Pengenalan C++ untuk Interface

Perkembangan Bahasa Pemrograman

Bahasa Mesin

- Level terendah
- Isi:
  - kode-kode mesin yg hanya dapat diinterpretasikan langsung oleh mesin komputer
- Berupa kode numerik 0 dan 1
- Microcode:
  - sekumpulan instruksi dalam bahasa mesin
- (+) : Eksekusi cepat
- (-) : Sulit dipelajari manusia

Bahasa Assembly

- Bahasa simbol dari bahasa mesin
- Contoh: ADD, MUL, SUB, dll

- Macro instruksi:
  - sekumpulan kode dalam bahasa assembly

- (+) : Eksekusi cepat, masih dapat dipelajari daripada bahasa mesin, file kecil
- (-) : Tetap sulit dipelajari, program sangat panjang

Bahasa Tingkat Tinggi

- The 3rd Generation Programming Language

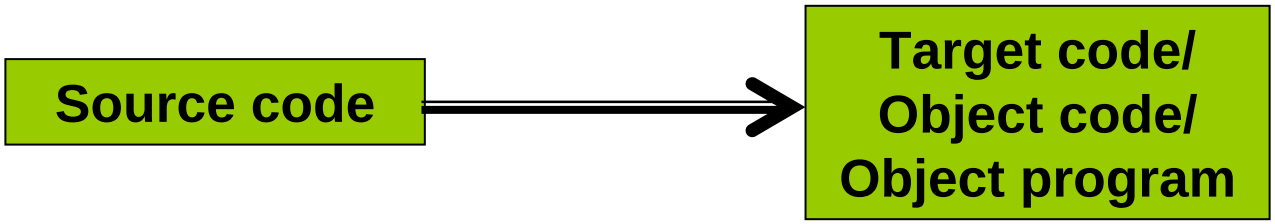
- Lebih dekat dengan bahasa manusia
- Memberi banyak fasilitas kemudahan dalam pembuatan program, mis.: variabel, tipe data, konstanta, struktur kontrol, loop, fungsi, prosedur, dll.
- Contoh: Pascal, Basic, C++, Java

- (+) : Mudah dipelajari, mendekati permasalahan yang akan dipecahkan, kode program pendek
- (-) : Eksekusi lambat

Specific Problem Oriented

- The 4th Generation Programming Language
- Digunakan langsung untuk memecahkan suatu masalah tertentu
- Contoh: SQL untuk database, Visual Basic, Delphi

Translator



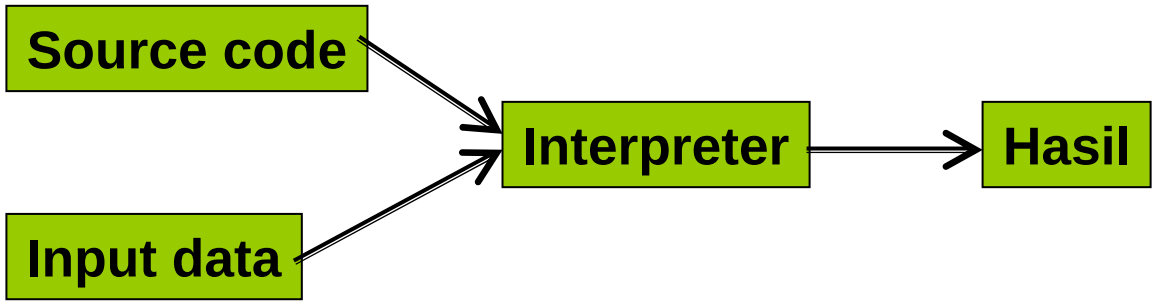
- *Source code*
  - ditulis dengan bahasa pemrograman tertentu
- *Object code*
  - bisa bermacam-macam, tergantung pada *translator*-nya

Macam Translator

- **Assembler**
- Source code adalah bahasa assembly
- Object code adalah bahasa mesin

Interpreter

- ***Input***
  - *source code* : bahasa scripting (PHP, ASP, Basic, dll)
  - masukan program dari *user*
- ***Output***
  - Tidak ada *object code*
  - Translasi internal



Interpreter (2)

- Program tidak harus dianalisis seluruhnya dulu tapi bersamaan dengan jalannya program

- (+) :
  - mudah bagi *user*
  - *debugging* cepat

- (-) :
  - eksekusi program lambat
  - tidak langsung menjadi program *executable*

