**STRUKTUR DATA**

**Riya Widayanti**

STRUKTUR DATA Adalah

Metode untuk mengorganisasikan data di dalam memori komputer, sehingga data dapat diolah secara efisien, yaitu:

1. Menggunakan tempat yang sehemat mungkin di memori
2. Menggunakan waktu yang secepat mungkin untuk menyimpan dan mengambil data
3. Seringkali dua keadaan di atas tidak bisa dicapai sekaligus, tetapi harus dicapai suatu kompromi dari dua keadaan di atas dan untuk itulah Struktur Data dipelajari.

TIPE DATA

Mendeklarasikan tipe data adalah suatu cara untuk menetapkan:

1. Informasi apa yang terkait dengan data

2. Ukuran memori yang digunakan oleh data

3. Operasi apa yang dapat dilakukan terhadap data

CONTOH:

int x; char c; float y;

**Var x**: menempati 2-byte memori, informasi yang terkait adalah data integer(bilangan bulat), operasi yang dapat dilakukan: kali, bagi, tambah, kurang dan sisa hasil bagi.

 Rentang nilai bilangan: -3278 s/d 32767

**Var c:** menempati 1-byte memori informasi yang terkait adalah kata karakter (tabel ASCII), operasi yang dilakukan: menyambung, menyalin karakter (concatenation, copy, etc)

Var y: menempati 4-byte memori informasi yang terkait adalah data float (bilangan pecahan), operasi yang dapat dilakukan: kali, bagi, tambah, kurang.

 Rentang nilai bilangan bulat: -3.4x10-38 s/d 3.4x10-38

KELAS TIPE DATA

Tipe Data Atomik(int, char, float)

Tipe Data Terstruktur(array, struct)

Abstraksi:

Suatu proses yang sengaja mengurangi rincian dari suatu objek, dengan harapan dapat memahami objek tersebut dengan baik atau lebih nudah.

Tingkatan abstraksi sesuai dengan seberapa besar rincian-rincian dihilangkan. Tingkatan abstraksi dapat berbeda, tergantung dari sudut pandang seseorang terhadap suatu objek.

Contoh:

Abstraksi terhadap objek mobil:

1. Bagi pemilik, mobil adalah:

 - Merk/tipe/tahun mobil

 - Warna

2. Bagi Sopir, mobil adalah:

 - Stir (kemudi)

 - Pedal gas, kompling dan rem

3. Bagi montir, mobil adalah:

 - Mesin, karburator, pengapian, distributor, dsb.

Tipe Data Abstrak: Timbul dari hasil Imajinasi

Tipe Data Virtual: Terdapat dalam suatu bahasa pemrograman (virtual prosesor)

Tipe data Fisik: Secara fisik dalam memori komputer

Contoh:

|  |  |  |
| --- | --- | --- |
| **Tipe Data** | **Atomik** | **Terstruktur** |
| Abstrak | Jml mahasiswa | Tabel Suhu |
| Virtual | Integer | Array |

Keuntungan menggunakan TDA:

 1. Modularitas

 2. Penyembunyian Informasi

 3. Kebebasan Pelaksanaan

 4. Integritas Data

 5. Penyederhanaan Masalah

Langkah-langkah menterjemahkan TDA ke Virtual:

1. Pilih bahasa pemrograman yang akan digunakan: C/C++, Pascal
2. Pilih cara untuk merepresentasikan data dalam bentuk: Array, struct, record, set, dsb.
3. Tulis program untuk implementasi

JENIS-JENIS STRUKTUR DATA:

1. Linier: Struktur yang setiap elemen mempunyai hubungan satu-satu (one to one) dengan elemen lainnya
2. Tree/Hirarki:Struktur yang setiap elemennya mempunyai hubungan satu-banyak (one to many) dengan elemen lainnya.
3. Graph/Jaringan

Struktur yang setiap elemen mempunyai hubungan banyak-banyak (many to many) dengan elemen lainnya.

**ARRAY**

**Pengertian Array**

**Array** adalah kumpulan data bertipe sama yang menggunakan nama sama.

Antara satu variabel dengan variabel yang lain di dalam array dibedakan berdasarkan **subscript**, yang dituliskan dalam kurung siku atau yang sering disebut dengan **indeks array.**

Berikut ini contoh variabel bernama c yang mempunyai 7 lokasi memori yang semuanya bertipe int:

|  |  |
| --- | --- |
| C[0] | 10 |
| C[1] | 20 |
| C[2] | 30 |
| C[3] | 40 |
| C[4] | 75 |
| C[5] | 80 |
| C[6] | 12 |

**Deklarasi Array**

Array dideklarasikan dengan mencantumkan tipe dan nama variabel yang diikuti dengan banyaknya lokasi memori yang ingin dibuat, dimana nomor indeks dimulai dari nol (0).

