PERTEMUAN 1
TIPE DATA, VARIABEL, DAN KONSTANTA

A. TUJUAN PEMBELAJARAN

Setelah menyelesaikan pertemuan ini, mahasiswa mampu mempraktekkan:

1. Pendahuluan
2. Tipe Data dan Variabel
3. Konstanta

B. URAIAN MATERI

1. Pendahuluan

Saat melakukan pemrograman dalam bahasa pemrograman apa pun, Anda perlu menggunakan berbagai variabel untuk menyimpan berbagai informasi. Variabel hanyalah lokasi memori yang dicadangkan untuk menyimpan nilai. Ini berarti bahwa ketika Anda membuat variabel, Anda menyediakan beberapa ruang di memori.

Anda mungkin ingin menyimpan informasi dari berbagai tipe data seperti char, widechar, integer, floatingpoint, doublefloatingpoint, boolean dll. Berdasarkan tipe data variabel, sistem operasi mengalokasikan memori dan memutuskan apa yang dapat disimpan di memori cadangan.

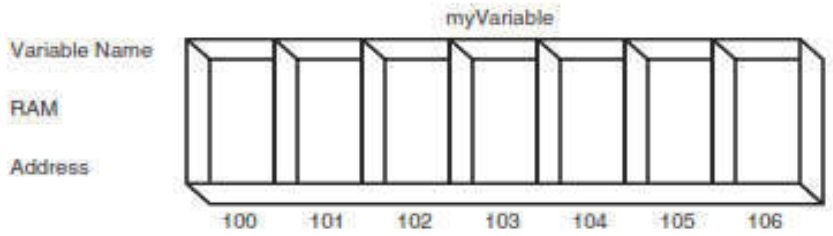
2. Tipe Data dan Variabel

a. Apa itu variabel

Secara Umum (missal Dari sudut pandang programmer, variabel adalah lokasi di memori komputer Anda di mana Anda dapat menyimpan nilai dan kemudian Anda dapat mengambil nilai itu.

Untuk memahami hal ini, Anda harus terlebih dahulu memahami sedikit tentang cara kerja memori komputer. Memori komputer Anda dapat dianggap sebagai serangkaian lubang kecil, semuanya berbaris dalam baris panjang. Setiap lubang kubus — atau lokasi memori — diberi nomor secara berurutan. Angka-angka ini dikenal sebagai alamat memori.

Variabel tidak hanya memiliki alamat, mereka memiliki nama. Misalnya, Anda dapat membuat variabel bernama myAge. Variabel Anda adalah label di salah satu lubang kecil ini sehingga Anda dapat menemukannya dengan mudah, tanpa mengetahui alamat memori sebenarnya.



Gambar 1.1 representasi dari memori

b. Pemberian Nama Variabel

Nama Variabel ditentukan atau dikarang sendiri oleh pembuat program dengan syarat sebagai berikut :

1) Tidak boleh sama dengan nama atau kata yang sudah disiapkan oleh kmputer (*reserved wrd*) seperti *keyword*, dan *Function.* Juga harus berbeda dengan nama label dan konstanta yang dibuat oleh pemrogram.
2) Maksimum 32 karakter, bila lebih dari 32 karakter, maka karakter selebihnya tidak diperhatikan oleh computer. Huruf besar dan huruf kecil berbeda .
3) Tidak bi=oleh mengandung spasi atau blank.
4) Karakter pertama harus huruf atau karakter garis bawah (*Under Score*) dan karakter berikutnya boleh huruf atau angka atau karakter garis bawah.

c. Mengatur Selain Memori

Saat Anda mendefinisikan variabel di C ++, Anda harus memberi tahu compiler tidak hanya namanya, tetapi juga jenis informasi apa yang akan disimpannya: integer, char, dan sebagainya. Ini adalah jenis variabel. Dengan kata lain untuk tipe yang mungkin Anda lihat adalah tipe data.

