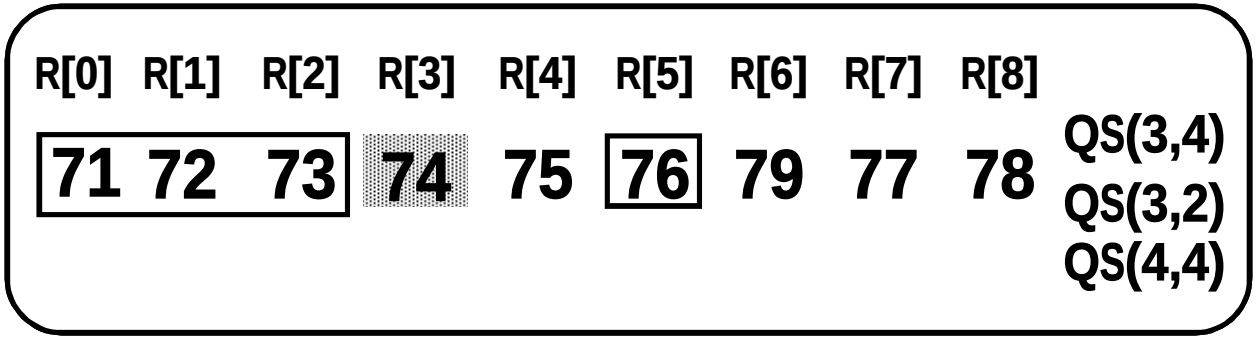




Quick Sort

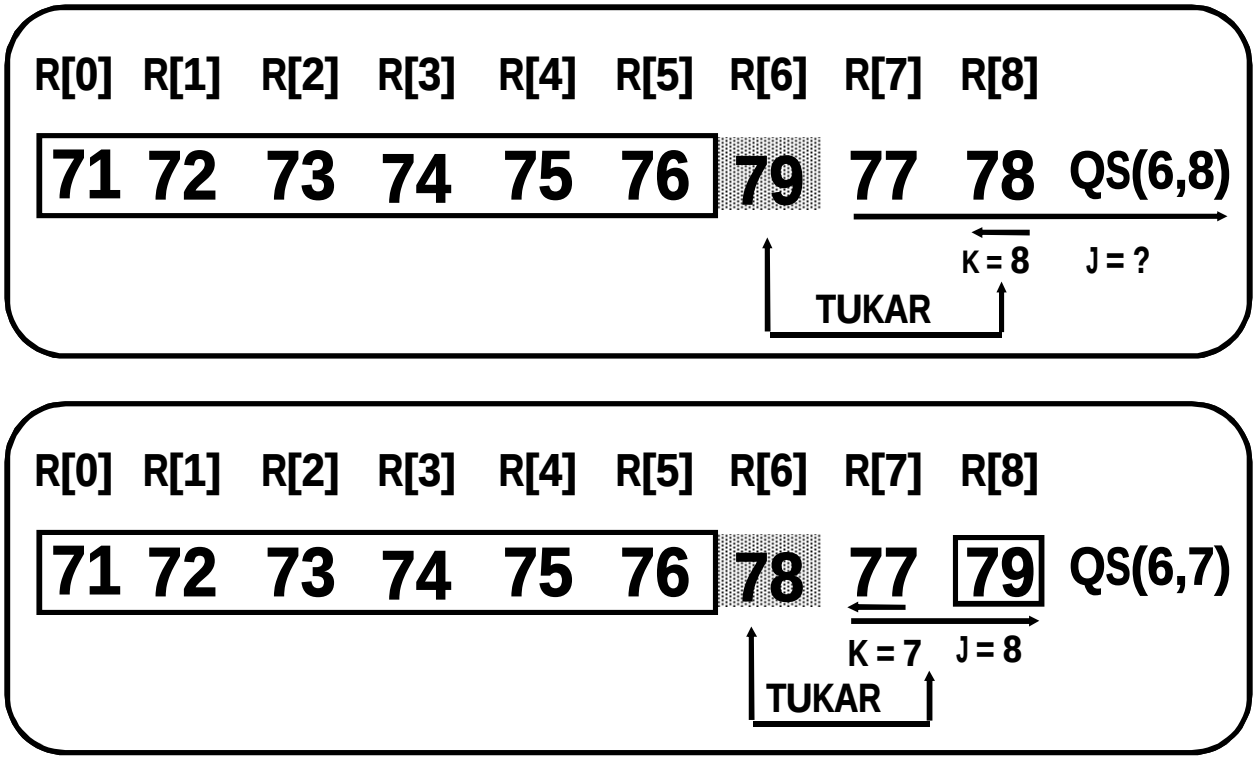
- Pola Pergerakan Elemen





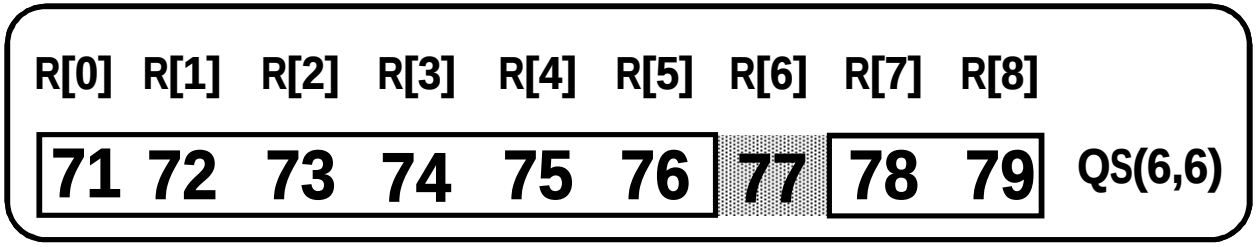
Quick Sort

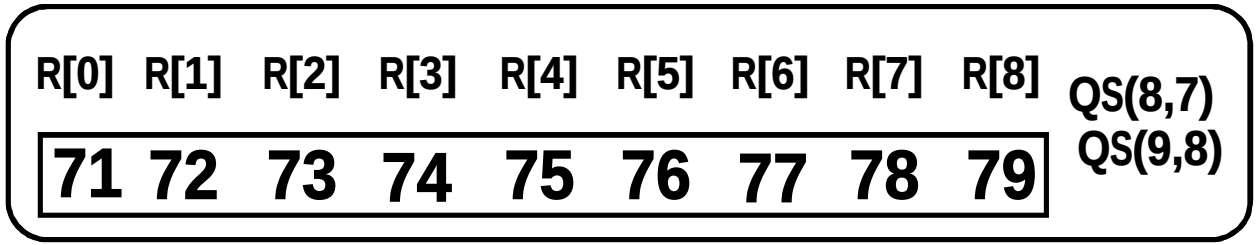
- Pola Pergerakan Elemen



Quick Sort

- Pola Pergerakan Elemen

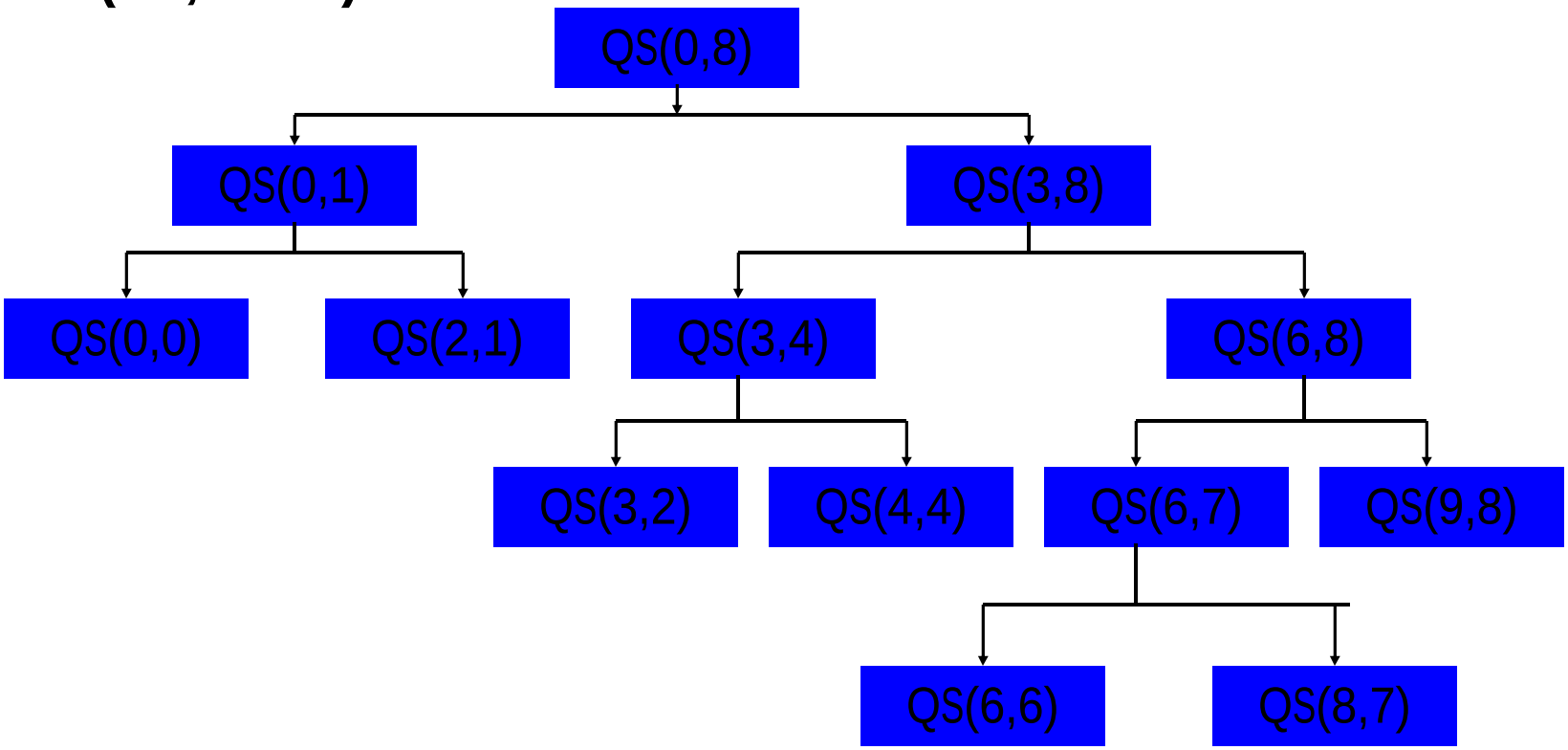




Quick Sort

- **Skema Pemanggilan QS**

  (Kiri, Kanan) secara Rekursif



Quick Sort

- Program

```
void QuickSort(int L,int R) {
        int j,k;
        if(L < R){
            j = L;  k = R + 1;
            do {
                    do{ j=j+1;} while(Arr[j] < Arr[L]);
                    do{ k=k-1;} while(Arr[k] > Arr[L]);
                    if(j < k) Tukar(&Arr[j],&Arr[k]);
            } while(j <= k);
           Tukar(&Arr[L],&Arr[k]);
           QuickSort(L,k-1);
           QuickSort(k+1,R);
        }
}
```

Coba buat fungsi Tukar !

Latihan

- Urutkan secara ascending data berikut, dan simulasikan perubahan data pada setiap putaran, dengan algoritma :
  - Bubble sort

14 6 23 18 7 47 2 83 16 38

Latihan

- Urutkan secara ascending data berikut, dan simulasikan perubahan data pada setiap putaran, dengan algoritma :
  - Selection sort

14 6 23 18 7 47 2 83 16 38

Latihan

- Urutkan secara ascending data berikut, dan simulasikan perubahan data pada setiap putaran, dengan algoritma :
  - Insertion sort

14 6 23 18 7 47 2 83 16 38

Latihan

- Urutkan secara ascending data berikut, dengan algoritma :
  - Quick sort
  - **Kemudian Gambarkan Skema Pemanggilan QS**