**int c[7];**

* 1. Tipe data elemen array
	2. Nama array
	3. Jumlah elemen array

Setalah didefinisikan elemen array dapat diakses dalam bentuk:

nama\_array[indeks]

Array dapat diberikan nilai awal(inisialisasi) pada saat pendefinisian.

**Operasi Terhadap Larik**

Larik dapat digunakan :

* Untuk memuat informasi awal ke dalamnya, yang kemudian akan digunakan dalam pemrosesan
* Untuk memproses elemen-elemen dari larik
* Untuk menyimpan hasil pemrosesan ke dalam elemen dari larik
* Untuk menyimpan informasi yang kemudian ditulis pada sebuah laporan

Operasi yang paling umum yang dilakukan terhadap sebuah larik adalah :

* Memuat sekumpulan nilai awal ke dalam elemen sebuah larik
* Pemrosesan elemen sebuah larik
* Penelusuran sebuah larik, menggunakan pencarian linier atau biner untuk sebuah elemen tertentu
* Menuliskan isi dari sebuah larik ke sebuah laporan

**Pemberian nilai pada Variabel Array dalam notasi pemrograman C++**

Variabel pada array diberi nilai dengan berbagai cara:

Cara Pertama:

int a[5] ={4,5,8,9,12};

Cara Kedua:

int a[0] = 4;

int a[1] = 5;

int a[2] = 8;

int a[3] = 9;

int a[4] = 12

Cara Ketiga: (memanfaatkan struktur pemrograman REPEATITION, boleh memakai FOR, While, Do...)

for (i=0;i<10;i++)

{

 cin>>a[i];

}

**Array Dimensi Satu**

**Pemberian nilai variabel array dengan sebuah proses aritmatika**

Contoh pengisian **array b** dengan mengkalkulasi 3 kali **array a**

for (i=0;i<10;i++)

{

 cin>>a[i];

 b[i] = 3 \* a[i];

}

**Atau**

for (i=0;i<10;i++)

{

 cin>>a[i];

 if (a[i] % 2 == 0)

 b[i] ++;

 else

 c[i] ++;

}

**Kasus 10.1**

Buatlah algoritma dan program untuk menghitung jumlah setiap elemen dalam suatu array.

Algoritma untuk kasus di atas adalah:

1. Tentukan elemen array sebanyak yang diinginkan
2. Tentukan nilai awal indeks, batas akhir indeks dan kenaikannnya (dalam hal ini nilai awal indeks = 0, batas akhir indeks adalah jumah elemen array di atas yaitu 12 dikurangi dengan 1, dan kenaikknannya adalah 1)
3. Lakukan perulangan sesuai dengan langkah 2
4. Lakukan penjumlahan masing-masing elemen array sampai batas akhir indeks terpenuhi.
5. Tampilkan penjumlahan semua elemen array.
6. Selesai

Program untuk kasus di atas:

#include <iostream.h>

#include <conio.h>

void main()

{

 int a[12] ={1,3,5,4,7,2,99,16,45,67,89,45};

 int total=0;

 for (int i = 0; i<= 12;i++)

 total +=a[i];

 cout<<"Total seluruh elemen array adalah :"<<total;

}

**Kasus 10.2**

Buatlah program untuk membuat inputan 5 suhu dengan keluaran sebagai berikut:

 Masukkan 5 data suhu:

 Suhu[1] : …

 Suhu[2] : …

Suhu[3] : …

Suhu[4] : …

Suhu[5] : …

Program untuk kasus di atas adalah:

#include <iostream.h>

void main()

{

 **float suhu[5];**

 cout<<"Masukan 5 data suhu"<<endl;

 for (int i= 0; i< 5; i++)

 {

 cout<<"Suhu ["<<i+1<<"]:";

 cin>>suhu[i];

 }

}

Hasil program di atas bila adalah:



Lakukan pengembangan dari kasus 10.2 dengan inputan suhu di atas dengan mencari rata-rata, maksimum dan minimum.

