3. Elemen Dasar C++

S. Indriani L., M.T

Himpunan karakter

Suatu pengenal berupa satu atau beberapa karakter

- Huruf

  A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

  a b c d e f g h i j k l m n o p q r s t u v w x y z
- Digit

  0 1 2 3 4 5 6 7 8 9
- Simbol dan lain-lain

  - + _ * …dsb

Pengenal (identifier)

- Pengenal adalah suatu nama yang biasa dipakai dalam pemograman untuk menyatakan:
  - ✓variabel
  - ✓konstanta bernama
  - ✓tipe data
  - ✓fungsi
  - ✓label
  - ✓objek

  serta hal-hal lain yang dideklarasikan atau didefinisikan oleh pemogram.

Penamaan Pengenal

- Pada C++, huruf kecil dan huruf kapital pada suatu pengenal tidak dianggap sama.

  Sifat ini dikenal dengan *Case Sensitive*.
- Contoh pengenal absah dan tidak absah.

Absah	Tidak Absah	
nama	2 semester	(tidak boleh diawali angka)
NAMA	Nama-barang	(tanda – tidak diperkenankan)
nama_barang	#brg	(simbol # tidak boleh digunakan)
kuartal_2	nama barang	(tidak boleh mengandung spasi)

- Untuk menghindari kesalahan, pengenal tidak boleh menggunakan nama yang tergolong sebagai kata kunci.
- Harus pula dihindari pemakaian pengenal yang dipakai untuk nama fungsi, konstanta, variabel ataupun hal lain yang digunakan pada pustaka C++ (contohnya seperti `cout`)

Kata Kunci

- Kata kunci (*keyword*) adalah pengenal sistem yang mempunyai makna khusus bagi kompiler.
- Kegunaan dari golongan ini tidak dapat diubah.

asm	else	operator	template
auto	enum	private	this
break	extern	protected	typedef
case	float	public	union
char	for	register	unsigned
class	friend	return	virtual
const	goto	short	void
continue	if	signed	volatile
default	inline	sizeof	while
delete	int	static	
do	long	struct	
double	new	switch	

Tipe Data

Tipe	Sebutan dalam bahasa C++	Ukuran (bits)	Range
Character	unsigned char	8	0 s/d 255
	char atau signed char	8	-128 s/d 127
Interger	unsigned int atau unsigned	16	0 s/d 65,535
	int atau signed int atau signed	16	-32,768 s/d 32,767
	unsigned long atau unsigned long int	32	0 s/d 4,294,967,295
	long atau long int atau signed long atau signed long int	32	-2,147,483,648 s/d 2,147,483,647
Floating point single precision	Float	32	3.4 E-38 s/d 3.4 E38
Floating point double precision	double	64	1.7E-308 s/d 1.7E308
	long double	80	3.4E-4932 s/d 1.1E4932

Variabel dan Konstanta

- Data pada C++ tersusun dari:

  - Variabel

    Variabel digunakan dalam program untuk menyimpan suatu nilai, dan nilai yang ada padanya dapat diubah selama eksekusi program berlangsung

  - Konstanta

    Konstanta menyatakan nilai yang tetap. Misalnya, 234 adalah sebuah konstanta bilangan bulat.

Mendeklarasikan & Mendefinisikan variabel

- Pendefinisian variabel:

  tipe data_variabel;

  contoh:

```
int jumlah;
float harga_per_unit,total_penjualan
```

- Memberikan nilai ke variabel:

  variabel=nilai;

  contoh:

```
jumlah=10;
harga_per_unit=17.5;
```

Karakter-karakter khusus

Karakter	Keterangan
\0	Karakter ber-ASCII nol (karakter nul)
\a	Karakter bel
\b	Karakter backspace
\f	Karakter formfeed (ganti halaman)
\n	Karakter newline (pindah baris)
\r	Karakter carriage return (ke baris awal) tanpa linefeed
\t	Karakter tab horisontal
\v	Karakter tab vertikal
\ooo	Karakter yang nilai oktalnya adalah ooo (3 digit oktal)
\xhh	Karakter yang nilai heksadesimalnya adalah hh (dua digit heksadesimal)

Operator

1. Operator Assign (=)
2. Operator Aritmatika (+, -, *, / , %)
3. Operator Penurunan (--) dan Kenaikan (++)
4. Operator Majemuk (+=, -=, *=, /=, %=, <<=, >>=, &=, |=)
5. Operator Relasional (==, !=, >, <, >=, <=)
6. Operator Logika (!, &&, | |)
7. Operator Kondisi (?:)

Pengantar Operator

- Operator merupakan simbol yang biasa dilibatkan dalam program untuk melakukan sesuatu operasi atau manipulasi. Misalnya:
  - Menjumlahkan dua buah nilai
  - Memberikan nilai ke suatu variabel
  - Membandingkan kesamaan dua buah nilai
- Sifat operator:

Sifat	Keterangan	Contoh
Unary	Operator ini hanya melibatkan sebuah operand	-1
Binary	Operator ini melibatkan dua buah operand	1+2
Ternary	Operator ini melibatkan tiga buah operand	(a > b) ? a : b

Operator

1. Operator Assign (=)

   → Operator (=) akan memberikan nilai kedalam suatu variabel.

