Analisis *Top Down input* & *Output* Runtunan

Pertemuan 3

Overview

- Deskripsi
- Tujuan Instruksional
- Referensi
- Overview *Library Header*
- Analisa *Top Down*
- *Input* & *Output*
- *Runtunan*

Deskripsi

Pada pertemuan ini akan dipelajari mengenai Analisa *Top Down, input* & *Output* dalam Bahasa C/C++, runtunan dan pemilihan

Tujuan Instruksional

Mahasiswa diharapkan dapat :

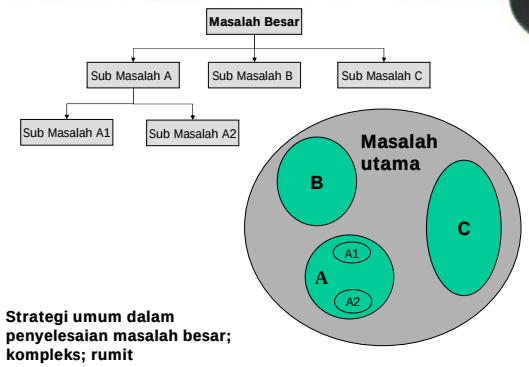
- Menjelaskan analisis *Top-Down*
- Menjelaskan proses *input* dan *output*, serta cara penulisannya dalam program
- Membedakan proses *input* dan *output*
- Menjelaskan proses runtunan(*equence*) dan pemilihan*(selection)*
- Menggunakan analisis *Top-Down*
- Menggunakan proses *input* dan *output*
- Menggunakan proses runtunan(*equence*) dan pemilihan*(selection*

Referensi

1. **Deitel, H.M. and Deitel, P.J., "C++ How to Program, 2nd Edition", Prentice Hall, 1994 (Bab 3 dan 12)**
2. **Deitel, H.M. and Deitel, P.J., "C How to Program, 4nd Edition", Prentice Hall, 2004 (bab 5,8,9 dan 21)**
3. **Herianto,"Presentasi Pemrograman Terstruktur.ppt",2004**
4. **Dan sumber lain**

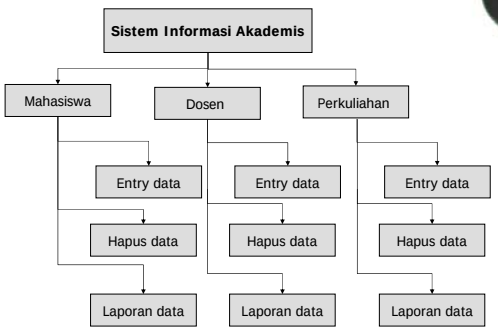
Analisis *Top Down*

Top Down

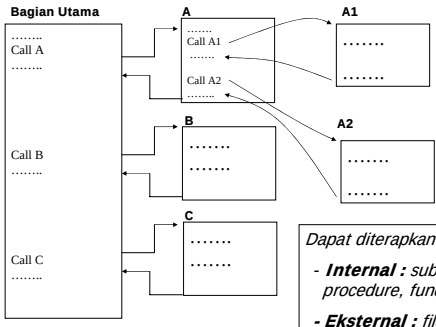


Strategi umum dalam penyelesaian masalah besar; kompleks; rumit

Contoh *Top Down*



Implementasi : Metode Modular



Dapat diterapkan secara :

- ***Internal :*** *sub program, procedure, function*
- ***Eksternal :*** *file unit, header, modul*

Input & Output

I/O C-style

C-style IO is an acquired taste. Learn to like it.

Basic functions:

- o printf, scanf, fprintf, fscanf, sprintf, sscanf, etc.
- o gets, puts, getc, putc, getchar
- o read, write, fread, fwrite

We will cover the basics of the "formated" family of functions (printf, scanf, etc).

printf

*printf(char *format_string, ...);*
fprintf(FILE, char *format_string, ...);*
snprintf(char buf, size_t n, char *format_string, ...);*

- o In C, all devices are treated like files
- o Three standard files are:
  - `stdin` Often the keyboard
  - `stdout` Often the text console
  - `stderr` Often the text console
- o `printf(....)` is `fprintf(stdout, ....)`
- o The format string is a pattern for the output; it describes how to display the arguments to printf.
- o Snprintf write to the string "buf". The variable n specifies the size of the buffer.
- o printf returns the number of characters written

format string

- Format strings are normal strings with embedded “conversion specifications” which are placeholders for arguments
- Conversion specifications are a ‘%’ and a letter with an optional set of arguments in between the ‘%’ and letter.
- To print a ‘%’, you need to write ‘%%’

Example:

```
printf(“Here is a number: %d\n”, 10);
```

%d is the conversion specification for signed integers.