**Kasus 10.3**

#include <iostream.h>

#include <iomanip.h>

void main()

{

 **int i, n[10];**

 cout<<"Masukan 5 data suhu"<<endl;

 for (i= 0; i< 10; i++)

 n[i]=0;

 cout<<"Elemen"<<setw(13)<<"Value"<<endl;

 for (i= 0; i< 10; i++)

 cout<<setw(5)<<i<<setw(12)<<n[i]<<endl;

}

Hasil program di atas bila adalah:



**Kasus 10.4**

Buatlah algoritma dan program untuk menampilkan bilangan 1 sampai bilangan 10, dengan pangkatnya masing-masing. Adupun batas nilai maksimal yang disimpan adalah 100.

 Bilangan Pangkat

1. 1
2. 4
3. 9
4. 16
5. 25
6. 36
7. 49
8. 64
9. 81
10. 100

Algoritma untuk permasalahan di atas adalah di atas adalah sebagai berikut:

1. Tentukan elemen array untuk menampung nilai perkalian
2. Tentukan nilai awal indeks, batas akhir indeks dan kenaikannya (dalam hal ini, nilai awal indeks adalah 0, batas akhir indeks adalah 10, dan kenaikkannya adalah 1)
3. Lakukan perulangan sesuai dengan langkah 2
4. Nilali awal indeks ditambah dengna 1
5. Lakukan perkalian masing-masing elemen array sampai batas akhir indeks terpenuhi.
6. Tampilkan perkalian semua elemen arary.
7. Selesai

Program untuk kasus di atas adalah

#include <iostream.h>

#include <iomanip.h>

void main()

{

 int pangkat[10], k;

 cout<<"Bilangan"<<setw(13)<<"Pangkat"<<endl;

 for (int i= 0; i< 10; i++)

 {

 k= i+1;

 pangkat[i]=k\*k;

 cout<<setw(7)<<k<<setw(12)<<pangkat[i]<<endl;

 }

}

Hasil program di atas bila adalah:

****

**Array Dimensi Dua**

C++ menyediakan array dimensi dua, contoh data mahasiswa baru di fakultas Teknik tahun 2001-2004

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Jurusan | 2001 | 2002 | 2003 | 2004 |
| Teknik Industri | 53 | 70 | 65 | 60 |
| Teknik Planalogi | 10 | 13 | 21 | 15 |
| Teknik Informatika | 86 | 132 | 242 | 195 |

Contoh di atas merupakan matrik L berukuran 3 X 4 dapat dideklarasikan dalam C++ sebagai berikut: {[53,70,65,60],[10,13,21,15],[86,132,242,195]} dan definisi variabel utnuk setiap elemen tersebut adalah:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 |
| 0 | L[0,0] | L[0,1] | L[0,2] | L[0,3] |
| 1 | L[1,0] | L[1,1] | L[1,2] | L[1,3] |
| 2 | L[2,0] | L[2,1] | L[2,2] | L[2,3] |

Kasus 10.5

#include <iostream.h>

#include <conio.h>

void main()

{

**int l[3][4];**

int tahun, jurusan;

clrscr;

l[0][0] = 53;

l[0][1] = 70;

l[0][2] = 65;

l[0][3] = 60;

l[1][0] = 10;

l[1][1] = 13;

l[1][2] = 21;

l[1][3] = 15;

l[2][0] = 86;

l[2][1] = 132;

l[2][2] = 242;

l[2][3] = 195;

 while (1)

 {

 cout<<"Jurusan (0=T.Industri, 1=T.Planalogi, 2=T.Informatika)";

 cout<<"\nMasukan Jurusan =";

 cin>>jurusan;

 if ((jurusan == 0) || (jurusan ==1) ||(jurusan ==2))

 break;

 }

 while (1)

 {

 cout<<"Tahun (2001-2004) =";

 cin>>tahun;

 if ((tahun>=2001) && (tahun <= 2004))

 {

 tahun -=2001;

 break;

 }

}

cout<<"Julah Mahasiswa yang masuk = "<<l[jurusan][tahun]<<endl;

}

Hasil program di atas bila adalah:

****

**Latihan**

1. Diketahui data-data sebagai berikut:

|  |  |
| --- | --- |
| Nama | Nilai Ujian |
| Andi | 84 |
| Melissa | 76 |
| Herman | 54 |
| Joko | 46 |
| Wayan | 76 |

Buatlah algoritma dan program untuk membaca nilai-nilai tersebut dan mencari siapa sajakah yang lulus ujian. Syarat lulus ujian bahwa nilai ujian>=70.

