Algoritma Pemrograman 2B (Pemrograman C++)

Materi 1

PENGENALAN C++

Pengetahuan Pemrograman (1)

- Pemrograman tidak hanya coding.
- Terutama itu berarti penataan solusi untuk masalah dan kemudian menyempurnakan langkah demi langkah solusi.
- Ketika disempurnakan ke tingkat yang cukup dalam, Anda telah menciptakan algoritma.
- Maka saatnya untuk menerjemahkan setiap langkah dari algoritma ke kode program.

Pengetahuan Pemrograman (2)

- Misalkan ada masalah yang perlu dipecahkan. Kemudian dimulai dengan menulis urutan operasi yang perlu dilakukan untuk memecahkan masalah.
- Mulai dari awal lagi dan fokus pada satu operasi pada satu waktu dan mencari tahu apakah operasi perlu disempurnakan untuk langkah-langkah yang lebih rinci. Kemudian Anda melanjutkan ke tingkat berikutnya dan menyempurnakan lebih lanjut.

Pengetahuan Pemrograman (3)

- Proses perbaikan berlangsung sampai Anda tiba di tingkat yang cukup dalam untuk memulai coding.
- Menciptakan algoritma untuk memecahkan masalah sebagian besar tugas dari pekerjaan pemrograman.
- Banyak orang melakukan kesalahan dengan mulai membuat kode, yang membuat fokus pada detail kode dan melupakan masalah yang sebenarnya harus dipecahkan.

Pengetahuan Pemrograman (4)

- Yang menghasilkan kode yang tidak terstruktur dan tidak efisien yang sulit untuk dipahami dan dipelihara.
- Itulah mengapa perlu ditekankan bahwa perlu melakukan latihan struktur logika pemikiran dan membangun sebuah algoritma yang baik sebelum mulai membuat kode.

Sejarah C (1)

- Perkembangan bahasa C erat kaitannya dengan perkembangan sistem operasi UNIX, dimana keduanya dikembangkan di Laboratorium AT&T Bell (USA).
- Pertama kali bahasa C diimplementasikan pada komputer DEC PDP-11 yang menggunakan sistem operasi UNIX oleh Dennis Ritchie di Laboratorium Bell tahun 1972.

Sejarah C (2)

- Banyak ide penting dari C diambil dari bahasa BCPL yang dikembangkan oleh Martin Richards melalui bahasa B yang ditulis Ken Thompson. BCPL dan B tidak memiliki tipe data sedangkan C memiliki beragam tipe data.
- Pada tahun 1978 Dennis Ritchie dan Brian W Kernighan mengeluarkan buku THE C PROGRAMMING LANGUAGE, buku inilah yang dijadikan acuan dari pembuatan berbagai versi bahasa C yang ada.

Sejarah C(3)

- Pada tahun 1983 ANSI (American Nasional Standart Institute) membuat standarisasi bahasa C yang kemudian dijadikan sebagai referensi dari berbagai bahasa C yang beredar saat ini. C yang dihasilkan disebut C standard ANSI atau ANSI C. Tahun 1987 Borland menciptakan sebuah compiler C yang dikenal dengan nama TURBO C.
- Dalam beberapa literatur bahasa C digolongkan sebagai bahasa tingkat menengah (*Medium Level Language*). Penggolongan ini tidak berarti bahwa bahasa C lebih sulit dibandingkan dengan bahasa pemrogramman tingkat tinggi seperti : PASCAL, BASIC.

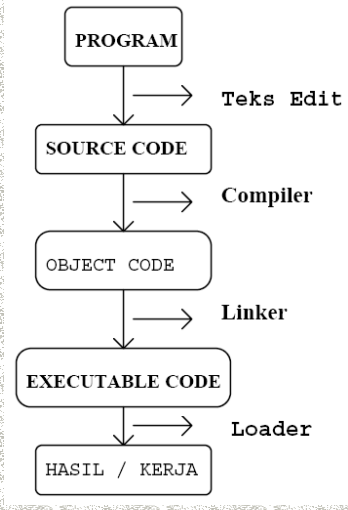
Sejarah C(4)

- Pada kenyataannya bahasa C mengkombinasikan elemen dalam bahasa tingkat tinggi dan bahasa tingkat rendah.
- Hampir semua operasi yang dapat dilakukan oleh bahasa mesin dapat dilakukan oleh C dengan penyusunan program yang lebih sederhana dan mudah

Sejarah C(5)

- Bahasa C dalam pemakaiannya memerlukan suatu translator. Jenis translator dalam bahasa C adalah interpreter dan compiler. Interpreter merupakan translator yang menterjemahkan bahasa C ke dalam bahasa mesin satu persatu. Contoh interpreter yang beredar saat ini yaitu Run/c.
- Sedangkan compiler merupakan translator yang menterjemahkan bahasa C ke dalam bahasa mesin secara keseluruhan. Contoh compiler : Turbo C, Microsoft C dan Lattice C.