2. Operator Aritmatika

   → Operator *binary*

Operator	Keterangan
+	Penjumlahan
-	Pengurangan
*	Perkalian
/	Pembagian
%	Sisa Pembagian (modulus)

→Operator *Unary*

Operator	Keterangan	Contoh
-	Tanda Minus	-2
+	Tanda plus	+4

Untuk operasi aritmatika memiliki prioritas !

Prioritas Operator Aritmatika

- Operator yang mempunyai prioritas tinggi akan diutamakan dalam hal pengerjaan dibandingkan dengan operator yang memiliki prioritas lebih rendah.

Operator	Prioritas
+ -- (khusus yang berkedudukan sebagai awalan)	Tertinggi
- (Unary minus)	
* / %	
+ -	Terendah

- Jika operator memiliki prioritas sama (tabel diatas terletak pada baris yang sama), operator yang terletak di sebelah kiri dalam suatu ungkapan yang akan diutamakan untuk dikerjakan terlebih dahulu.

contoh prioritas operator aritmatika

- `x=2*3%2;`

  operator * dan % mempunyai prioritas yang sama, namun karena yang terletak disebelah kiri adalah *, maka 2*3 akan dikerjakan terlebih dahulu

- Tanda kurung untuk mengubah prioritas

  tanda kurung biasa digunakan untuk mengubah urutan pengerjaan, misalnya:

  `x=(2+3)*2;`

3. Operator Penurunan dan Kenaikan

Operator	Keterangan
++	Operator Penaikan (*increment*)
--	Operator Penurunan (*decrement*)

Operator penaikan digunakan untuk menaikkan variabel sebesar satu, sedangkan operator penurunan dipakai untuk menurunkan variabel sebesar satu. Penempatan operator terhadap variabel dapat dilakukan di muka atau dibelakangnya.

contoh:

x=x+1; bisa ditulis menjadi → ++x; atau x++;

y=y-1; bisa ditulis menjadi → --y; atau y--;

- Efek dari penempatan operator *increment* dibelakang contoh:

```
int r=10;
int s;
s=10 + r++;
```

hasilnya: r =11, s=20

→ `s=10 + r++;`

identik dengan:

```
s=10+r;
r=r+1;
```

s diisi dengan penjumlahan nilai 10 dan r, dengan demikian s akan bernilai 20. setelah s diisi dengan 20, nilai r baru dinaikkan (karena operator ++ ditulis dibelakang, disebut *post-increment*) yang artinya dinaikkan belakangan setelah penjumlahan antara 10 dan r dilaksanakan)

- Efek dari penempatan operator *increment* `didepan`: contoh:

```
int r=10;
int s;
s= 10 + ++r;
```

hasilnya: r =11, s=21

→ `s= 10 + ++r;`

identik dengan:

```
r=r+1;
s=10+r
```

nilai r mula-mula dinaikkan terlebih dahulu (sebab operator `++` ditempatkan di depan, disebut *pre-increment*) kemudian dijumlahkan dengan 10 dan diberikan ke s.

4. Operator Majemuk

digunakan untuk memendekkan penulisan operasi penugasan

Operator	Contoh	Keterangan
+=	x+=2;	Kependekan dari x=x+2;
-=	x-=2;	Kependekan dari x=x-2;
*=	x*=2;	Kependekan dari x=x*2;
/=	x/=2;	Kependekan dari x=x/2;
%=	x%=2;	Kependekan dari x=x%2;
<=	x<=2;	Kependekan dari x=x<2;
>=	x>=2	Kependekan dari x=x>2;
&=	x&=2	Kependekan dari x=x&2;
|=	x|=2;	Kependekan dari x=x|2;
^=	x^=2	Kependekan dari x=x^2

```
//contoh program operator majemuk

#include<iostream.h>
#include<conio.h>

void main()
{
   clrscr();
   int x = 2;
   cout << “x = ” << x << endl;
   x+=3;
   cout << “setelah x+=3,x->   “ << x <<endl;
   x*=3;
   cout << “setelah x*=3,x->   “ << x <<endl;
}
```

Ungkapan Kondisi

- Ungkapan kondisi adalah ungkapan yang menjadi dasar begi pernyataan berkondisi (misalnya if), dimana ungkapan ini memberikan nilai benar dan salah. Hasil ungkapan berupa:

  0 kalau ungkapan bernilai salah.

  1 kalau ungkapan bernilai benar.

- Adapun elemen yang membentuk ungkapan ini adalah operator:
  - ✓Relasi, dan
  - ✓Logika

5. Operator Relasi

- Operator relasi biasa digunakan untuk membandingkan dua buah nilai.