2. Diketahui data-data sebagai berikut:

**Data Nilai Siswa**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nama | Nilai Fisika | Nilai Matematika | Nilai Kimia | Nilai Biologi |
| Andi | 84 | 84 | 94 | 64 |
| Melissa | 76 | 66 | 76 | 86 |
| Herman | 54 | 84 | 54 | 84 |
| Joko | 46 | 66 | 86 | 76 |
| Wayan | 76 | 66 | 66 | 76 |

Buatlah algoritma dan pemrograman untuk menghitung nilai rata-rata tiap siswa dan nilai rata-rata tiap matakuliah.

**Pendahuluan**

Program komputer yang dibuat untuk menjawab permasalahan umumnya berukuran besar. Seringkali dalam membuat program besar tersebut adalah memecah program menjadi potonga-potongan program yang kecil yang dinamakan modul. Teknik pemrograman seperti ini dinamakan teknik pemrograman modular. Beberapa bahasa pemrogramn menamakan modul dengan sebutan rutin(routine), prosedure atau fingsi.

Modularisasi program mempunyai dua keuntungan:

1. Untuk aktifitas yang harus dilakukan lebih dari satu kali, modularisasi menghindari penulisan teks program yang sama berulang kali. Modul tersebut cukup dituliskan sekali saja, lalu modul tersebut dapat diakses(diistilahkan dengan dipanggil dari bagian lain di dalam program). Hal ini bermanfaat bila ingin menghemat ukuran program.
2. Kemudahan menulis dan menemukan kesalahan (debug) program. Hal ini sangat berguna pada masalah yang besar.

Modul program di dalam C++ disebut fungsi (function)

**Dasar Fungsi**

Pada umumnya fungsi memerlukan masukan yang dinamakan argumen atau parameter. Hasil akhir fungsi akan berupa sebuah nilai(nilai balik fungsi).

Konsep Program dibagi menjadi sejumlah modul



Bentuk umum fungsi:

Penentu-tipe nama fungsi (daftar parameter)

Dekalarasi parameter

{

isi fungsi

}

Penentu tipe fungsi untuk menentukan tipe keluaran fungsi yang dapat berupa salah satu tipe data C++ yang berlaku, misal char atau int

Sebuah contoh program yang menggunakan fungsi:

**Kasus 9.1**

#include <iostream.h>

#include <conio.h>

void garis(); /\*Prototipe fungsi garis()\*/

void main()

{

clrscr(); /\*Hapus layar\*/

garis(); /\*panggil fungsi garis()\*/

cout<< "Selamat Belajar Algoritma dan Pemrograman dengan C++ \n";

cout<< "Yakinlah Anda Pasti Bisa \n";

garis();

cout<< "Program C++ memang sangat mudah \n";

garis();

}

void garis() /\*Definisi fungsi garis()\*/

{

int i;

for (i=0;i<49;i++)

cout<<"=";

cout<<endl;

}

Hasil Program tersebut adalah:



**Jangkauan sebuah variabel**

Jangkauan dari sebuah variabel adalah bagian dari sebuah program dimana variabel didefinisikan dan dimana variabel dapat dereferensikan. Apabila sebuah daftar dibuat dari semua modul dimana sebuah variabel dapat direferensikan, daftar tersebut mendefinisikan jangkauan dari variabel. Variabel dapat berupa variabel global, dimana jangkauan variabel adalah keseluruhan program, dan variabel lokal dimana jangkauan variabel hanya modul dimana dia didefinisikan.

**Data global**

Ketika informasi atau data menggunakan nama variabel yang sama pada kedua modul pemanggil maupun modul yang dipanggil dalam sebuah program, data tersebut dikenal sebagai data global, dan jangkauan dari variabel adalah keseluruhan program karena data dapat diakses oleh setiap modul dalam program. Akan tetapi semua data tidak harus menjadi global.

**Data lokal**

Variabel yang didefinisikan dalam sebuah subordinat modul disebut dengan variabel lokal. Variabel-variabel lokal ini tidak dikenal oleh modul pemanggil maupun modul-modul yang lain.

**Efek samping**

Sebuah efek samping adalah sebuah bentuk komunikasi silang sebuah modul dengan bagian lain dari sebuah program. Efek samping muncul ketika sebuah modul subordinat mengubah nilai variabel global dalam sebuah modul. Efek samping tidak perlu mengganggu. Akan tetapi cenderung mengurangi kemudahan sebuah program dikelola dan kita perlu berhati-hati terhadap pengaruhnya.

**Melewatkan parameter**

Metode lain untuk berkomunikasi antar modul adalah dengan melewatkan parameter atau argumen antar modul. Parameter adalah *item* data sederhana yang ditransfer dari modul pemanggil ke modul subordinat pada saat pemanggilan. Ketika modul pemanggil memanggil sebuah modul subordinat dalam *pseudocode*, harus memuat nama modul yang dipanggil dengan sebuah daftar parameter yang akan dilewatkan ke modul yang dipanggil di dalam tanda kurung.