- Kode Program → .C
- Hasil kompilasi →.obj
- Object code sudah berbentuk kode mesin, tapi kode ini belum bisa dimengerti oleh komputer. Agar dapat dimengerti oleh komputer maka object code bersama dengan object code yang lain serta file library (file yang berisi rutin untuk tugas tertentu) perlu dikaitkan (linking) dengan menggunakan linker. Sehingga terbentuk suatu file yang executable (program yang dapat dijalankan secara langsung dalam lingkungan sistem operasi).
- Program hasil linker ini disimpan dalam sebuah file executable dengan ciri mempunyai ekstension .EXE



Gambar 1. Proses penterjemahan bahasa C ke dalam bahasa mesin dengan menggunakan compiler

Kelebihan Bahasa C(1)

- C mempunyai banyak jenis data yang dikenal & operator untuk manipulasi data.
- C menyediakan berbagai struktur data dan pengendalian proses. Sehingga memungkinkan untuk membuat program yang terstruktur (program yang mudah dipakai dan dikembangkan).
- C mudah dipahami dibandingkan dengan bahasa mesin, karena bahasa C berorientasi pada permasalahan bukan pada mesin..

Kelebihan Bahasa C(2)

- C mempunyai kecepatan eksekusi yang mendekati kecepatan eksekusi bahasa mesin
- C memungkinkan manipulasi data dalam bentuk bit maupun byte secara efisien dan dapat memanipulasi alamat dari suatu data
- C memakai sedikit memori
- C merupakan salah satu bahasa pemrograman yang terstruktur.

Sejarah C++

- Merupakan perluasan dari C
- Ditemukan pada tahun 1980-an oleh Bjarne Stroustrup (Laboratorium Bell)
- Menyediakan kemampuan untuk bahasa pemrograman berorientasi objek
  - Objek: komponen perangkat lunak dapat digunakan kembali
  - Object-oriented programs
    - Mudah dipahami, dikoreksi dan dimodifikasi
- Bahasa Dominan di dalam industri dan akademisi

Kekurangan Bahasa C

- Banyaknya operator serta flexibilitas penulisan program yang terkadang membingungkan pemakai.
- Programmer C tingkat pemula pada umumnya kurang mengenal pointer dan tak terbiasa menggunakannya, padahal kelebihan C ada pada pointer ini.

MENGAPA C ???

- Semakin banyak program dan software yang ditulis dalam C
- C adalah bahasa yang modern, efisien, portable, powerful, flexible, friendly menghasilkan program yang kompak dan cepat
- C adalah bahasa yang mendominasi pemakaian komputer dengan sistem operasi UNIX.

MENGAPA C ???

C adalah bahasa yang portable:

- program yang ditulis pada satu sistem dapat dijalankan pada sistem lain tanpa atau dengan sedikit perubahan

C adalah bahasa yang efisien :

- menghasilkan program yang kompak/ringkas dan cepat

C adalah bahasa modern :

- mempunyai fasilitas - fasilitas kontrol yang diperlukan oleh teori – teori *computer science* maupun aplikasi

C adalah bahasa yang friendly :

- keterikatannya/ restriksinya tidak ketat

MENGAPA C ???

C adalah bahasa yang powerful dan fleksible

- UNIX sebagian besar ditulis dalam C
- C compiler ditulis dalam C
- Interpreter lain (FORTRAN, APL, Pascal, LISP, BASIC) bisa ditulis dalam C
- Digunakan baik dalam memecahkan masalah fisik maupun engineering
- Digunakan untuk animasi
- Mempunyai kontrol yang biasanya hanya dimiliki oleh bahasa assembly.