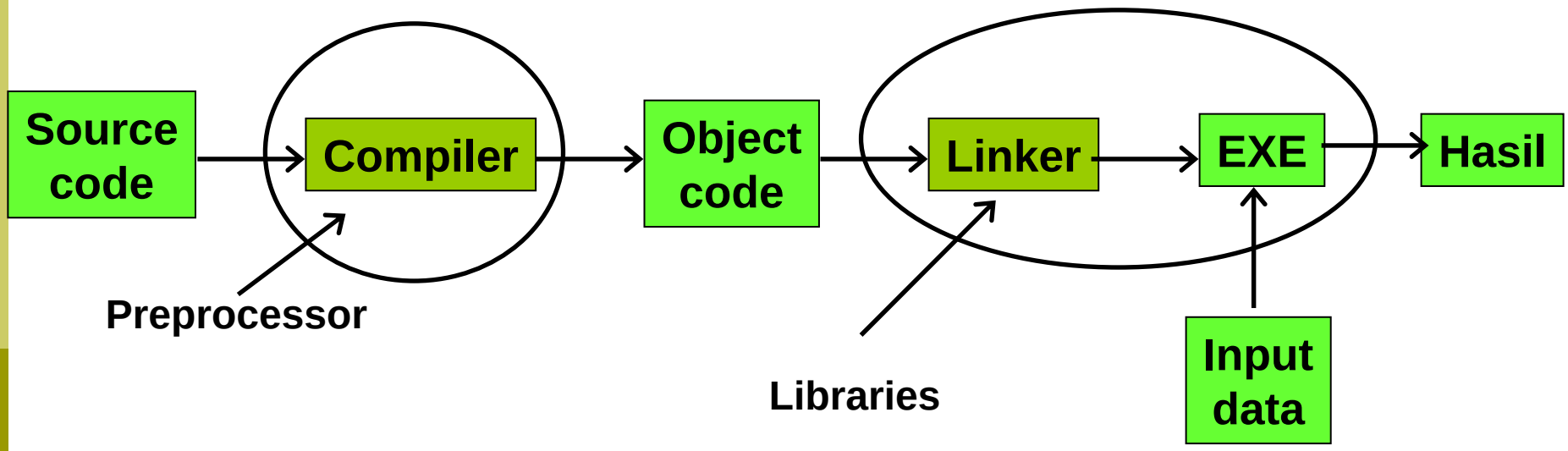
Kompiler

- ***Input***
  - *source code :* bahasa Pascal, C, C++
- ***Output***
  - *object code* : bahasa assembly atau EXE

Kompiler (2)

- *Compile time*
  - saat pengubahan *source code* menjadi *object code*
- *Runtime*
  - saat eksekusi *object code*, (dan menerima *input* dari *user*)

Kompiler (3)


Source code
Compiler
Preprocessor
Object code
Linker
Libraries
EXE
Input data
Hasil
Compile time
Runtime

Bahasa C

- Bahasa pemrograman tingkat menengah
- 1972:
  - Dirancang oleh **Dennis M Ritchie** di **Bell Laboratories**
- 1978:
  - Dennis dan Brian W. Kernighan mempublikasikan bahasa C melalui “The C Programming Language”
- 1989:
  - Bahasa C distandarisasi ANSI

Contoh Program

```
#include <stdio.h>

void main()
{
   printf(“Halo! Selamat Belajar C”);
}
```

Bahasa C

- Bahasa C dikatakan sebagai bahasa pemrograman terstruktur, karena strukturnya menggunakan fungsi-fungsi sebagai program-program bagian (*subroutine/ module*).
- Fungsi-fungsi selain fungsi utama disebut *subroutine/ module* dan ditulis setelah fungsi utama (**main**) atau diletakkan pada file pustaka (*library*).
- Jika fungsi-fungsi diletakkan pada file pustaka dan akan dipakai disuatu program, maka nama file *header*nya harus dilibatkan dalam program menggunakan *preprocessor directive* **#include**

Bahasa C

- Struktur Program C adalah:
  - Suatu program C minimal harus memiliki function **main()**, tanpa function itu maka program C **tidak dapat dieksekusi** tapi **bisa dikompilasi**.

```
<preprosessor directive>

void main(){

      <statement>;

      <statement>;

      <statement>;

}
```

```
<preprosessor directive>
int main(){
      <statement>;
      <statement>;
      <statement>;
      return 0;
}
```

Statement & Preprosesor Directive

- Statement adalah suatu baris instruksi/perintah tertentu. Statement menyebabkan suatu tindakan akan dilakukan oleh komputer.
- Preprosessor Directive adalah bagian yang berisi pengikutsertaan file atau berkas-berkas fungsi maupun pendefinisian konstanta atau fungsi makro tertentu.

Contoh suatu program C (2)

```
#include <stdio.h>
int main()
{
    int a,b,c;
    printf("Enter the first value:”);
    scanf(“%d”,&a);
    printf("Enter the second value:”);
    scanf(“%d”,&b);
    c = a + b;
    printf("%d + %d = %d\n",a,b,c);
    return 0
}
```

Keterangan

- Deklarasi variabel menyebabkan komputer menyediakan tempat yang diberi nama (*identifier*) **a**, **b** dan **c** dengan ukuran integer (2 byte = 16 bit).
- **printf** akan membuat komputer mengirim teks yang berada dalam fungsi tersebut ke layar monitor, sedangkan **scanf** membuat komputer menanti masukan dari pemakai melalui *keyboard*.