Jenis variabel memberi tahu kompiler berapa banyak ruang yang harus disisihkan dalam memori untuk menampung nilai variabel.

Setiap kubus berukuran satu byte. Jika jenis variabel yang Anda buat berukuran dua byte, itu membutuhkan dua byte memori, atau dua kubus. Jenis variabel (misalnya, int) memberi tahu kompiler berapa banyak memori (berapa banyak lubang kubus) yang harus disisihkan untuk variabel tersebut. Karena komputer menggunakan bit dan byte untuk merepresentasikan nilai, dan karena memori diukur dalam byte, penting bagi Anda untuk memahami dan terbiasa dengan konsep ini.

d. Ukuran Integer

Variabel char (digunakan untuk menampung karakter) paling sering satu byte. Int pendek adalah dua byte pada kebanyakan komputer; int panjang biasanya empat byte, dan int (tanpa kata kunci pendek atau panjang) bisa dua atau empat byte. Jika Anda menjalankan Windows 95/98, Windows XP, atau Windows NT / 2000/2003, Anda dapat mengandalkan int Anda menjadi empat byte selama Anda menggunakan kompiler modern.

Tetapi Anda tidak boleh mengandalkan kebenaran itu di sistem Anda — periksa. Anda yakin bahwa ukuran sebuah int short akan lebih kecil dari atau sama dengan ukuran sebuah int, dan bahwa ukuran sebuah int akan sama atau lebih kecil dari ukuran sebuah int long.

e. Signed dan Unsigned

Selain itu, semua jenis ini hadir dalam dua jenis: Unsigned. Terkadang Anda membutuhkan angka negatif, dan terkadang tidak. Bilangan bulat (short dan long) tanpa kata "unsigned" diasumsikan signed. Signed integer baik negatif atau positif. bilangan bulat unsigned selalu positif. Ingat: tanda tangan adalah default untuk variabel integer.

Karena Anda memiliki jumlah byte yang sama (dan karenanya jumlah bit) untuk integer signed dan unsigned, bilangan terbesar yang dapat Anda simpan dalam integer unsigned adalah dua kali lebih besar dari bilangan positif terbesar yang dapat Anda simpan dalam integer signed. Integer short unsigned dapat menangani angka dari 0 hingga 65.535. Separuh

angka yang diwakili oleh signed adalah negatif, sehingga signed short hanya dapat mewakili angka dari –32.768 hingga 32.767.

f. Jenis Variabel Fundamental

Beberapa jenis variabel lainnya dibangun ke dalam C ++. Mereka dapat dengan mudah dibagi menjadi variabel integer (jenis yang dibahas sejauh ini), variabel floating-point, dan variabel char.

Variabel floating-point memiliki nilai yang dapat diekspresikan sebagai pecahan — yaitu, bilangan real. Variabel karakter memiliki satu byte dan digunakan untuk menampung 256 karakter dan simbol dari kumpulan karakter ASCII dan ASCII yang diperluas.

Kumpulan karakter ASCII adalah kumpulan karakter yang distandarisasi untuk digunakan di komputer. ASCII adalah singkatan dari American Standard Code for Information Interchange. Hampir setiap sistem operasi komputer mendukung ASCII, meskipun banyak yang mendukung rangkaian karakter internasional lainnya juga.

Jenis variabel yang digunakan dalam program C ++ dijelaskan pada Tabel 2.1. Tabel ini menunjukkan jenis variabel, berapa banyak ruang yang diasumsikan buku ini dalam memori, dan jenis nilai apa yang dapat disimpan dalam variabel ini.