Visual C++

- Penerapan C++ dalam Microsoft
  - Includes extensions
  - Microsoft Foundation Classes (MFC)
  - Common library
    - GUI, graphics, networking, multithreading, …
    - Shared among Visual Basic, Visual C++, C

Pengenalan Program C

- Program terdiri dari koleksi satu / lebih fungsi, salah satu diantaranya harus berupa main().
- Fungsi terdiri dari sebuah header dan sebuah body/badan.
- Header berisi *preprocessor statement* seperti #include dan nama fungsi
- Badan ditandai oleh { } dan berisi sejumlah statement yang masing-masing diakhiri tanda ;

Struktur Program C

```
# include file
# define var konstan
        deklarasi fungsi
        main()
                {
                        deklarasi variabel global
                        :
                        <pernyataan - pernyataan>
                        :
                }
                nama fungsi (arg1, arg2,...)
                {
                        deklarasi variabel lokal
                        :
                        < pernyataan - pernyataan>
                        :
        }
```

Contoh Program C

```
// my first program in C++
#include <iostream>

int main()
{
  std::cout << "Hello World!";
}
```

Hello World!

Penjelasan Program (1)

- Baris 1: // my first program in C++

  Baris diawali dengan dua tanda slash (//) adalah comments oleh programmer dan tidak memiliki efek pada perilaku program. Programmer menggunakannya untuk penjelasan pendek mengenai kode program.

Penjelasan Program (2)

- Baris 2: #include <iostream>

  Sebuah tanda hash (#) adalah instruksi membaca dan ditafsirkan oleh apa yang dikenal sebagai *preprocessor*. Ini adalah baris khusus yang diterjemahkan sebelum melakukan kompilasi program. Dalam kasus ini, #include <iostream>, menginstruksikan *preprocessor* untuk menyertakan sebuah section standard kode C++, yang dikenal sebagai *header iostream*, yang mengijinkan untuk membentuk operasi input dan output, seperti menulis output program ini pada layar (Hello World).

Penjelasan Program (3)

- Line 3: Sebuah blank line.

  Baris kosong tidak memiliki efek apapun pada sebuah program. Ini hanya memudahkan untuk dibaca.

Penjelasan Program (4)

- Line 4: int main ()

  Ini merupakan deklarasi sebuah fungsi. Sebuah fungsi adalah sebuah kelompok kode yang diberikan sebuah nama: Dalam kasus ini, diberi nama "main“. Fungsi ini diawali dengan sebuah type (int), sebuah nama (main) dan pasangan parentheses (()).

  Fungsi yang diberi nama main adalah fungsi khusus dalam semua program C++ ; Ini adalah fungsi yang dipanggil ketika sebuah program dijalankan.

Penjelasan Program (5)

- Lines 5 dan 7: { dan }

  Kurung kurawal buka ({) pada baris 5 mengindikasikan awal dari definisi fungsi main, dan kurung kurawal tutup (}) pada baris 7, mengindikasikan akhir program. Segala sesuatu diantara kurung kurawal adalah badan fungsi yang mendefinisikan apa yang terjadi ketika main dipanggil.

Penjelasan Program (6)

- Line 6: std::cout << "Hello World!";

  Ini adalah C++ statement. Sebuah statement adalah sebuah ekspresi yag dapat menghasilkan beberapa efek. Ini merupakan isi sebuah program. Statement dieksekusi dalam urutan yang sama sesuai urutan di badan fungsi.
- Catatan: Setiap statemen dalam bahasa C diakhiri dengan tanda semicolon (;)

```
1  /* Fig. 2.5: fig02_05.c
2     Addition program */
3  #include <stdio.h>
4
5  int main()
6  {
7     int integer1, integer2, sum;        /* declaration */
8
9     printf( "Enter first integer\n" );  /* prompt */
10    scanf( "%d", &integer1 );           /* read an integer
11    printf( "Enter second integer\n" ); /* prompt */
12    scanf( "%d", &integer2 );           /* read an integer
13    sum = integer1 + integer2;          /* assignment of sum
14    printf( "Sum is %d\n", sum );       /* print sum */
15
16    return 0;  /* indicate that program ended successfully
17 }
```

```
Enter first integer
45
Enter second integer
72
Sum is 117
```

- 1. Initialize variables
- 2. Input
- 2.1 Sum
- 3. Print

Output Program

- Program Output

Penjelasan Program(1)