Keterangan (2)

- Pada program ini akan dikerjakan proses aritmatika, yaitu proses memberi nilai (*assignment* yang dipakai tanda “=”) variabel “c” dengan nilai yang ada dalam variabel “a” ditambah nilai yang ada dalam variabel “b”
- Yang terakhir adalah proses mencetak ke layar monitor dengan format yang sesuai

Statement

Contoh Statement

Instruksi/ Statement	***Tindakan***
`A = b * c ;`	Menghitung
`printf("Antonius Rachmat C");`	Menampilkan literal string
`scanf("%f",&Celcius);`	Menerima input data
`if(N<0) printf("negatif");`	Mengendalikan proses

Jenis Statement

- Macam:
  1. Statement kosong
  2. Statement ungkapan
  3. Statement kendali
  4. Statement jamak

Statement Kosong

- Empty statement = null statement
- Statement yang hanya terdiri dari pengakhir titik koma (;) saja
- Tidak ada tindakan yang akan dilakukan
- Contoh:
  - Memberi jarak waktu/delay

```
For (J=0; J<50000; J++);
```

Statement Ungkapan

- Expression statement
- Statement yang dibentuk dari suatu ungkapan
- Diakhiri dengan titik koma (;)
- Contoh:

```
scanf("%f",&Panjang);
scanf("%f",&Lebar);
Luas=Panjang*Lebar;
X=Y;
Y=Y+1;
```

Statement Kendali

- Control statement
- Statement yang digunakan untuk mengendalikan proses dari program, yaitu:
  - Penyeleksian kondisi (percabangan):
    - If, case dan switch
  - Lompatan (perulangan)
    - for, while, do-while, goto, break dan continue
- Contoh:

```
if (N<0) printf("Nilai N negatif");
```

Statement Jamak

- Compound statement = block statement
- Statement yang terdiri dari gabungan beberapa statement tunggal yang ditulis pada posisi di antara tanda kurung kurawal **(“{” dan “}”)**
- Contoh:

```
{ scanf(“%f”,&Panjang);
  scanf(“%f”,&Lebar);
  Luas=Panjang*Lebar;
  Printf(“Luas = %f”,Luas);
}
```

Struktur Program C (3)

- Selain function *main()* dapat ditambahkan function lain
- Jika function akan diletakkan di sembarang tempat dari function *main()*, maka function tersebut harus dideklarasikan terlebih dahulu sebelum function *main()*

```
#include <stdio.h>
int jumlahkan(int a, int b);

void main()
{  printf("Hasil 5 + 3 adalah %d", jumlahkan(5,3));
}

int jumlahkan(int a, int b)
{  return a+b;
}
```

```
#include <stdio.h>

int jumlahkan(int a, int b)
{   return a+b;
}

void main()
{  printf("Hasil 5 + 3 adalah %d", jumlahkan(5,3));
}
```

Identifier

- Identifier:
  - suatu tempat untuk menyimpan nilai
  - Diberi nama unik dan bisa memiliki tipe data
  - Dibagi menjadi 2:
    1. Konstanta
    2. Variabel
  - Dapat juga merupakan nama suatu elemen dalam program, mis.
    - Nama function
    - Nama prosedur
    - Nama tipe data, dll

Jenis Identifier

1. **Konstanta**
   - Identifier yang nilainya tetap selama program berjalan (dieksekusi)
   - Cara untuk mengubahnya hanya melalui *source code* saja

2. **Variabel**
   - Identifier yang nilainya dapat berubah atau diubah selama program berjalan (dieksekusi)
   - Pengubah: *user* atau proses

Standard Identifier

- Standard Identifier adalah identifier-identifier yang biasanya berupa fungsi-fungsi tertentu yang telah diberi makna tertentu oleh compiler bahasa C, tetapi tidak bersifat reserved sehingga masih bisa dipakai kembali oleh pemrogram.

- Contoh standard identifier :
- `#include <stdio.h>`
- `#include <conio.h>`
- `void main(){`
- `    clrscr();`
- `    printf("hallo bahasa C");`
- `}`

ATURAN PENULISAN IDENTIFIER

- Tidak boleh sama dengan nama keyword reserved, function, dan harus unik.
- Maksimum 32 karakter. Bila lebih, maka karakter selebihnya tidak akan diperhatikan oleh komputer.
- Case sensitive : membedakan huruf besar dan kecil
- Karakter pertama harus huruf atau underscore (_), selebihnya boleh angka.
- Tidak boleh mengandung spasi / blank

Keywords

- Adalah identifier yang telah didefinisikan oleh bahasa C
- Sifat:
  - Memiliki arti dan pemakaian tertentu
  - Reserved
  - Ditulis dalam huruf kecil
- Menurut standar ANSI: 32 keywords

Keywords (2)

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

Tipe Data

Type	Length	Range
unsigned char	8 bits	0 to 255
char	8 bits	-128 to 127
short int	16 bits	-32,768 to 32,767
unsigned int	32 bits	0 to 4,294,967,295
int	32 bits	-2,147,483,648 to 2,147,483,648
unsigned long	32 bits	0 to 4,294,967,295
enum	16 bits	-2,147,483,648 to 2,147,483,648
long	32 bits	-2,147,483,648 to 2,147,483,648