14 6 23 18 7 47 2 83 16 38

Latihan

- Buat fungsi / program untuk mengurut 10 data string berikut yang disimpan dalam array of string secara ascending, dengan algoritma :
  - Bubble sort

  Data sbb :

  Ali Ani Tono Bayu Amir Ani Budi Tini Ucok Paijo

Latihan

- Buat fungsi / program untuk mengurut 10 data struct mhs yang disimpan dalam array of structure, dengan algoritma Selection sort. Data diurut berdasarkan ipk secara ascending, jika ada ipk yang sama maka diurut lagi berdasarkan nama.

```
struct mhs{
    int nim;
   float ipk;
   char nama[20];
};
```

Pertemuan 15

Sorting dan Searching

Learning Outcomes

Pada akhir pertemuan ini, diharapkan mahasiswa akan mampu :

- Mendemonstrasikan simulasi algoritma sorting dan searching

Outline Materi

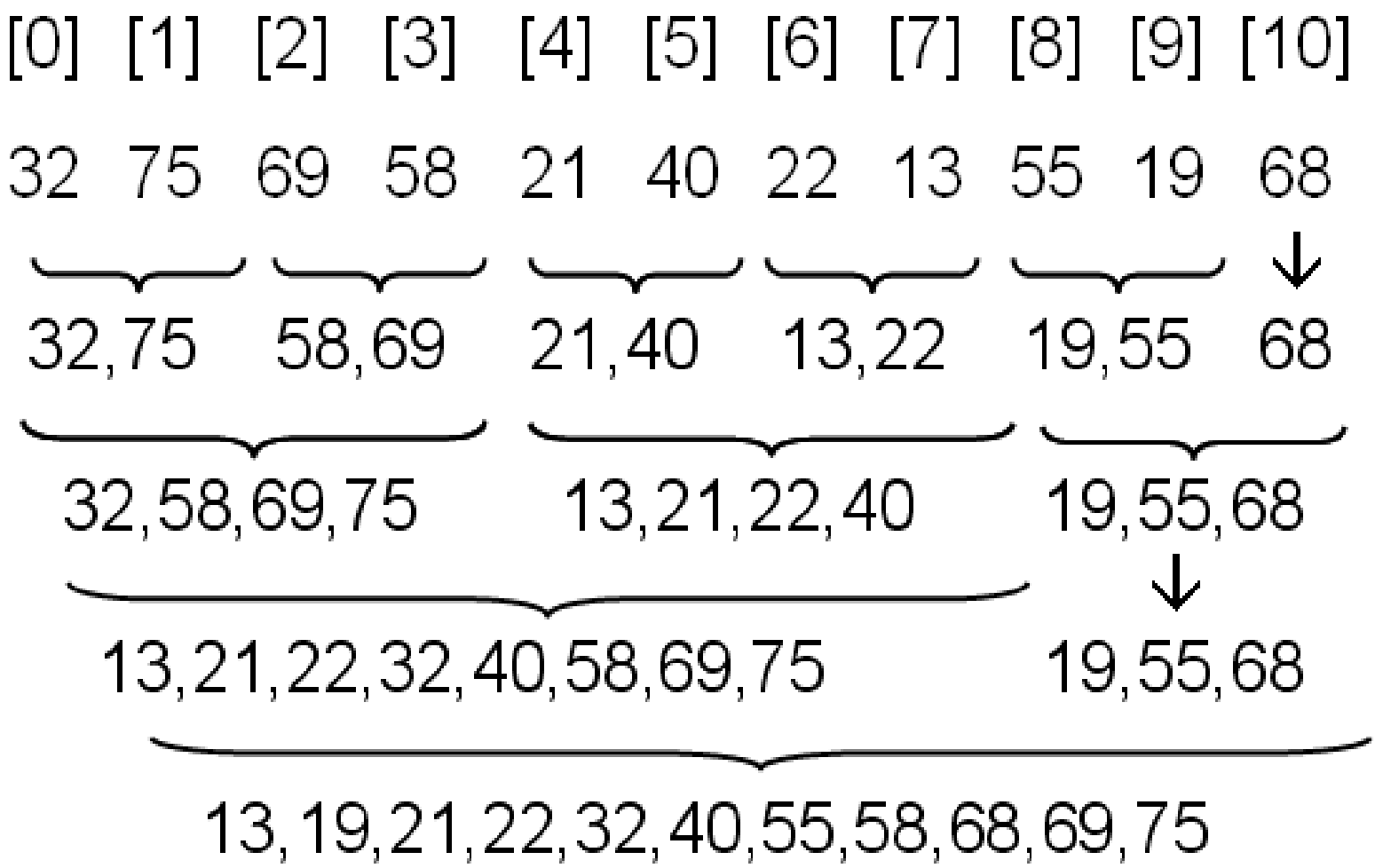
Sorting dan Searching

- Merge Sort
- Sequential search
- Binary search
- Interpolation search

Mergesort

- **Mergesort**
  - Menggabungkan 2 kumpulan data (yang masing-masing kumpulan telah terurut) menjadi satu kumpulan data yang terurut.
  - Kedua data tersebut dapat berupa *array* atau *file*.
  - Kompleksitas algoritma O(n log n) pada *worst-case*, tapi memerlukan *memory* yang lebih banyak jika data input adalah *array*

Mergesort


[0] [1] [2] [3] [4] [5] [6] [7] [8] [9] [10]
32 75 69 58 21 40 22 13 55 19 68
32,75 58,69 21,40 13,22 19,55 68
32,58,69,75 13,21,22,40 19,55,68
13,21,22,32,40,58,69,75 19,55,68
13,19,21,22,32,40,55,58,68,69,75

Mergesort

```
void mergesort(int *x, int n) {