- `int integer1, integer2, sum;`
  - Deklarasi Variabel
    - Variabel: lokasi dalam memori dimana sebuah nilai dapat disimpan
  - `int` bermakna variabel dapat menangani integers (`-1`, `3`, `0`, `47`)
  - Nama Variabel (identifiers)
    - `integer1`, `integer2`, `sum`
    - Identifiers: terdiri atas huruf, digits (tidak dapat diawali dengan sebuah angka/digit) dan underscores(`_`)
    - Case sensitive
  - Deklarasi muncul sebelum executable statements
    - Jika sebuah executable statement mengacu pada dan tidak mendeklarasikan variabel ia akan mengasilkan sebuah syntax (compiler) error

Penjelasan Program (2)

- `scanf( "%d", &integer1 );`
  - Memperoleh sebuah nilai dari pemakai
    - `scanf` menggunakan standard input (umumnya keyboard)
  - `scanf` statement ini memiliki dua arguments
    - %d - mengindikasi data harus berupa decimal integer
    - `&integer1` - lokasi dalam memori untuk menyimpan variabel
    - & untuk saat ini, hanya perlu diingat untuk menyertakannya dengan nama variabel dalam `scanf` statements

Penjelasan Program (3)

- = (assignment operator)
  - Menugaskan sebuah nilai ke sebuah variabel
  - Adalah sebuah operator biner (memiliki dua operands)

```
sum = variable1 + variable2;
sum gets variable1 + variable2;
```

  - Variabel yang menerima nilai terletak di sisi kiri

Penjelasan Program (4)

- `printf( "Sum is %d\n", sum );`
  - Sama dengan `scanf`
    - %d bermakna decimal integer akan dicetak
    - `sum` menentukan integer apa yang akan dicetak
  - Kalkulasi dapat dibentuk di dalam `printf` statements

```
printf( "Sum is %d\n", integer1 +
  integer2 );
```

Konsep Memori

- Variabel
  - Nama Variabel menghubungkan ke lokasi dalam memori komputer
  - Setiap variabel memiliki nama, tipe, ukuran dan nilai
  - Kapan saja sebuah nilai baru diletakkan ke dalam sebuah variabel (melalui `scanf`, sebagai contoh), ia menggantikan (dan menghapus) nilai sebelumnya
  - Membaca variabel dari memori tidak merubah isinya

Tipe Data (1)

- Semua variabel harus didefinisikan di dalam C.
- Memiliki bentuk :
  - ✓ tipe data mendefinisikan variabel sebelum variabel digunakan
  - ✓ Definisi dari satu variabel akan memberikan tempat penyimpanan untuk variabel dan mendefinisikan tipe data yang akan ditangani dalam lokasi
  - ✓ Mempunyai bentuk → `typename variablename;`
  - ✓ Contoh : `int myInteger;`
    `char myCharacter;`

Tipe Data (2)

Tipe data dikelompokka menjadi:

- **Character types:** menrepresentasikan sebuah karakter, seperti 'A' atau '$'. Merupakan satu-byte character.

```
char myGrade;
```

Membuat variabel character yang disebut "myGrade".

Tipe Data (3)

- **Numerical integer types:** dapat menyimpan seluruh nilai, seperti 7 atau 1024. Mereka ada dalam beragam ukuran, dan dapat berupa *signed* atau *unsigned*, bergantung pada apakah mereka mendukung nilai negative atau tidak.

Tipe Data (4)

- **Floating-point types:** dapat merepresentasikan nilai real, seperti 3.14 atau 0.01, dengan tingkat presisi yang berbeda, tergantung pada 2 tipe floating-point yang digunakan.
- float (single precision) – 4 bytes
- double - 8 bytes

Tipe Data (5)

- **Boolean type:** dikenal dalam C++ sebagai bool, hanya dapat merepresentasikan satu dari dua keadaan, true atau false.

Tipe Data (6)

Data Type		Abbreviation	Size (byte)	Range
char	`char`		1	-128 ~ 127
	`unsigned char`		1	0 ~ 255
int	`int`		2 or 4	$-2^{15} \sim 2^{15}-1$ or $-2^{31} \sim 2^{31}-1$
	`unsigned int`	`unsigned`	2 or 4	0 ~ 65535 or $0 \sim 2^{32}-1$
	`short int`	`short`	2	-32768 ~ 32767
	`unsigned short int`	`unsigned short`	2	0 ~ 65535
	`long int`	`long`	4	$-2^{31} \sim 2^{31}-1$
	`unsigned long int`	`unsigned long`	4	$0 \sim 2^{32}-1$
float			4	
double			8	

Deklarasi Variabel

```
type v1,v2,v3, …, vn
```

Example:

```
int i;
int j;
float k;
char c;
short int x;
long int y;
unsigned int z;
int a1, a2, a3, a4, a5;
```

Inisialisasi Variabel (1)

- Ketika variabel dideklarasikan, mereka memiliki sebuah nilai yang tidak diketahui sampai diberikan sebuah nilai untuk pertama kali. Sebuah variabel dapat memiliki sebuah nilai tertentu saat ia dideklarasikan. Ini disebut *initialization* variabel.