Sebuah contoh :

Print\_page\_headings (page\_count, line\_count)

Parameter dapat memiliki satu dari tiga fungsi berikut :

1. Untuk melewatkan informasi dari modul pemanggil kepada modul subordinat. Modul subordinat kemudian akan menggunakan menggunakan informasi tersebut dalam pemrosesan, tetapi tidak mengkomunikasikan informasi kembali ke modul pemanggil.
2. Untuk melewatkan informasi dari modul subordinat kepada modul pemanggil. Modul pemanggil kemudian menggunakan parameter tersebut untuk pemrosesan selanjutnya.
3. Untuk memenuhi peran komunikasi dua arah. Informasi dapat dilewatkan modul pemanggil ke modul subordinat yang diinginkan dan kemudian melewatkan kembali ke modul pemanggil.

Parameter-parameter yang dilewatkan antar modul dapat dimasukkan dalam *hierarchy chart* atau *structure chart* dengan menggunakan simbol berikut :

untuk parameter data

untuk parameter status

Parameter data memuat variabel yang sesungguhnya atau *item* data yang akan dilewatkan antar modul. Parameter status bertindak sebagai tanda program dan harus memuat hanya satu dari dua nilai : benar atau salah. Tanda program tersebut diatur benar atau salah sesuai dengan kondisi tertentu, yang kemudian dipergunakan untuk mengontrol pemrosesan selanjutnya.

Pada saat merancang program modular, pemrogram harus menghindari penggunaan parameter data untuk menunjukkan status, karena hal tersebut dapat mempengaruhi program dalam dua hal :

1. Membingungkan pembaca program karena sebuah variabel telah melebihi batas ketentuan, misalnya dipergunakan untuk lebih dari satu maksud
2. Menyebabkan kesalahan yang tidak terprediksi ketika program dirugikan pada beberapa waktu kemudian, karena pemrogram yang memelihara bisa tidak peduli terhadap dua maksud dari variabel

**Prototipe FUNCTION dalam C++**

Sebuah fungsi tidak dapat dipanggil kecuali sudah dideklarasikan, deklarasi fungsi dikenal dengna sebutan prototipe fungsi. Prototipe fungsi berupa:

* Nama fungsi
* Tipe nilai balik fungsi
* Jumlah dan tipe agumen

Dan diakhiri dengan titik koma, sebagaimanapada pendeklarasian variabel. Sebagai contoh:

1. long kuadatrat(long l);

Pada contoh pertama, fungsi kuadrat() mempunyai argumen bertipe long dan nilai balik bertipe long.

1. int maks(int a, int b, int c);

Pada contoh kedua, fungsi maks() memiliki tiga buah argumen, masing-masing bertipe int dan nilai balik juga bertipe int.

1. double maks(double x, double y);

Pada contoh ketiga, fungsi maks() mempunyai dua buah argumen, dengan masing-masing argumen bertipe double.

1. void garis();

Pada contoh keempat, fungsi garis() tidak memiliki argumen dan nilai baliknya tidak ada (void)

**Manfaat dari prototipe fungsi** adalah untuk menjamin tipe argumen yang dilewatkan paada pemanggilan fungsi benar-benar sesuai. Tanpa adanya prototipe fungsi, amatlah mudah bagi pemrogram untuk melakukan kesalahan tanpa sengaja dalam melewatkan argumen.

Pada prototipe fungsi, nama argumen boleh ditiadakan.

Sebagai contoh:

Long kuadrat (long);

Merupakan alternatif dari

Long kuadrat (long l);

Namun keberadaan nama argumen terkadang membantu bagi pembaca, dalam hal lebih memberikan kejelasan.

**Definisi fungsi**

Setiap fungsi yang dipanggil di dalam program harus didefinisikan. Letaknya bisa dimana saja. Khusus untuk fungsii yang disediakan di sistem, definisinya sebenarnya ada dalam pustaka, yang akan digaungkan dengan program sewaktu proses linking.

**Kasus 9.2**

Contoh lain misalkan Anda diminta membuat algoritma dan program untuk menentukan bilangan terbesar dari 2 buah bilangan yang diketahui dengan menggunakan fungsi. Anda terlebih dahuu harus mendeklarasikan fungsi untuk menampung dan membandingkan 2 buah bilangan tersebut.