   int *r, i, j, k, l1, l2, u1, u2, size;
   r = (int *) malloc (n * sizeof(int));
   size = 1;
   while(size<n) {
      l1= 0; k= 0;
      while (l1 + size < n) {
         l2= l1+size;
         u1= l2-1;
         u2= (l2+size-1<n)?l2+size-1: n-1;
         for(i=l1, j=l2; i<=u1 && j<=u2; k++)
            r[k]=(x[i]<=x[j])? x[i++]: x[j++];
         while (i<=u1) r[k++]= x[i++];
         while (j<=u2) r[k++] = x[j++];
         l1= u2+1;
      }
      for(i=l1; k<n; i++) r[k++]= x[i];
      for(i=0; i<n; i++) x[i]= r[i];
      size *=2;
   }
   free(r);
}
```

Searching

- *Searching* adalah proses mendapatkan (*retrieve*) informasi berdasarkan kunci tertentu dari sejumlah informasi yang telah disimpan
- ***single match*** **vs** ***multiple matches***
  - Siapa nama mahasiswa dengan NIM 0800123456 ? *single match*
  - Siapa saja yang mendapat nilai algoritma $\geq$ 85? *multiple matches*

Searching

- **Kunci** (key) digunakan untuk melakukan pencarian record yang diinginkan didalam suatu list.
- Kunci harus **unik**, artinya tidak boleh ada kunci yang sama dalam satu list.
- Kunci yang merupakan bagian dari record, disebut : **Kunci - Intern** (Internal-Key). Kunci yang tidak merupakan bagian dari record, tetapi berkaitan dengan record, disebut **Kunci - Luar** (External-Key), biasanya berupa File Indeks.
- **Contoh** : Satu record data mahasiswa terdiri dari (Nama, NIM, Jenis-Kelamin, Alamat, Tempat, Tanggal Lahir).
- NIM digunakan sebagai kunci karena NIM merupakan attribut yang bersifat unik, tidak ada yang sama.