Inisialisasi Variabel (2)

Ada 3 cara untuk inisialisasi variabel:

- Pertama, dikenal sebagai *c-like initialization*, terdiri dari pembubuhan sebuah tanda sama dengan diikuti dengan nilai variabel yang diinisialisasi:

  type identifier = initial_value;

  Contoh : *int* x = 0;

Inisialisasi Variabel (3)

- Metode ke dua, dikenal sebagai *constructor initialization*, apit nilai inisialisasi diantara parentheses (()):
  type identifier (initial_value);
- Contoh : *int* x (0);

Inisialisasi Variabel (4)

- Metode ke tiga, dikenal sebagai *uniform initialization*, sama seperti yang ke dua tetapi menggunakan kurung kurawal ({})
  type identifier {initial_value};
- Contoh : *int* x {0};

Inisialisasi Variabel (5)

```
1  // initialization of variables
2  #include <iostream>
3  using namespace std;
4  int main ()
5  {
6   int a=5;  int b(3); int c{2}; int result;
7   a = a + b;
8   result = a - c;
9  cout << result;
10  return 0;
11  }
```

Tipe Data String (1)

- Salah satu kekuatan bahasa C++ adalah tipe compound/gabungan.
- Sebuah contoh tipe compound adalah *string*. Variabel dari tipe ini mampu menyimpan sederetan karakter, seperti kata atau kalimat.
- Perbedaan pertama dengan tipe data fundamental yakni saat deklarasi dan menggunakan variabel dari tipe ini, program perlu menyertakan header ketika tipe didefinisikan dalam standard library (header <string>):

Tipe Data String (2)

```
1  // my first string
2  #include <iostream>
3  #include <string>
4  using namespace std;
5  int main ()
6  { string mystring;
7   mystring = "This is a string";
8   cout << mystring;
9  return 0;
}
```

Operator (1)

- Setelah variabel dan konstanta dideklarasikan, kita dapat mulai mengoperasikan dengan menggunakan *operator*.
- Operator assignment/penugasan → =

  Contoh : *x = 5;*

  x = y;

  y = 2 + (x = 5);

  x = y = z = 5;

Operator (2)

Operator Aritmatika

C opetration	Arithmetic operator	Algebraic expression	C expression
Addition	`+`	$f + 7$	`f + 7`
Subtraction	`-`	$p - c$	`p - c`
Multiplication	`*`	bm	`b * m`
Division	`/`	x/y or $\frac{x}{y}$ or $x \div y$	`x / y`
Remainder	`%`	r mod s	`r % s`

Operator (3)

- Perhitungan Aritmatika
  - Gunakan * untuk perkalian dan / untuk pembagian
  - Pembagian Integer menghilangkan sisa
    - 7 / 5 menghasilkan nilai 1
  - Operator Modulus (%) menghasilkan sisa hasil bagi
    - 7 % 5 menghasilkan nilai 2

Operator (4)

- Operator precedence
  - Beberapa operator aritmatika dilaksanakan sebelum yang lain (seperti, perkalian sebelum penjumlahan)
    - Gunakan parenthesis jika diperlukan
  - Contoh : Carilah nilai rata2 dari tiga variabel a, b dan c
    - Do not use: `a + b + c / 3`
    - Use: `( a + b + c ) / 3`

Operator (5)

- Urutan pengerjaan

Operator(s)	Operation(s)	Order of evaluation (precedence)
( )	Parentheses	Evaluated first. If the parentheses are nested, the expression in the innermost pair is evaluated first. If there are several pairs of parentheses “on the same level” (i.e., not nested), they are evaluated left to right.
* / %	Multiplication Division Remainder	Evaluated second. If there are several, they are evaluated left to right.
+ -	Addition Subtraction	Evaluated last. If there are several, they are evaluated left to right.