Tabel 1.1tipe Variabel

Type	Size	Values
unsigned short int	2 bytes	0 to 65,535
short int	2 bytes	–32,768 to 32,767
unsigned long int	4 bytes	0 to 4,294,967,295
long int	4 bytes	–2,147,483,648 to 2,147,483,647
int	4 bytes	–2,147,483,648 to 2,147,483,647
unsigned int	4 bytes	0 to 4,294,967,295
char	1 byte	256 character values
bool	1 byte	true or false
float	4 bytes	1.2e–38 to 3.4e38
double	8 bytes	2.2e–308 to 1.8e308

g. Mendefinisikan Variabel

Anda membuat, atau mendefinisikan, variabel dengan menyatakan tipenya, diikuti dengan satu atau beberapa spasi, diikuti dengan nama

variabel dan titik koma. Nama variabel dapat berupa kombinasi huruf apa saja, tetapi tidak boleh berisi spasi. Nama variabel legal termasuk x, J23qrsnf, dan myAge. Nama variabel yang baik memberi tahu Anda untuk apa variabel itu; menggunakan nama yang baik memudahkan untuk memahami alur program Anda. Pernyataan berikut mendefinisikan variabel integer yang disebut myAge:

```
int myAge;
```

Ingat bahwa C ++ peka huruf besar / kecil, jadi myAge adalah nama variabel yang berbeda dari MyAge. Sebagai praktik pemrograman umum, coba gunakan nama yang memberi tahu Anda untuk apa variabel itu. Nama-nama seperti myAge atau howMany jauh lebih mudah dipahami dan diingat daripada nama-nama seperti xJ4 atau theInt. Jika Anda menggunakan nama variabel yang bagus, Anda akan membutuhkan lebih sedikit komentar untuk memahami kode Anda.

Coba eksperimen ini: Tebak apa yang dilakukan bagian-bagian program ini, berdasarkan beberapa baris kode pertama:

```
#include<stdio.h>
int main()
{
unsigned short x;
unsigned short y;
unsigned int z;
x=65536;
y=65535;
 z = x * y;
  printf ("x = %u \n ",x);
 printf ("y = %u \n ",y);
 printf ("z = %u \n ",z);
}
```

Lengkapi bagian- bagian source dibawah ini

```
Contoh 2
main ()
{
unsignedshortWidth;
unsignedshortLength;
unsignedshort Area;
Area = Width * Length;
}
```

h. Sensitivitas huruf

C ++ peka terhadap huruf besar / kecil. Dengan kata lain, huruf besar dan kecil dianggap berbeda. Variabel bernama age berbeda dengan Age, yang berbeda dengan AGE.

i. Keyword

Beberapa kata disediakan oleh C ++, dan Anda tidak boleh menggunakannya sebagai nama variabel. Ini adalah keyword yang digunakan oleh compiler untuk mengontrol program Anda. Keyword mencakup if, while, for, dan main. Namun secara umum, nama yang masuk akal untuk variabel hampir pasti bukan kata kunci.

Variabel Anda mungkin berisi keyword sebagai bagian dari namanya tetapi tidak seluruh nama. Variabel seperti main_flag dan forEver benar-benar legal sedangkan main dan for tidak akan legal.

j. Membuat Lebih dari Satu Variabel Sekaligus

Anda dapat membuat lebih dari satu variabel berjenis sama dalam satu pernyataan dengan menulis tipe dan kemudian nama variabel, dipisahkan dengan koma. Sebagai contoh:

```
unsigned int myAge, myWeight; // dua variabel unsigned int
long area, width, length; // tiga tipe data long
```

Seperti yang Anda lihat, myAge dan myWeight masing-masing dideklarasikan sebagai variabel integer unsigned. Baris kedua mendeklarasikan tiga variabel panjang individu bernama area, width, dan length. Tipe (long) ditetapkan ke semua variabel, jadi Anda tidak dapat mencampur tipe dalam satu pernyataan definisi.

k. Menetapkan nilai ke variabel

Anda menetapkan nilai ke variabel dengan menggunakan operator (=). Jadi, Anda akan menetapkan 5 ke width dengan menulis

```
unsignedshortWidth; Width = 5;
```