Tipe Data (2)

Type	Length	Range
float	32 bits	3.4 x 10-38 to 3.4 x 10+38
double	64 bits	1.7 x 10-308 to 1.7 x 10+308
long double	80 bits	3.4 x 10-4932 to 3.4 x 10+4932
near (pointer)	32 bits	Not applicable
far (pointer)	32 bits	Not applicable

- Cara untuk mengetahui ukuran sebuah tipe data di C: **`sizeoff(<tipe_data>)`**

Bahasa C - Int

- Bilangan Bulat
- Rangenya : -32768 sampai 32767 (16 bit)
- Deklarasinya :
  - int a;
- Untuk memberi nilai :
  - a = 1000;
- Contoh operasi :
  - a = a + 1000; //berapa nilai a sekarang ?

Bahasa C – Float & Double

- Bilangan real (pecahan, floating point)
- Float
  - Nilainya antara 1 E -36 sampai 1 E 36
  - Presisi 7 digit
  - 32 bit (4 byte)
- Double
  - Nilainya antara 1 E -303 sampai 1 E 303
  - Presisi 13 digit
  - 64 bit (8 byte)

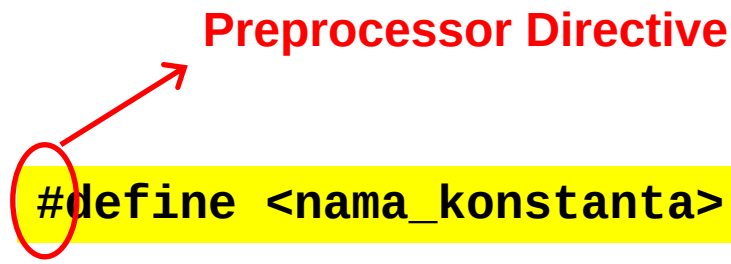
Bahasa C – Float & Double

- Deklarasinya :
  - float dataku;
  - double luaskubus;
- Untuk memberi nilai :
  - dataku = 3.245;
  - luaskubus = 4.56789665;

Review: Deklarasi Identifier

- **Variabel**
  - Bentuk umum: `<tipe_data> <nama_variabel>;`
  - Contoh:
    - `int umur;`

- **Konstanta**
  - Bentuk umum: `#define <nama_konstanta> <nilai>`
  - Contoh:
    - `#define pi 3.14`
    - `#define nama “antonius”`



Konstanta

- **Konstanta, bentuk 2:**
  - Contoh:
    - **const float phi = 3.14;**
    - **const char nama[] = “antonius”;**

const <tipe_data> <nama_konstanta> = <nilai>;

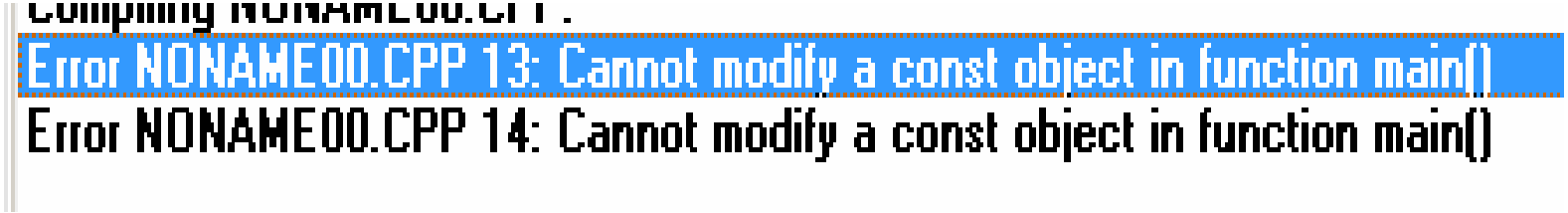
```c
#include <stdio.h>
const float phi = 3.14;
const char nama[] = "antonius";

void main(){
    printf("%f\n",phi);
    float luas = 7 * 7 * phi;
    printf("%f\n",luas);
    printf("%s\n",nama);
}
```

Error

Error karena #define:

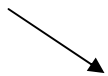
Error karena const:



Preprocessor Directive

- Bagian yang berisi pengikutsertaan file atau **berkas-berkas fungsi** maupun pendefinisian **konstanta**

```
#include <stdio.h>
#define pi 3.14
```



Tidak diakhiri titik koma

Bahasa C – File Header

- Formatnya : [**namafile.h**]
- Ada 2 macam penulisan
  - Diapit tanda < dan >
    - Contoh : **#include <stdio.h>**, digunakan bila mengakses file header dari library standar
  - Diapit tanda “ “
    - Contoh : **#include “tugas.h”**, digunakan bila mengakses file header tugas.h yang ada di direktori kerja

Karakter *Escape*

- Yaitu karakter yang diawali dengan **** (backslash)
- Masing-masing memiliki makna tertentu