Algoritma untuk permasalahan di atas:

1. Deklarasikan fungsi untuk menampung 2 buah bilangan( bil pertama dan bil kedua)
2. Deklarasikan fungsi untuk mencari bilangan terbesar
3. Tentukan dua buah bilangan tersebut
4. Tentukan variabel untuk menampung fungsi dari bilangan terbesar
5. Sesuai dengan langkah satu, lakukan pengujian untuk 2 buah bilangna tersebut
6. Jika bilangan pertama lebih besar dari bilangan kedua, dan jawabannya ya maka bilangan pertama yang menjadi ilangan terbesar
7. Dari langkah 6 bila jawabannya tidak, maka bilangan kedua yang menjadi bilangan terbesar.
8. Sesuai langkah 2, Cetak bilangan terbesar.

Program untuk kasus diatas adalah:

#include <iostream.h>

#include <conio.h>

int findmax(int n1, int n2); /\*Prototipe fungsi garis()\*/

void printmax(int m);

void main()

{

int x, y, k;

clrscr(); /\*Hapus layar\*/

cout<<"Masukkan Bilangan Pertama:";

cin>>x;

cout<<"Masukkan Bilangan Kedua:";

cin>>y;

k = findmax(x,y);

printmax(k);

}

int findmax(int n1, int n2) //definisi fungsi

{

if (n1>n2)

{

return (n1);

}

else

{

return (n2);

}

}

void printmax(int m)

{

cout<<"Bilangan terbesar adalah:"<<m;

}

Hasil program diatas (misalkan bil 1 dimasukkan angka 34 dan bil 2 dimasukkan angka 45)



**Fungsi Tanpa Nilai Balik**

Adakalanya suatu fungsi tidak perlu memiliki nilai balik. Misal fungsi yang hanya dimasukkan untuk menampilkan suatu keterangan saja. Pada fungsi seperti ini, tipe nilai balik fungsi yang diperlukan adalah void.

Contoh

Void tampilkan\_judul()

{

cout<<”Universitas INDONUSA Esa Ungul”<<endl;

cout<<”Jalan Terusan Arjuna Tol Tomang”<<endl;

cout<<”Jakarta Barat”<<endl;

}

Pada contoh di atas, tidak ada pernyataan return, mengingat fungsi tidak memiliki nilai balik. Namun penggunaan pernyataan return secara eksplisit juga bisa diperkenankan.

Void tampilkan\_judul()

{

cout<<”Universitas INDONUSA Esa Ungul”<<endl;

cout<<”Jalan Terusan Arjuna Tol Tomang”<<endl;

cout<<”Jakarta Barat”<<endl;

return;

}

**Lingkup Variabel**

Lingkup variabel menentukan keberadaan suatu variabel tertentu dalam suatu fungsi. Jenis variabel berdasarkan kelas penyimpanan:

* 1. variabel otomatis
	2. variabel eksernal
	3. variabel statis

**Variabel Otomatis**

Variabel yang didefinisikan di dalam suatu fungsi berlaku sebagai variabel lokal bagi fungsi. Artinya variabel tersebut hanya dikenal di dalam fungsi tempat variabel didefinisikan.

#include <iostream.h>

#include <conio.h>

void coba(); //Prototipe Fungsi

void main()

{

 int x= 30; //variabel lokal pada main()

 double y= 5.5;//variabel lokal pada main()

 clrscr(); //Hapus Layar

 cout<<"Pada main() nilai x = " <<x<<"\n";

 cout<<"Pada main() nilai y = " <<y <<"\n";

 coba();

 cout<<"Pada main() nilai x = " <<x<<"\n";

 cout<<"Pada main() nilai y = " <<y<<"\n";

}

//Definisi fungsi coba()

void coba()

{

 int x = 50;

 double y = 11.5;

 cout<<"Pada coba() nilai x = " <<x<<"\n";

 cout<<"Pada coba() nilai y = " <<y<<"\n";

}

Hasil Program adalah:



Tampak bahwa perubahan x dan y pada coba() tidak mempengaruhi variabel dengan nama yang sama pada main(). Ini membuktikan bahwa variabel-variabel tersebut bersifat lokal bagi masing-masing fungsi yang mendefinisikannya.