Conversion Specifications

Converion specifications tell how to translate a data value into a string

Conversion Specifications:
- %d, %i -- signed integer
- %u -- unsigned integer
- %f -- floating point number
- %c -- character
- %s -- string
- %x -- hexadecimal value
- %p -- pointer

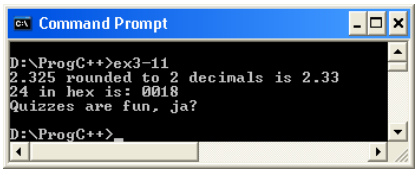
Options:
- l -- long (32-bit value)
- ll -- long long (64-bit value)
- *n* -- field width of *n* digits
- .*n* -- precision of *n* digits
- 0 -- fill unused field with 0s

There are many more! Read man pages, or Google it.

printf quiz!

Figure out the output of the following:

- printf(“%.3f rounded to 2 decimals is %.2f\n”, 2.325, 2.325);
- printf(“%d in hex is: %04x\n”, 24, 24);
- printf(“Quizzes are fun, ja?\n”);



scanf

*scanf(char *format_string, ...);*

fscanf(FILE, char *format_string, ...);*

sscanf(char, char *format_string, ...);*

- `scanf(....)` is `fscanf(stdin, ....)`
- All arguments ot scanf must be pointers (or arrays)
- scanf does almost no size checks. It is easy to get a buffer overflow here. Make sure you use a field length specifier with the %s conversion specifer!!!
- scanf returns the number of *items* read.

scanf Examples

```
int items_read;
```

Read a number:

```
int num;
items_read = scanf(“%d”, &num);
```

always check the return value of scanf

Read a character:

```
char ch;
items_read = scanf(“%c”, &ch);
```

Read a string of max length, 79 chars:

```
char buf[80];
buf[79]=‘\0’;   // Ensure a terminating NULL.
items_read = scanf(“%79s”, buf);
```

Read number after pattern of “a:<num>”:

```
int num;
items_read = scanf(“a:%d”, &num);
```


I/O C++-style

C++-style IO is easier for simple stuff

Basic classes:
- iostream (cout, cin, cerr)
- ostringstream, istringstream

```
cout << “Hello World!” << endll;
cout << “Boo! “ << 10 << ‘c’ << endl;
cerr << “Here’s the error stream” << endl;

int n;
cin >> n;

char ch;
cin >> ch;
```

I/O C++-style continued...

...but harder for complex stuff

```
printf("%.3f rounded to 2 decimals is
%.2f\n", 2.325, 2.325);
```

...becomes...

```
cout << setprecision(3) << 2.325
     << " rounded to 2 decimals is "
     << setprecision(2) << 2.3.25
     << endl;
```


C and C++ I/O compared

C-style I/O:

- No type safety. What happens with `printf("%d", 'c');`?
- Conversion specifications have a high learning curve.
- Almost all the state of the I/O is contained in the function call.

C++ style I/O:

- Manipulators are very verbose/annoying
- Global state gets changed. When you do "`cout << 2.4555`", what precision are you set at? You don't know. It's worse with threads.
- You get more customizability since C++ I/O is classed based.

NEVER mix C and C++ I/O...until you know what `ios::sync_with_stdio()` does.

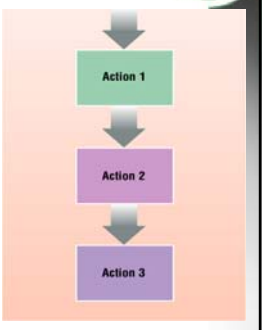
Struktur Dasar Proses

- Terdapat 3 struktur dasar proses dalam algoritma/program, yaitu:
  - Runtunan (*Sequence*)
  - Pemilihan/pengambilan keputusan (*Selection*)
  - PEngulangan (*Looping*)

Sequence/Runtunan

Simple Sequence

- Aksi-aksi di dalam algoritma/program dilaksanakan oleh pemroses sesuai dengan urutan penulisannya
- Statement diproses dalam suatu urutan yang telah ditentukan (*Top-Down*)
- Statement diproses per baris secara berurutan tanpa ada yang terlewatkan (kecuali ada statement Goto)
- Sebuah baris statemen akan diproses/dieksekusi setelah baris statemen sebelumnya selesai dieksekusi
- Tidak memungkinkan terjadinya *parallel processing* (eksekusi banyak baris secara bersamaan)