Karakter *Escape* (2)

Karakter	Arti
\a	Bunyi bel (*speaker* komputer)
\b	Mundur satu spasi (*backspace*)
\f	Ganti halaman *(form feed)*
\n	Ganti baris baru *(new line)*
\r	Ke kolom pertama baris yang sama *(carriage return)*
\t	Tabulasi horisontal
\v	Tabulasi vertikal
\0	Nilai kosong *(null)*
\’	Karakter petik tunggal
\””	Karakter petik ganda
\\	Garis miring terbalik *(back slash)*

Sifat Data Numerik *Integer*

Nilai numerik pecahan yang disimpan dalam integer akan dibulatkan ke **bawah**. Jadi, nilai pecahan **dibuang**

- Contoh:
  - 2.38 → 2
  - 4.928 → 4

Sifat Data Numerik *Integer* (2)

Nilai variabel yang melebihi jangkauannya akan dipotong sepanjang jumlah bit yang tersedia

- Contoh:

  Jika dideklarasikan variabel integer (16 bit = 2 byte) berarti hanya menyimpan sampai dengan 32,767.

  Jika variabel diisi nilai 70,000 (1 0001 0001 0111 0000), padahal 70,000 menempati 17 bit maka bit paling kiri akan dipotong menjadi (0001 0001 0111 0000),

Sifat Data Numerik *Integer* (3)

- Contoh:

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

void main()
{   int x;

    x = 70000;

    printf("x = %d\n",x);  //hasil 4464
}
```

Casting

- Pemaksaan suatu tipe data ke tipe data lain

```c
#include <stdio.h>

void main(){
   int a = 5;
   int b = 2;

   printf("5/2 dlm integer = %d\n",a/b);
   //printf("%f",a/b);

   float c = 5.0;
   float d = 2.0;

   printf("5.0/2.0 dlm float = %.2f\n",c/d);
   printf("5.0/2.0 dlm int = %d\n",c/d);

   printf("5/2 casting ke float = %f\n",(float)a/b);
   printf("5/2 casting ke float = %f\n",(float)a/(float)b);
   printf("5.0/2.0 casting ke int = %d\n",(int)c/d);
   printf("5.0/2.0 casting ke int = %d\n",(int)c/(int)d);

}
```

Sifat Data Karakter

Karakter disimpan dalam memori berupa kode ASCII.

- ASCII
  - Berdasarkan English Alphabet
  - Dipublikasikan tahun 1967
  - Di-*update* tahun 1986
  - Terdiri dari 95 karakter yang *printable* (33-126) dan 32 (0-31) *non-printable/control character*

Sifat Data Karakter (1)

- Menggunakan tanda petik satu (‘)
- Contoh:

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

void main()
{   char hrf;

    hrf = ‘A’;

    printf(“Nilai desimal karakter %c adalah %d”,
    hrf, hrf);
}
```

Sifat Data Karakter (2)

Pada tipe data karakter dapat dilakukan operasi matematika

- Contoh:

```
char hrf;
hrf = 'A';
printf("Nilai desimal karakter %c adalah
%d\n",hrf, hrf);
printf("Huruf kecilnya = %c",(hrf+32));
```

Sifat Data String

- Bahasa C tidak memiliki tipe data ***String***
- ***String*** diperlakukan sebagai *array of character* (kumpulan karakter)
- Menggunakan tanda petik dua (“)

- Deklarasi:

```
char nama[20]=“anton”;
printf(“%s”,nama);
```

Operator

Kategori	Operator
Arithmetic	+ - * / %
Logical (boolean and bitwise)	& | ^ ! ~ && || true false
String concatenation	+
Increment, decrement	++ --
Shift	<< >>
Relational	== != < > <= >=
Assignment	= += -= *= /= %= &= |= ^= <<= >>=

Operator (2)

Kategori	Operator
Member access	.
Indexing	[ ]
Cast	( )
Conditional	*?:*
Delegate concatenation and removal	+ -
Type information	As is sizeof typeof
Overflow exception control	Checked unchecked
Indirection and Address	* -> [ ] &

Operator Aritmatika

Oprtr	Contoh	Keterangan
+	op1 + op2	Menjumlahkan dua operand
-	op1 - op2	Mengurangkan dua operand
*	op1 * op2	Mengalikan dua operand
/	op1 / op2	Membagi dua operand
%	op1 % op2	Menghasilkan sisa hasil bagi dari pembagian operand

- Operator Modulus tidak dapat dioperasikan ke tipe data float atau double

Operator Aritmatika (2)

Oprtr	Contoh	Keterangan
++	op++	**Op dinaikkan nilainya 1 setelah dilakukan operasi pada op**
++	++op	**Op dinaikkan nilainya 1 sebelum dilakukan operasi pada op**
--	op--	Op diturunkan nilainya 1 setelah dilakukan operasi pada op
--	--op	Op diturunkan nilainya 1 sebelum dilakukan operasi pada op
-	-op	Menegaskan nilai op menjadi positif jika negatif atau sebaliknya

```
void main(){
    int a = 5;
    int h = a++;
    printf("h = %d\n",h);
    printf("a = %d",a);
}
```

```
h = 5
a = 6
```

```
#include <stdio.h>

void main(){
    int a = 5;
    int h = ++a;
    printf("h = %d\n",h);
    printf("a = %d",a);
}
```

```
h = 6
a = 6
```