Operator	Keterangan
==	Sama dengan (bukan penugasan)
!=	Tidak sama dengan
>	Lebih dari
<	Kurang dari
>=	Lebih dari atau sama dengan
<=	Kurang dari atau sama dengan

Contoh:

x == y

hasil ungkapan ini bernilai:

1 kalau nilai x sama dengan y

0 kalau nilai x tidak sama dengan y

```
//contoh prgram operator relasi

#include<iostream.h>
#include<conio.h>

void main()
{  int nilai;

   clrscr();

   nilai = 3>2;  //hasil ungkapan: benar
   cout << "nilai =   " <<  nilai <<endl;

   nilai = 2>3;  //hasil ungkapan: salah
   cout << "nilai =   " << nilai <<endl;
}
```

6. Operator Logika

- Operator Logika biasa digunakan untuk menghubungkan dua buah ungkapan kondisi menjadi sebuah ungkapan kondisi.
- Operator-operator ini berupa:

  && → operator logika *and* (dan)

  || → operator logika *or* (atau)

  ! → operator logika *not* (bukan)

- Operator || , &&, dan !

```
(ungkapan1 || ungkapan 2)
(ungkapan1 && ungkapan 2 )
(!ungkapan)
```

- `(ungkapan1 || ungkapan 2)`

  Bentuk ungkapan menggunakan operator AND, hasil ungkapan bernilai benar jika *ungkapan1* dan *ungkapan2* bernilai benar

- `(ungkapan1 && ungkapan 2 )`

  Bentuk ungkapan mengunakan operator OR, hasil ungkapan bernilai benar hanya kalau *ungkapan1* dan *ungkapan2* bernilai benar

- `(!ungkapan)`

  Bentuk ungkapan menggunakan operator NOT, hasilnya akan berupa:

  - benar kalau ungkapan bernilai salah

  - salah kalau ungkapan bernilai benar

```
//contoh program operator logika

#include<iostream.h>
#include<conio.h>

void main()
{
  int x = 200;

  clrscr();

  cout<<“(x>=1)&&(x<=50) ->” <<((x>=1)&&(x<=50) )<<endl
  cout<<“(x>=1)||(x<=50) ->” <<((x>=1)||(x<=50) )<<endl
}
```

7. Operator kondisi

- Operator kondisi biasa dipakai untuk mendapatkan sebuah nilai dari dua buah kemungkinan, berdasarkan suatu kondisi.
- Format pemakaiannya:

  `ungkapan1 ? ungkapan2 : ungkapan3`
- Ada tiga ungkapan yang dilibatkan. Oleh karena itu operator ?: tergolong sebagai operator ternary.

  pada bentuk seperti diatas, hasil dari ungkapan berupa:
  - Nilai dari `ungkapan2` kalau `ungkapan1` bernilai benar
  - Nilai dari `ungkapan3` kalau `ungkapan1` bernilai salah

```
// contoh program operator kondisi

#include<iostream.h>
#include<conio.h>

void main()
{
     int bil1,bil2, minim;
     clrscr();
     bil1 = 53;
     bil2 = 6;
     minim = bil1<bil1?bil1:bil2;
     cout<<"Bilangan terkecil="<<minim<<endl;
}
```

Operator Bitwise (Manipulasi Bit)

- Untuk keperluan memanipulasi data dalam bentuk bit, C++ menyediakan enam buah operator, sbb:

Operator	Keterangan	Contoh
<<	Geser bit ke kiri	25<<2
>>	Geser bit ke kanan	25>>2
&	Bitwise AND	25&2
|	Bitwise OR	25|2
^	Bitwise XOR	25^2
-	Bitwise NOT (komplemen)	-25

- Seluruh operator *bitwise* hanya bisa dikenakan pada operand bertipe interger atau karakter.

- Kegunaan dari operator-operator ini diantaranya adalah untuk mengakses bit secara individual didalam memori, misalnya pada memori gambar.
- Operator bitwise mempunyai prioritas lebih rendah dibandingkan operator aritmatika.
- Contoh: operator << (geser ke kiri)

  nilai<<jumlah bit digeser ke kiri

  0000000001011101 = 93

  / / / / / / / / / / / / / / / ←digeser ke kiri 1 bit

  0000000010111010 (bagian kanan selalu disisipi 0)= 186

```
//contoh program operator bitwise

#include<iostream.h>
#include<conio.h>

main()
{  unsigned char x=93;

   clrscr();

   cout << “nilai x semula=   ” << x << ‘\n’;

   x = x<<1;      //geser ke kiri 1 bit
   cout << “nilai x kini =  “ << x <<‘\n’;
}
```

Soal

1. Buat program penggunaan operator aritmatika untuk menentukan nilai dari: d = b*b-4ac
2. Buat program untuk menampilkan kode ASCII dari suatu karakter melalui konversi tipe

Jawaban

```
#include<iostream.h>
#include<conio.h>
void main()
{
   char kar='A';
   clrscr();
   cout<<"Nilai ASCII dari "<<kar;
   <<"yaitu"<<int(kar)<<endl;
}
```