Anda dapat menggabungkan langkah-langkah ini dan menginisialisasi Width saat Anda menentukannya dengan menulis

```
unsignedshortWidth = 5;
```

Inisialisasi terlihat sangat mirip dengan assignment, dan dengan variabel integer, perbedaannya kecil. Nanti, ketika konstanta tercakup, Anda akan melihat bahwa beberapa nilai harus diinisialisasi karena tidak dapat diberi nilai.

```
Contoh
0: // Demonstrationofvariables
#include <iostream>
#include <cstdlib>
```

```
int main()
{
unsignedshort int Width = 5, Length;
Length = 10;

// membuat sebuahshortunsigned dan inisialisasi dengan hasil
// dari mengalikan Width dengan Length
unsignedshort int Area = Width * Length;
std::cout<< “Width: “ <<Width<< “\n”;
std::cout<< “Length: “ <<Length<<std::endl;
std::cout<< “Area: “ << Area <<std::endl;
system (“PAUSE”) ;
return 0;
}
```

l. Typedef

Ini bisa menjadi membosankan, berulang, dan, yang paling penting, rawan kesalahan untuk terus menulis unsigned short int. Anda dapat membuat sinonim untuk tipe yang sudah ada dengan menggunakan kata kunci typedef, yang merupakan singkatan dari definisi tipe.

typedef digunakan dengan menulis kata kunci typedef diikuti dengan tipe yang ada dan kemudian nama baru.Sebagai contoh,

typedefunsignedshort int USHORT

membuat nama baru USHORT yang dapat Anda gunakan di mana pun, Anda mungkin telah menulis short int unsigned.

```
Contoh
// *****************
 // Demonstratestypedefkeyword
#include
typedefunsignedshort int USHORT; //typedefdefined
 int main()
{
USHORT Width = 5;
USHORT Length;
Length = 10;
USHORT Area = Width * Length;
std::cout<< “Width: “ <<Width<< “\n”;
std::cout<< “Length: “ <<Length<<std::endl;
std::cout<< “Area: “ << Area <<std::endl;
return 0;
 }
```

m. Kapan Menggunakan short dan Kapan Menggunakan long

Salah satu sumber kebingungan bagi programmer C ++ baru adalah kapan mendeklarasikan variabel menjadi tipe long dan kapan mendeklarasikannya menjadi tipe short. Aturannya, jika dipahami, cukup mudah: Jika ada kemungkinan nilai yang ingin Anda masukkan ke dalam variabel akan terlalu besar untuk tipenya, gunakan tipe yang lebih besar.

Seperti yang terlihat pada Tabel 1.1,unsigned short integer, dengan asumsi bahwa keduanya adalah dua byte, hanya dapat menyimpan nilai hingga 65.535. Signed short integers hanya dapat menampung setengahnya. Meskipun unsigned long integers dapat menampung angka

yang sangat besar (4.294.967.295), itu masih cukup terbatas. Jika Anda membutuhkan angka yang lebih besar, Anda harus pergi ke float atau double, dan kemudian Anda kehilangan beberapa presisi. Float dan double dapat menampung angka yang sangat besar, tetapi hanya 7 atau 19 digit pertama yang signifikan di sebagian besar komputer. Artinya, angka tersebut dibulatkan setelah banyak digit.

n. WrappingAround in unsignedIntegers

Fakta bahwa unsigned long integersmemiliki batas nilai yang dapat mereka pegang jarang menjadi masalah, tetapi apa yang terjadi jika Anda kehabisan ruangan?