Sequence

- Instruksi dikerjakan secara berurutan.
  - dari atas ke bawah

step	Instruksi	
	Algoritma	Program
1	INPUT *jmlBrg*, *hrgSat*	scanf("%d",&*jmlBrg*); *scanf("%f",&hrgSat);*
2	*harga* ← *jmlBrg* * *hrgSat*	*harga* = *jmlBrg* * *hrgSat;*
3	OUTPUT *harga*	printf("%.0f\n",*harga);*

Contoh Sequence (1)

- Algoritma/Program akan dikerjakan berdasarkan input-an:

step	Instruksi	
	Algoritma	Program
1	INPUT *jmlBrg*, *hrgSat*	scanf("%d",&*jmlBrg*); *scanf("%f",&hrgSat);*
2	*harga* ← *jmlBrg* * *hrgSat*	*harga* = *jmlBrg* * *hrgSat;*
3	OUTPUT *harga*	printf("%.0f\n",*harga);*

- Proses yang terjadi pada saat algoritma/program dijalankan:

step	Variabel			Output
	jmlBrg	*hrgSat*	*harga*	

Contoh Sequence (2)

- Mulai dengan langkah-1:
  - Nilai yang dimasukan 2, 1500

step	Instruksi	
	Algoritma	Program
1	INPUT *jmlBrg*, *hrgSat*	scanf("%d",&*jmlBrg*); *scanf("%f",&hrgSat);*
2	*harga* ← *jmlBrg* * *hrgSat*	*harga* = *jmlBrg* * *hrgSat;*
3	OUTPUT *harga*	printf("%.0f\n",*harga);*

step	Variabel			Output
	jmlBrg	*hrgSat*	*harga*	
1	2	1500		

Contoh Sequence (3)

- Langkah-2:
  - Hitung perkalian, simpan hasilnya di variabel harga

step	Instruksi	
	Algoritma	Program
1	INPUT *jmlBrg*, *hrgSat*	scanf("%d",&*jmlBrg*); *scanf("%f",&hrgSat);*
2	*harga* ← *jmlBrg* * *hrgSat*	*harga* = *jmlBrg* * *hrgSat;*
3	OUTPUT *harga*	printf("%.0f\n",*harga);*

step	Variabel			Output
	jmlBrg	*hrgSat*	*harga*	
1	2	1500		
2	2	1500	3000	

Contoh Sequence (4)

- Langkah-3:
  - Tampilkan isi variabel harga

step	Instruksi	
	Algoritma	Program
1	INPUT *jmlBrg*, *hrgSat*	scanf("%d",&*jmlBrg*); *scanf("%f",&hrgSat);*
2	*harga* ← *jmlBrg* * *hrgSat*	*harga* = *jmlBrg* * *hrgSat;*
3	OUTPUT *harga*	printf("%.0f\n",*harga);*

step	Variabel			Output
	jmlBrg	*hrgSat*	*harga*	
1	2	1500		
2	2	1500	3000	
3	2	1500	3000	3000

Pemilihan (*Selection*)

Tujuan

- Sebuah aksi dikerjakan jika kondisi tertentu dipenuhi. Dalam hal ini kita memerlukan konsep untuk membandingkan suatu keadaan/kondisi dari keadaan/kondisi lain sehingga kita mendapatkan pilihan yang terbaik
- Mengontrol jalannya algoritma/program agar dapat memilih salah satu dari sekian banyak pilihan yang ada
- Memilih satu atau lebih statement yang akan diproses berdasarkan kondisi yang telah ditetapkan
- Pemilihan solusi berdasarkan kriteria tertentu yang telah ditetapkan sebelumnya, untuk mendapatkan hasil yang optimal
- Jenis :
  - *One way selection*
  - *Two way selection*
  - *Multi ways selection*

Struktur Dasar dalam Algoritma(1)

1. IF <kondisi>
   THEN
   <Aksi>
   END IF

Contoh :

IF 10 > 5
THEN
Output ("10 lebih besar")
END IF

2. IF <kondisi>
   THEN
   <Aksi1>
   ELSE
   <Aksi2>
   END IF

Contoh :

IF X > Y
THEN
Output ("X lebih besar")
ELSE
Output ("X lebih kecil")
END IF

Struktur Dasar dalam Program (1)

One Ways Selection

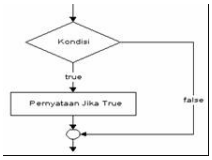
o Syntax :