Operator Relasional

Oprtr	Contoh	Keterangan
>	op1 > op2	Menghasilkan true jika op1 lebih besar dari op2
<	op1 < op2	Menghasilkan true jika op1 lebih kecil dari op2
>=	op1 >= op2	Menghasilkan true jika op1 lebih besar atau sama dengan op2
<=	op1 <= op2	Menghasilkan true jika op1 lebih kecil atau sama dengan op2
!=	op1 != op2	Menghasilkan true jika op1 tidak sama dengan op2

Operator Kondisional

Oprtr	Contoh	Keterangan
&&	op1 && op2	Menghasilkan true jika op1 dan op2 true
||	op1 || op2	Menghasilkan true jika op1 atau op2 true
!	!op1	Menghasilkan true jika op1 bernilai false
&	op1 & op2	Bitwise AND
|	op1 | op2	Bitwise OR
^	op1 != op2	Menghasilkan true jika salah satu true, tetapi tidak keduanya

Contoh

- Misalnya, A bernilai 5, B bernilai 7, dan C bernilai ‘a’, maka ungkapan di bawah ini mempunyai hasil akhir benar atau salah?

A < B || B == 7 && C > ‘z’

Contoh: Hasil

- Hasil akhir: **benar**
- Langkah-langkah:
  1. Jenjang operator relasional lebih tinggi dibandingkan dengan operator logika, jadi operator relasional dikerjakan lebih dahulu
  2. Operator logika ‘&&’ mempunyai jenjang lebih tinggi dari operator ‘||’, sehingga operator ‘&&’ dikerjakan lebih dahulu
  3. Bagian yang paling akhir dikerjakan adalah operator ‘||’

Beberapa Ungkapan

Ungkapan	Arti
X / Y	X dibagi Y
X = 10	X diisi nilainya dengan 10
Y = Y + 1	Y diisi dengan nilai Y sebelumnya ditambah 1
Y = X	Y diisi dengan nilai X
X += Y	Sama dengan X = X + Y
X /= Y	Sama dengan X = X / Y

Pemberian Komentar

- Pemberian komentar sangat penting dalam menulis program agar program tersebut terdokumentasi dengan baik.
- Program yang terdokumentasi dengan baik berarti alur dan logika program tersebut jelas, dapat dibaca dengan mudah pada lain waktu.
- Semua komenter dalam bahasa C tidak akan dibaca atau akan diabaikan oleh compiler bahasa C.
- Komentar dalam banyak baris diawali dengan tanda /* , kemudian setelah semua komentar ditulis, diakhiri dengan tanda */ sebagai penutupnya.
- Sedangkan untuk komentar dalam satu baris saja, ditulis dengan tanda // di awal kalimat komentar.


Input-Output

Output di Bahasa C

- Header stdio.h
- **printf**(<string>,[<variabel>]),
  puts(<string>) atau **putchar**(<char>)

Contoh di C:

```
#include <stdio.h>

void main(){

      char nama[50] = "anton";

      printf("Hallo saya bernama %s, saya sedang belajar C!",nama);

}
```

Output Tidak Terformat

- **putchar(char)** dan **puts(char[]).**
- puts diakhiri dgn enter

Contohnya:

```
#include <stdio.h>
void main()
{     char N,D[15] = "antonius rc";
      N = 'X';
      putchar(N);
      puts(D);
}
```

Hasilnya adalah : **Xantonius rc**

Output Tidak Terformat

- **(+)** Bentuknya sederhana

- **(-)** Tidak tidak dapat digunakan untuk menampilkan bentuk yang rumit
- **(-)** Hanya dapat menggunakan sebuah argumen saja.

Output Terformat

Perintah untuk menampilkan hasil terformat adalah **printf()**

Kode Format	Kegunaan
%c	Menampilkan sebuah karakter
%s	Menampilkan nilai string
%d	Menampilkan nilai desimal integer
%i	Menampilkan nilai desimal integer
%u	Menampilkan nilai desimal integer tak bertanda
%x	Menampilkan nilai heksa desimal integer
%o	Menampilkan nilai oktal integer
%f	Menampilkan nilai pecahan
%e	Menampilkan nilai dalam notasi saintifik
%g	Sebagai pengganti %f atau %e tergantung yg terpendek
%p	Menampilkan suatu alamat memori untuk pointer

Menampilkan Karakter

- Menampilkan karakter di C secara terformat, kita dapat menggunakan **“%c”.**
- Untuk menampilkan sebuah karakter dengan lebar 3 posisi (tiga karakter di depan, karakternya blank), maka gunakan **“%3c”**
- Untuk membuat rata kiri (blank ada di sebelah kanan karakternya) dapat digunakan simbol (*flag*) minus, misalnya “**%-3c**”.

```
#include <stdio.h>

void main(){
      char c = 'a';
      printf("%3c\n",c);
      printf("%-3c\n",c);
}
```