Teknik Sequential

- Merupakan teknik yang sederhana dan langsung, dapat digunakan pada struktur data array, dan data tidak perlu urut.
- **Algoritma :**
  1. n banyak record array x
  2. Untuk setiap x[i], $0 \leq i \leq n-1$, uji apakah x[i] = kunci.
  3. Jika x[i] = kunci maka data yang dicari ketemu di indeks = i. Selesai.
  4. Jika x[i] $\neq$ kunci maka lanjutkan pencarian hingga data terakhir yaitu i = n-1.
  5. Jika i= n-1 dan x[i] $\neq$ kunci berarti data yang dicari tidak ada dan set indeks = -1. Selesai.

Teknik Sequential

/** Fungsi SequentialSearch untuk mencari data Key pada array X dgn jumlah elemen N, Return Index Array jika data ada, dan return –1 jika data Key tidak Ada **/

```
int SequentialSearch( int *X, int Key, int N)
{
        int i;
        for(i=0; i < N ; i++)  if(Key==X[i])  return(i);
        return(-1);
}
```

Teknik Binary

- Teknik ini hanya dapat dilakukan pada list yang telah terurut dan berada pada struktur array.
- Biasa digunakan untuk list yang besar ukurannya.
- Pencarian data dimulai dari pertengahan data yang telah terurut
- Jika kunci pencarian lebih kecil daripada kunci posisi tengah maka kurangi lingkup pencarian pada separuh data yang pertama (*first half*)

Teknik Binary

- **Algoritma :**
  1. n banyak record array.
  2. Kiri=0, kanan= n-1.
  3. Mid =(int) (kiri+kanan)/2
  4. Jika x[mid]=kunci maka index = mid. Selesai.
  5. Jika x[mid]<kunci maka kanan=mid-1.
  6. Jika x[mid]>kunci makan kiri = mid+1.
  7. Jika kiri $\leq$ kanan dan x[mid] $\neq$ kunci, maka ulangi 3.
  8. Jika x[mid] $\neq$ kunci maka inde x = -1. Selesai.

Teknik Binary

*/** Fungsi BinarySearch untuk mencari data Key pad array X dgn jumlah elemen N, Return Index dari Array jika data ada, dan return –1 jika data Key tidak ada **/*

```
int binarySearch(int *x, int key, int n) {
        int mid;
        int kiri=0, kanan = n-1;
        while(kiri<=kanan) {
                mid = (kiri+kanan) / 2;
                if(key==x[mid]) return(mid);
                else if(key<x[mid]) kanan = mid-1;
                     else kiri = mid+1;
        }
        return(-1);
}
```

Teknik Binary

Data di-sort berdasarkan NIM

	NIM	Nama	IPK
[0]	0700400167	Sablin Yusuf	3.37
[1]	0700400218	Tasya Turino	3.15
[2]	0700400243	Lee Shen Long	3.25
[3]	0700400303	James Lawalata	2.94
[4]	0700400491	Maruli Sinaga	3.65
[5]	0700400539	Fadli Dahlan	2.78
[6]	0700400838	Cindi Anastasia	2.94
[7]	0700400950	Ike Moniaga	2.80

Teknik Binary

Kunci pencarian? 0700400539

[0]	0700400167	←Lo	
[1]	0700400218		
[2]	0700400243		
[3]	0700400303	←Mid	
[4]	0700400491		←Lo
[5]	0700400539		←Mid
[6]	0700400838		
[7]	0700400950	←Hi	←Hi

Ditemukan pada indeks [5] ⇨ Fadli Dahlan, 2.78

Teknik Binary

Kunci pencarian? 0700400500

[0]	0700400167	←Lo			
[1]	0700400218				
[2]	0700400243				
[3]	0700400303	←Mid			
[4]	0700400491		←Lo	←Lo,Hi,Mid	←Hi,Mid
[5]	0700400539		←Mid		←Lo
[6]	0700400838				
[7]	0700400950	←Hi	←Hi		