Ketika sebuah unsigned integer mencapai nilai maksimumnya, ia membungkus dan memulai kembali, seperti odometer mobil. Contoh dibawah ini akan menunjukkan apa yang terjadi jika Anda mencoba memasukkan nilai yang terlalu besar ke dalam short integer.

```
Contoh
#include
 int main()
{
unsignedshort int smallNumber;
smallNumber = 65535;
std::cout<< “smallnumber:” <<smallNumber<<std::endl;
smallNumber++;
std::cout<< “smallnumber:” <<smallNumber<<std::endl;
smallNumber++;
std::cout<< “smallnumber:” <<smallNumber<<std::endl;
return 0;
}
```

o. WrappingAround a signed Integer

signed integer berbeda dari unsigned integer di mana separuh nilainya negatif. Alih-alih membayangkan odometer mobil tradisional, Anda dapat membayangkan odometer yang berputar ke atas untuk bilangan positif dan ke bawah untuk bilangan negatif. Satu mil dari nol bisa jadi 1 atau –1. Ketika Anda kehabisan bilangan positif, Anda langsung menemukan bilangan negatif terbesar dan kemudian menghitung mundur ke nol.

Contoh dibawah ini akan menunjukkan apa yang terjadi ketika Anda menambahkan 1 ke bilangan positif maksimum dalam bilangan unsigned short integer.

```
Contoh
#include


int main()
{
short int smallNumber;
smallNumber = 32767;
std::cout<< “smallnumber:” <<smallNumber<<std::endl;
smallNumber++;
std::cout<< “smallnumber:” <<smallNumber<<std::endl;
smallNumber++;
std::cout<< “smallnumber:” <<smallNumber<<std::endl;  return 0;
}
```

3. Konstanta

Seperti variabel, konstanta adalah lokasi penyimpanan data. Tetapi variabel dapat bervariasi, sedangkan konstanta seperti yang mungkin sudah anda sudah duga, tidak bervariasi.

Anda harus menginisialisasi sebuah konstanta saat Anda membuatnya, dan Anda tidak dapat menetapkan nilai baru nanti setelah sebuah konstanta diinisialisasi.

C ++ memiliki dua jenis konstanta: literal dan simbolik.

a. Konstanta Literal

Konstanta literal adalah nilai yang diketik langsung ke program Anda di mana pun diperlukan. Sebagai contoh:

```
int myAge = 39;
```

myAge adalah sebuah variabel, bertipe int, 39 adalah konstanta literal. Anda tidak dapat menetapkan nilai ke 39, dan nilainya tidak dapat diubah.

b. Konstanta Simbolis

Konstanta simbolis adalah konstanta yang diwakili oleh sebuah nama, seperti halnya variabel. Tidak seperti variabel, bagaimanapun, setelah konstanta diinisialisasi, nilainya tidak dapat diubah. Jika program Anda memiliki satu variabel integer bernama students dan nama class lain, Anda dapat menghitung berapa banyak siswa yang Anda miliki, berdasarkan jumlah kelas yang diketahui, jika Anda mengetahui ada 15 siswa per kelas:

```
students = classes * 15;
```

Dalam contoh ini, 15 adalah konstanta literal. Kode Anda akan lebih mudah dibaca dan lebih mudah dipertahankan jika Anda mengganti konstanta simbolis untuk nilai ini:

```
students = classes * studentsPerClass
```

Jika nanti Anda memutuskan untuk mengubah jumlah siswa di setiap class, Anda dapat melakukannya di mana Anda menentukan studentsPerClass konstan tanpa harus membuat perubahan di setiap tempat Anda saat menggunakan nilai itu. Risikonya adalah Anda bisa melewatkan salah satu tempat di mana Anda menggunakan nilai dan mengalami kesulitan untuk men-debug kesalahan logika.

c. Mendefinisikan Konstanta dengan #define

Untuk mendefinisikan cara yang kuno, jahat, dan tidak benar secara politis, Anda akan memasukkan:

```
#define studentsPerClass 15
```

Begitulah cara melakukannya di versi lama bahasa C, standar ANSI memperkenalkan kata kunci const ke bahasa tersebut. Perhatikan bahwa studentsPerClass tidak berjenis tertentu (int, char, dan seterusnya). #define melakukan substitusi teks sederhana. Setiap kali preprocessor melihat kata studentsPerClass, ia menempatkan 15 di teks.