Suatu variabel otomatis mempunyai sifat:

* + variabel hanya akan diciptakan pada saat fungsi dipanggil
	+ Pada saat fungsi berakhir (selesai dieksekusi), variabel otomatis akan sirna
	+ Tidak ada inisialisasi secara otomatis (saat variabel diciptakan). Inisialisasi program akan dikerjakan setiap kali fungsi dipanggil.
	+ Hanya dapat diakses di dalam fungsi yang mendefinisikannya

**Variabel Eksternal**

 Variabel eksternal merupakan kebalikan variabel otomatis. Variabel eksternal adalah ariabel yang didefinisikan di luar fungsi manapun. Variabel ini dikenal juga sebagai variabel global sebab variabel ini dikenal di semua fungsi.

Kasus 9.3 Contoh penggunaan variabel eksternal

#include <iostream.h>

#include <conio.h>

int a = 10; //variabel eksternal

void kali(); //Prototipe Fungsi

void main()

{

 clrscr();

 cout<<a<<endl;

 kali();

 cout<<a<<endl;

 kali();

 cout<<a<<endl;

}

//Definisi Fungsi

void kali()

{

 a = a\*2;

}

Hasil program di atas adalah:



Tampak bahwa meskipun di dalam fungsi main() dan kali() tidak mendefinisikan variabel a, ternyata variabel ini dikenal di dalam kedua fungsi tersebut. Nilai dari variabel a dapat diubah dari dalam fungsi kali().

Penggunaan variabel eksternal diusahakan sesedikit mungkin atau sedapat mungkin tidak usah digunakan. Tidka lain adalah karena ini mudah sekali berubah oleh pernyataanpenugasan yang letaknya bisa di mana saja. Ini bisa menimbulkan efek samping yang suli untuk melacaknya, terutama untuk program besar.

**Variabel Statis**

 Baik variabel eksternal maupun otomatis dapat berkedudukan sebagai variabel statis. Suatu variabel statis mempunyai sifat:

1. Jika variabel lokal berdiri sebagai variabel statis, maka:
	* variabel tetap hanya dapat diakses pada fungsi yang mendefinisikannya
	* Variabel tidak hilang saat diekskusi fungsi berakhir. Nilainya akan tetap dipertahankan, sehingga akan dikenali pada pemanggilan fungsi untuk tahap berikutnya.
	* Inisialisasi oleh pemrogram akan dilakukan sekali saja selama program dijalankan. Jika tidak ada inisialisasi secara eksplisit, variabel diisi dengan nol.
2. Jika variabel eksternaldijadikan sebagai variabel statis variabel ini dapat diakses oleh semua file yang didefinisikan pada file yang sama dengan variabel eksternal tersebut.

Kasus 9.4 Contoh penggunaan variabel static

#include <iostream.h>

#include <conio.h>

void panggil(); //Prototipe Fungsi

void main()

{

 int baru = 50;

 clrscr();

 panggil();

 panggil();

 panggil();

 cout<<"Pada main() nilai baru ="<<baru<<endl;

}

//Pada fungsi berikut variabel baru didefinisikan sebagai variabel statis

//Definisi Fungsi

void panggil()

{

 static int baru= 100;

 baru = baru \*2;

 cout<<"Pada panggil() nilai baru ="<<baru<<endl;

}

Hasil untuk program di atas adalah:



Berdasarkan hasil di atas, terlihat bahwa variabel statis baru pada fungsi panggil hanya diinisialisasikan sekali saja kemudian seap fungsi panggil() diakses nilai variabel tersebut diproses.

Tampak pula bahwa variabel bernama sama yang didefinisikan di fungsi main() tidak ada kaitannya dengan variabel yang ada di fungsi panggil().

**Nilai Bawaan untuk Argumen Fungsi**

Salah satu keistimewaan C++ yang sangat bermanfaat dalam pemrogramn adalah adanya kemampuan untk menyetel nilai bawaan (default) argumen fungsi. Argumen-argumen yang mempunyai nilai bawaan nantinya dapat tidak disertakan di dalam pemanggilan fungsi dan dengan sendirinya C++ akan menggunakan nilai bawaan dari argumen yang tidak disertakan.