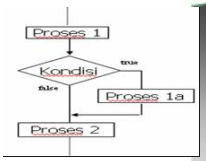
```
if (expression)
{
  statement ke-1;
  ....
  statement ke-n;
}
```

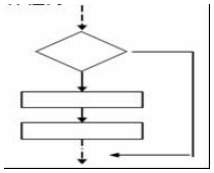
o Semua statement dalam blok if di atas akan dijalankan jika ekspression bernilai True

Struktur Dasar dalam Program (2)

One Ways Selection







Struktur Dasar dalam Program (3)

Two Ways Selection

o Syntax :

```
if (expression)
{
   statement a ke-1;
   ....
   statement a ke-n;
}
else
{
   statement b ke-1;
   ....
   statement b ke-n;
}
```

o Semua Statement a dalam blok if di atas akan dijalankan jika ekspression bernilai **True**

o Semua Statement b dalam blok if di atas akan dijalankan jika ekspression bernilai **False**

Struktur Dasar dalam Program (4)

Two Ways Selection

o Syntax di atas dapat disederhanakan dengan menggunakan *ternary* operator. Syntaxnya adalah sebagai berikut :

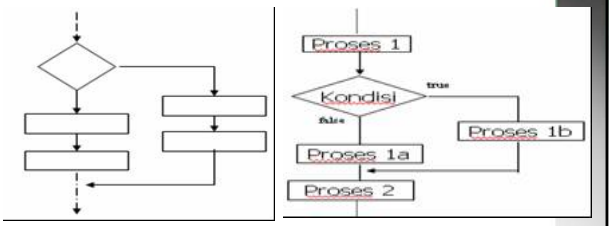
(expression**) ?** statement a **:** statement b;

o Statement a dalam blok di atas akan dijalankan jika ekspression bernilai **True**

o Statement b dalam blok di atas akan dijalankan jika ekspression bernilai **False**

Struktur Dasar dalam Program (5)

Two Ways Selection



Struktur Dasar dalam Program (6)

Multi Ways Selection

o Syntax :

```
if (expression ke-1)
{
  statement a ke-1; ....;
  statement a ke-n;
}
else if (expression ke-2)
   {
     statement b ke-1;....;
     statement b ke-n;
   }
   ....
else if (expression ke-n)
   {
   statement y ke-1;....;
   statement y ke-n;
   }
   else
   {
   statement z ke-1;....;
   statement z ke-n;
   }
```

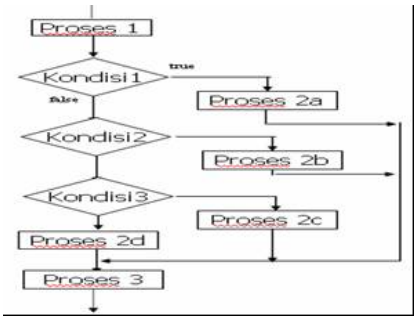

Struktur Dasar dalam Program (7)

Multi Ways Selection

- o Semua Statement a dalam blok if di atas akan dijalankan jika expression ke-1 bernilai True
- o Jika expression ke-1 bernilai False, maka expression ke-2 akan dicek. Jika expression ke-2 bernilai True Semua Statement b dalam blok if di atas akan dijalankan.
- o Dan seterusnya

Struktur Dasar dalam Program (8)

Multi Ways Selection



Struktur Dasar dalam Algoritma (2)

```
• DEPEND ON <Variabel>
        Kondisi 1 : <Aksi  1>
        Kondisi 2 : <Aksi 2>
        ...
        Kondisi n : <Aksi n>
  END DEPEND ON
```

Contoh :

```
DEPEND ON x, y
        x < y : Output ("nilai x lebih kecil dari y")
        x > y : Output ("nilai x lebih besar dari y")
        x = y : Output ("nilai x sama dengan y")
 END DEPEND ON
```


Struktur Dasar dalam Program (9)

Statement SWITCH

o Tujuan :

- Menyederhanakan bentuk multi ways selection
- Mengurangi kemungkinan error karena tidak perlu menulis syntax switch lagi

o Syntax :

```
switch (expression)
{
  case const-expression ke-1 :
    {statement a ke-1;....; statement a ke-n; break;}
  case const-expression ke-2 :
    {statement b ke-1;....; statement b ke-n; break;}
  ...
  case const-expression ke-n :
    {statement y ke-1;....; statement y ke-n; break;}
  default :
    {statement z ke-1;....;statement z ke-n; break;}
}
```

Struktur Dasar dalam Program (10)

Statement SWITCH