d. Mendefinisikan Konstanta dengan konst

Meskipun #define berfungsi, ada cara yang lebih baik, tidak terlalu menggemukkan, dan lebih menarik untuk menentukan konstanta di C ++:

```
constunsignedshort int studentsPerClass = 15;
```

Contoh ini juga mendeklarasikan konstanta simbolis bernama studentsPerClass, tapi kali ini studentsPerClass diketik sebagaiunsigned short int. ini membutuhkan waktu lebih lama untuk mengetik, tetapi menawarkan beberapa keuntungan. Perbedaan terbesar adalah konstanta ini memiliki tipe, dan compiler dapat memaksakannya untuk digunakan sesuai dengan tipenya.

e. Konstanta Enumerated

Konstanta yang disebutkan membuat sekumpulan konstanta. Misalnya,Anda dapat mendeklarasikan COLOR sebagai enumeration; dan Anda dapat menentukan bahwa ada lima nilai untuk COLOR: RED, BLUE, GREEN, WHITE, and BLACK.

Sintaks untuk konstanta yang disebutkan adalah dengan menulis kata kunci enum, diikuti dengan nama jenis, tanda kurung kurawal buka, masing-masing nilai dipisahkan oleh koma, dan diakhiri tanda kurung tutup kurawal dan titik koma. Berikut contohnya:

```
enum COLOR { RED, BLUE, GREEN, WHITE, BLACK };
```

Pernyataan ini melakukan dua tugas:

1) Itu membuat COLOR menjadi nama enumerasi, yaitu tipe baru
2) Itu membuat RED menjadi konstanta simbolis dengan nilai 0, BLUE konstanta simbolis dengan nilai 1, GREEN sebagai konstanta simbolis dengan nilai 2, dan seterusnya.

Setiap konstanta yang disebutkan memiliki nilai integer. Jika Anda tidak menentukan sebaliknya, konstanta pertama akan bernilai 0, dan sisanya akan dihitung dari sana. Namun, salah satu konstanta dapat diinisialisasi dengan nilai tertentu, dan konstanta yang tidak diinisialisasi akan dihitung ke atas dari yang sebelumnya. Jadi, jika Anda menulis:

```
enumColor { RED=100, BLUE, GREEN=500, WHITE, BLACK=700 };
```

maka RED akan memiliki nilai 100, BLUE, nilai 101; GREEN, nilainya 500; WHITE, nilainya 501; dan BLACK, nilainya 700.

Keuntungan dari teknik ini adalah Anda bisa menggunakan nama simbolis seperti BLACK atau WHITE alih-alih angka yang mungkin tidak berarti seperti 1 atau 700.

Merupakan praktik umum untuk menggunakan semua huruf kapital untuk nama konstanta reguler dan enumerasi. Ini membuatnya jelas secara visual bahwa nama yang Anda baca adalah konstanta dan bukan variabel yang boleh Anda ubah. Compiler itu sendiri tidak peduli selama anda konsisten.

C. SOAL LATIHAN/TUGAS

1. Apa itu Variabel?
2. Apa perbedaan dari initialization dan defining untuk sebuah variabel?
3. Apa perbedaan antara sebuah #define constant dan const?
4. Apa yang dimaksud konstanta literal dan simbolis?
5. Apakah variabel DOG, dog, Dog, doG sama? Jelaskan!

D. REFERENSI

Soulié, Juan. 2007. *C++ Language Tutorial, Juni 2007. Diambil dari :*

https://www.cplusplus.com/files/tutorial.pdf (6 November 2020)

Moh.Sjukani . 2014. ALGORITMA (Algoritma & Struktur Data 1) dengan *C, C++ dan Java, Edisi 9, Mitra Wacana Media.*