Hasilnya:

```
  a
a
```

Menampilkan String Terformat

FORMAT	KETERANGAN
"%s"	Menampilkan semua nilai karakter pada nilai string
"%Ns"	Menampilkan semua karakter rata kanan dengan lebar N posisi; N adalah konstanta numerik bulat.
"%-Ns"	Menampilkan semua karakter rata kiri dengan lebar N posisi; N adalah konstanta numerik bulat.
"%N.Ms"	Menampilkan rata kanan hanya M buah karakter pertama saja dengan lebar N posisi; M dan N adalah konstanta numerik bulat.
"%-N.Ms"	Menampilkan rata kiri hanya M buah karakter pertama saja dengan lebar N posisi; M dan N adalah konstanta numerik bulat.

Contoh Output Terformat

```
#include <stdio.h>

main()
{
      char D[15] = "Antonius Rachmat C";
      printf("12345678901234567890\n");
      printf("%s\n",D); /* semua karakter, rata kiri */
      printf("%20s\n",D); /* lebar 20, rata kanan */
      printf("%-20s\n",D); /* lebar 20, rata kiri */
      printf("%20.5s\n",D); /* 5 karakter lbr 20, rata kanan */
      printf("%-20.5s\n",D); /* 5 karakter lbr 20, rata kiri */
}
```

Hasil

```
12345678901234567890
Antonius Rachmat C
  Antonius Rachmat C
Antonius Rachmat C
               Anton
Anton
```

Integer Terformat

FORMAT	ARTI
"%d", "%i"	signed int
"%u"	unsigned int
"%ld","%li"	long int
"%hi"	short int
"%hu"	unsgined short int
"%lu"	unsigned long int

Contoh Integer Terformat

```
#include <stdio.h>
main()
{
            int i=1234;
            printf("%i\n",i);
            printf("%5i\n",i);
            printf("%7d\n",i);
            printf("%07d\n",i);
            printf("%-7d\n",i);
}
```

Hasilnya:

```
1234
 1234
   1234
0001234
1234
```

Menampilkan Bilangan Pecahan

FORMAT	ARTI
"%f"	float dgn nilai pecahan
"%e"	float dgn notasi saintifik
"%g"	terpendek dari "%f" atau "%e"
"%lf", "%le" atau "lg"	double
"%Lf", "%Le" atau "Lg"	long double

Contoh Pecahan

```
#include <stdio.h>
void main()
{
   float x=123.4567;
   printf("%3f %15f %020f\n",x,x,x);
   printf("%3e %15e %020e\n",x,x,x);
}
```

Hasilnya:

```
123.456703      123.456703 0000000000123.456703
1.23457e+02     1.23457e+02 0000000001.23457e+02
```

Contoh2:

```
#include <stdio.h>
void main()
{
    float F=12345.6789;
    printf("%15f\n",F);
    printf("%15.2f\n",F);
    printf("%015.2f\n",F);
    printf("%-15.2f\n",F);
    printf("%15.0f\n",F);
}
```

Hasilnya:

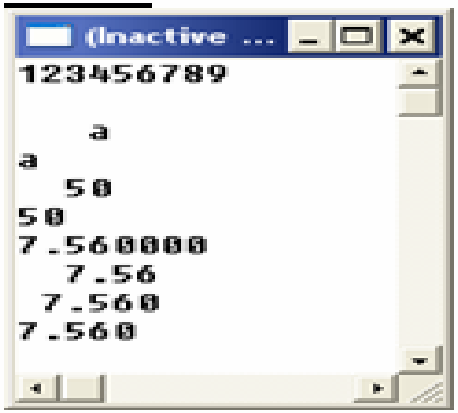
```
    12345.678711
        12345.68
000000012345.68
12345.68
             123456
```

Contoh Pecahan

Contoh3:

```
#include <stdio.h>
void main(){
                printf("123456789\n\n");
                printf("%4c\n",'a');
                printf("%-4c\n",'a');
                printf("%4d\n",50);
                printf("%-4d\n",50);
                printf("%6f\n",7.56);
                printf("%6.2f\n",7.56);
                printf("%6.3f\n",7.56);
                printf("%-6.3f\n",7.56);
}
```

Hasil:



Hexadesimal & Octal

```
#include <stdio.h>
void main()
{
    int x = 1234;
    printf("%x\n",x);     //hasil = 4d2
}
```

```
#include <stdio.h>
void main()
{
      int O=1234;
      printf("%o\n");      //Hasil = 2322
}
```

Membersihkan Layar & Meletakkan Kursor

Menggunakan header **conio.h** dengan fungsi yang bernama **clrscr()**

Contoh :