NIM 0700400500 tidak ada pada data

Teknik Interpolasi

- Sama seperti binary search, teknik ini hanya dapat dilakukan pada list yang telah terurut dan berada pada struktur array.
- Disebut interpolasi karena data yang dicari diperkirakan ada di dalam list.
- Menggunakan rumus umum :

$$P = \frac{\text{Kunci - k[min]}}{\text{k[max] - k[min]}}$$

$$\text{Posisi} = \text{Round}(P*(\text{max-min})) + \text{min}$$

Teknik Interpolasi

- Misalkan Array A terdiri dari 100 elemen (index 0 s/d 99) dengan kunci min = 200, max = 980 dan kunci target = 743.
- Rumus interpolasi digunakan untuk menentukan posisi target sbb:

  P= (743 - 200) / (980 - 200) = 543/780=0.69
- Poasisi taksiran adalah

  Round(0. 69 * 99) + 0 = 69
- Uji apakah A[69] = 743 (kunci)? Bila Kunci > k[69]

  Maka : k [min] = k[69+1]

  sedangkan k[max] tetap
- Bila Kunci < k[69]

  Maka : k [min] = tetap

  sedangkan k[max] = k[69-1].

Teknik Interpolasi

- Program

```
int interpolationSearch(int* x, int key, int n) {
    int mid, min = 0, max = n -1;
    while(x[min] < key && x[max] >= key) {
        mid = min + ((key-x[min]) * (max-min)) / (x[max] - x[min]);
        if(x[mid] < key) min = mid + 1;
        else if(x[mid] > key) max = mid - 1;
            else return mid;
    }
    if (x[min] == key) return min;
    else return -1; // not found
}
```

Teknik Interpolasi

	Kode	Judul buku	Penulis
[0]	025	The C++ Programming	Bjarne Stroustrup
[1]	034	Mastering Delphi 6	Marco Cantu
[2]	041	Professional C#	Simon Robinson
[3]	056	Pure Corba	Fintan Bolton
[4]	063	Advanced JSP	David Geary
[5]	072	Duration Calculus	Zhou Cao Zhen
[6]	088	Temporal Logic	Zohar Manna
[7]	096	Mythical Man-Month	F.P. Brooks

Kunci pencarian? 088
low = 0, high = 7
Posisi = (088-025)/(096-025) x (7-0)+0= 6
Kode[6]== kunci ? ⇨ ya
⇨ Temporal Logic, Zohar Manna

Kunci pencarian? 060
low = 0, high = 7
Posisi = (060-025)/(096-025) x (7-0)+0= 3
Kode[3]== kunci ? ⇨ tidak
low = 4, high = 7 ⇨ kode tidak ada dalam data

Latihan

- Tulis program untuk menggabungkan 2 *file* data yang masih acak. Masing-masing *file* data di-s*ort* dengan menggunakan *quicksort* dan ditulis kedalam dua *file* perantara, lalu kedua *file* perantara ini digabung dengan *mergesort* ke *file* lain.

Latihan

- **Ilustrasi Pencarian Biner pada daftar mahasiswa dengan kunci = NIM, dan target :**
  - **178**
  - **65**
  - **424**
- **Bandingkan banyaknya langkah jika pencarian diatas menggunakan teknik sekuensial.**

Indeks	Nama	NIM
0	Ahmad	34
1	Zamzami	65
2	Ursula	112
3	Hamdan	116
4	Budiman	124
5	Rahayu	176
6	Usman	178
7	Fifi	187
8	Alex	190
9	Lukman	208
10	Widodo	214
11	Tiur	268
12	Halim	333
13	Rumokoy	424

Latihan

- **Ilustrasi Pencarian interpolasi pada daftar mahasiswa dengan kunci = NIM, dan target :**
  - **178**
  - **65**
  - **424**

Indeks	Nama	NIM
0	Ahmad	34
1	Zamzami	65
2	Ursula	112
3	Hamdan	116
4	Budiman	124
5	Rahayu	176
6	Usman	178
7	Fifi	187
8	Alex	190
9	Lukman	208
10	Widodo	214
11	Tiur	268
12	Halim	333
13	Rumokoy	424

Latihan

- Apa kekurangan *interpolation search* dibandingkan *binary search*?

Review

Review :

Materi pertemuan 9 - 15