**Kasus 9.5** Lihat contoh berikut ini:

#include <iostream.h>

#include <conio.h>

void cetak(int jum); //Prototipe Fungsi

void main()

{

 clrscr();

 cetak(1);

}

//Definisi Fungsi

void cetak(int jum)

{

 for (int i=0;i<jum;i++)

 cout<<"Aku Pasti Bisa"<<endl;

 cout<<"Hidup Informatika"<<endl;

}

Hasil program di atas adalah



**Kasus 9.5** Sebuah contoh lain mengenai pemberian bilai awal terhadap argumen fungsi:

#include <iostream.h>

#include <conio.h>

void cetak(char x='\*', int jum= 10); //Prototipe Fungsi

void main()

{

 clrscr();

 cetak('A', 5); //Tampilkan A sebanyak 5 kali

 cetak('B'); //Tampilkan B sebanyak 10 kali

 cetak(); //Tampilan \* sebanyak 10 kali

}

//Definisi Fungsi

void cetak(char x, int jum)

{

 for (int i=0;i<jum;i++)

 cout<<x;

 cout<<endl;

}

Hasil Program di atas adalah:



**Referensi**

Pada C++ referensi digunakan untuk memberikan nama alias dari variabel. Bentuk peneklarasiannya:

**Int & ref = nama-variabel;**

Tanda & mengawali nama referensi.

Setelah pendeklarasian seperti di atas, ref menjadi nama alias dari nama\_variabel. Pengubahan nilai terhadap nama-variabel dapat dilakukan melalui nama\_variabel itu sendiri ataupun nelalui referensi ref, sebagaimana contoh berikut ini:

#include <iostream.h>

#include <conio.h>

void main()

{

 int asal;

 int &palsu = asal; //Deklarasi referensi

 clrscr();

 cout<<"Masukkan Bilangan Bulat sembarang: ";

 cin>>asal;

 cout<<"\nBilangan asal :"<<asal<<endl;

 cout<<"Bilangan palsu :"<<palsu<<endl;

 palsu = 55;

 cout<<"\nBilangan asal :"<<asal<<endl;

 cout<<"Bilangan palsu :"<<palsu<<endl;

}

Hasil Program tersebut (Bila dimasukkan angka 100 pada variabel asal) adalah:



**Kasus 9.6** Contoh program yang menggunakan operator:

#include <iostream.h>

#include <conio.h>

void main()

{

 int asal;

 int &palsu = asal; //Deklarasi referensi

 clrscr();

 cout<<"Masukkan Bilangan Bulat sembarang: ";

 cin>>asal;

 cout<<"\nBilangan asal :"<<asal<<endl;

 cout<<"Bilangan palsu :"<<palsu<<endl;

 asal++;

 cout<<"\nBilangan asal :"<<asal<<endl;

 cout<<"Bilangan palsu :"<<palsu<<endl;

 palsu++;

 cout<<"\nBilangan asal :"<<asal<<endl;

 cout<<"Bilangan palsu :"<<palsu<<endl;

}

Hasil dari program di atas bila dimasukkan angka 75 adalah;



Tampak penggunaan operator ++ pada asal dan palsu akan mengubah nilai keduanya, hal ini terjadi karena variabel asal dan palsu menunjuk pada alamat memori yang sama.

**Kasus 9.7** Berikut ini program untuk menampilkan alamat memori suatu variabel:

#include <iostream.h>

#include <conio.h>

void main()

{

 int asal;

 int &palsu = asal; //Deklarasi referensi

 clrscr();

 cout<<"Masukkan Bilangan Bulat sembarang: ";

 cin>>asal;

 cout<<"\nBilangan asal :"<<asal<<endl;

 cout<<"Bilangan palsu :"<<palsu<<endl;

 cout<<"Alamat memori asal :"<<&asal<<endl;

 cout<<"Alamat memori palsu :"<<&palsu<<endl;

}

Hasil program di atas bila dimasukkan angka 90 adalah:



**Latihan:**

* + 1. Buatlah algoritma dan program untuk menentukan jumlah gaji bersih dari gaji pokok yang diinputkan ditambah dengan jumlah bonus juga diinputkan. Untuk mencari bonus adalah jumlah bonus yang dimasukkan dikalikan dengan gaji pokok. Gaji bersih = gaji pokok + jumlah bonus.

Fungsi yang diinginkan adalah:

* + Fungsi Input Data
	+ Fungsi Mencari Bonus
	+ Fungsi Gaji Total
1. Buatlah program dalam bentuk MENU UTAMA yang berisi operasi aritmatika penjumlahan, pengurangan, perkalian dan pembagian. Menu utama tersebut berisi 5 pilihan yang tampil seperti berikut:

MENU UTAMA

* + 1. OPERASI PENJUMLAHAN
		2. OPERASI PENGURANGAN
		3. OPERASI PERKALIAN
		4. OPERASI PEMBAGIAN
		5. KELUAR

MASUKKAN KODE [1, 2, 3, 4,5]

Menu di atas akan di ulang secara terus menerus dengan meminta pengguna supaya memasukkan kode. Jika kode yang dimasukkan angka 5 maka akan muncul pesan “Terima Kasih”