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

void main()
{     clrscr();
      printf("Layar sudah bersih...");
}
```

Mengggunakan header **conio.h** dengan fungsi yang bernama **gotoxy()**

Contoh :

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

void main()
{    clrscr();
     gotoxy(20,1);printf("Layar sudah bersih...");
     gotoxy(20,3);printf("Ini baris ke 3, kolom 20");
     gotoxy(20,5);printf("Ini baris ke 5, kolom 20");
     gotoxy(25,6);printf("Ini kolom 25, baris ke 6");
     gotoxy(20,7);printf("Ini kolom 20, Baris ke 7");
}
```

Input Data

- Header **stdio.h:**
  - **gets()**
  - **scanf()**
- Header **conio.h:**
  - **getche()**
  - **getchar()**
  - **getch()**

Input Data Karakter Tidak Terformat

- getche(): Tanpa Enter, karakter terlihat
- getchar(): Dengan Enter, Karakter terlihat
- getch(): Tanpa Enter, karakter tdk terlihat

Contoh 1:

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

void main()
{ char hrf;
  printf("Masukkan sebuah karakter : "); hrf = getche();
  printf("\nNilai yang dimasukkan : %c\n",hrf);
}
```

Hasilnya :

```
Masukkan sebuah karakter : N
Nilai yang dimasukkan : N
```

Contoh Input

Contoh 2:

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

void main()
{       char hrf;
        printf("Masukkan sebuah karakter : ");
        hrf = getchar();
        printf("\nNilai yang dimasukkan : %c\n",hrf);
}
```

Hasilnya :

```
Masukkan sebuah karakter : N↵
Nilai yang dimasukkan : N
```

Contoh 3:

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

void main()
{       char hrf;
        printf("Masukkan sebuah karakter : ");
        hrf = getch();
        printf("\nNilai yang dimasukkan : %c\n",hrf);
}
```

Hasilnya :

```
Masukkan sebuah karakter :
Nilai yang dimasukkan : N
```

Input Data String tidak terformat

- Untuk memasukkan nilai string dapat dipakai fungsi **gets()**

Contoh:

```
#include <stdio.h>

void main()
{     char kata[10];
      printf("Masukkan suatu nilai string : ");
      gets(kata);
      printf("Nilai string yang dimasukkan : %s\n",kata);
}
```

Hasilnya :

```
Masukkan suatu nilai string : anton
Nilai string yang dimasukkan : anton
```

Input Data Terformat

- **Menggunakan scanf(kodeformat,variabel)**

Kode Format	Kegunaan
%c	Membaca sebuah karakter
%s	Membaca sebuah data string
%d	Membaca sebuah nilai desimal integer
%i	Membaca sebuah nilai desimal integer
%x	Membaca sebuah nilai heksa desimal
%o	integer
%f	Membaca sebuah nilai oktal integer
%e	Membaca sebuah data pecahan
%g	Membaca sebuah data pecahan
	Membaca sebuah data pecahan

Input Data Karakter Terformat

Contoh:

```
    #include <stdio.h>

    void main()
    {    char c1,c2,c3;
         printf("Masukkan 3 nilai karakter : ");
         scanf("%c%c%c",&c1,&c2,&c3);
         printf("\nAnda memasukkan : %c %c %c\n",c1,c2,c3);
    }
```

Hasilnya :

```
Masukkan 3 nilai karakter : abcde
Anda memasukkan : a b c
```

Input Data String Terformat

Contoh:

```
#include <stdio.h>

void main()
{    char kata[10];
     printf("Masukkan suatu nilai string : ");
     scanf("%s",kata);
     printf("Nilai string yang dimasukkan : %s\n",kata);}
```

Hasilnya :

```
Masukkan suatu nilai string : Anton
Nilai string yang dimasukkan : Anton
```

Perhatian

Scanf(<format>,<variabel>):

- Jika string yang dimasukkan memiliki whitespace karakter, maka input string hanya akan dibaca sampai dengan karakter sebelum whitespace saja!

Solusi:

- kode format **“%s”** dapat diganti dengan “**%[^\n]**”
  - Berarti bahwa karakter nilai string akan dibaca terus sampai ditemui penekanan tombol Enter (bentuk ‘^’ menunjukkan maksud ‘tidak’ dan karakter ‘\n’ artinye Enter). Sehingga dengan demikian semua karakter temasuk spasi dan tabulasi akan dibaca sampai ditemui penekanan tombol Enter.
- Atau dengan **gets(<string>)**

Contoh

```
#include <stdio.h>

void main()
{    char kata[20];
     printf("Masukkan suatu nilai string : ");
     scanf("%[^\n]",kata);
     printf("Nilai string yang dimasukkan : %s\n",kata);
}
```

Hasilnya :

```
Masukkan suatu nilai string : Antonius RC
Nilai string yang dimasukkan : Antonius RC
```

Memasukkan Nilai Numerik

- Menggunakan %d untuk integer
- Menggunakan %i untuk integer
- Menggunakan %ld atau %li untuk long integer
- Menggunakan %f untuk double dan float
- Menggunakan %le atau %lf dan %lg untuk long double