Operator (6)

- Compound assignment (+=, -=, *=, /=, %=, >>=, <<=, &=, ^=, |=)
- Contoh

Expression	Equivalent to …
y += x;	y = y+x;
x -= 5;	x = x – 5;
x /= y;	x = x / y;
price *= units + 1;	price = price * (units+1);

Operator (7)

- Increment and decrement (++, --)
- Beberapa ekspresi dapat dipendekkan bahkan increase operator (++) dan decrease operator (--). Contoh berikut adalah operasi yang sama

```
1 ++x;
2  x+=1;
3  x = x + 1;
```

Operator (8)

- Operator Relational dan comparison

  Dua ekspresi dapat dibandingkan menggunakan operator relational dan quality. Contoh, untuk mengetahui apakah dua nilai adalah sama atau apakah sebuah nilai lebih besar dari yang lain.
- Hasil operasi berupa nilai boolean true atau false.

Operator (9)

- Operator relational dalam C++ adalah:

Operator	Description
==	Equal to
!=	Not equal to
<	Less than
<=	Less than or equal to
>=	Greater than or equal to

Operator (10)

- Contoh :

1. (7 == 5) *// evaluates to false*
2. (5 > 4) *// evaluates to true*
3. (3 != 2) *// evaluates to true*
4. (6 >= 6) *// evaluates to true*
5. (5 < 5) *// evaluates to false*

Operator (11)

- **Logical operators (!, &&, ||)**

  Operator ! adalah operator C++ untuk operasi Boolean NOT. Ia hanya mempunyai satu operand, yakni di sebelah kanannya. Sebagai contoh:

  !(5 == 5) *// evaluates to false because the expression at its right (5 == 5) is true*

  !(6 <= 4) *// evaluates to true because (6 <= 4) would be false*

  !*true* *// evaluates to false*

  !*false* *// evaluates to true*

Operator (12)

- Operator logika && dan || digunakan ketika mengevaluasi dua ekspresi untuk mendapatkan hasil relational tunggal. Operator && sesuai dengan operasi logika Boolean AND, yang menghasilkan true jika kedua operand adalah true, dan sebaliknya false.

&& OPERATOR (and)

a	b	a && b
true	true	true
true	false	false
false	true	false
false	false	false

Operator (13)

- Operator || sesuai dengan operator logika Boolean OR, yang menghasilkan true jika salah satu operand adalah true, sehingga akan false hanya ketika kedua operand false. Berikut ini hasil yang mungkin dari a||b:

|| OPERATOR (or)

a	b	a || b
true	true	true
true	false	true
false	true	true
false	false	false

Operator (14)

- Sebagai contoh:

  1. ((5 == 5) && (3 > 6)) *// evaluates to false (true && false)*

  2. ((5 == 5) || (3 > 6)) *// evaluates to true (true || false)*

Operator (15)

- **Conditional ternary operator (?)**

  Operator conditional mengevaluasi sebuah ekspresi, mengembalikan sebuah nilai jika ekspresi menghasilkan nilai true, dan nilai yang lain jika ekspresi menghasilkan false. Syntaxnya adalah:

  condition ? result1 : result2

Operator (16)

- Jika kondisi adalah true, seluruh ekspresi menghasilkan result1, dan sebaliknya ke result2.
- 7==5 ? 4 : 3 *// menghasilkan nilai 3, karena 7 tidak sama dengan 5.*
- 7==5+2 ? 4 : 3 *// menghasilkan nilai 4, karena 7 sama dengan 5+2.*
- 5>3 ? a : b *// menghasilkan nilai a, karena 5 lebih besar dari 3.*
- a>b ? a : b *// menghasilkan yang mana saja yang lebih besar, a atau b.*

Operator (17)

- **Comma operator (,)**

  Operator comma (,) digunakan untuk memisahkan dua atau lebih ekspresi yang disertakan dimana hanya satu ekspresi yang diharapkan. Ketika himpunan ekspresi menghasilkan sebuah nilai, hanya ekspresi yang paling kanan yang terakhir dieksekusi.

Operator (18)

- Sebagai contoh, kode berikut:

  a = (b=3, b+2);

  pertama memberikan nilai 3 ke b, dan kemudian memberikan b+2 ke variabel a. Sehingga, pada akhirnya, variabel a akan berisi nilai 5 ketika variabel b berisi nilai 3.

Terima Kasih…..!