BAB II
TIPE DATA TURUNAN

A. TUJUAN

1. Mahasiswa mampu mengetahui dan memahami berbagai jenis tipe data turunan
2. Mahasiswa mampu memahami fungsi
3. Mahasiswa mampu memahami array
4. Mahasiswa mampu memahami pointer dan reference
5. Mahasiswa mampu mengimplementasikan tipe data tipe data turunan berdasarkan kasus yang diberikan menggunakan bahasa pemrograman C++.

B. ALAT DAN BAHAN

1. Laptop
2. Mouse
3. IDE C++ (Visual Studio Code)

C. KESEHATAN DAN KESELAMATAN KERJA

1. Hati-hatilah dalam memakai perangkat elektronik
2. Pastikan kabel listrik terpasang dan dalam kondisi baik
3. Lakukan praktikum dalam posisi duduk yang benar
4. Jauhkan kabel listrik dari sentuhan anda
5. Gunakan alas kaki celana Panjang dan kemeja
6. Gunakan kacamata anti radiasi layar

D. TEORI

1. Tipe Data Turunan

Tipe data turunan atau yang dikenal dengan istilah Derived Data Type merupakan tipe data yang diturunkan dari tipe data dasar yang ada. Dengan kata lain, tipe data turunan mengandung sekelompok tipe data dasar di dalamnya. Terdapat 4 jenis tipe data turunan yaitu:

- Fungsi
- Array
- Pointer
- Reference

Namun beberapa sumber tidak membatasi tipe data turunan hanya pada keempat tipe data di atas. Namun beberapa sumber juga memasukkan class, struct, enum, dll. Namun mereka tidak memiliki tipe data bentukan. Jadi semua dianggap sebagai tipe data turunan. Pada modul ini kita akan menggunakan defenisi di atas, yakni tipe deata turunan adalah fungsi, array, pointer dan reference.

2. Fungsi

a. Membentuk fungsi

Fungsi adalah blok kode atau segmen program yang didefinisikan untuk melakukan tugas tertentu yang didefinisikan dengan jelas. Suatu fungsi pada umumnya didefinisikan untuk mencegah programer menulis baris kode yang sama berulang kali untuk input yang sama. Semua baris kode disatukan dalam satu fungsi tunggal dan bisa dipanggil di mana saja saat diperlukan. main () adalah fungsi default yang didefinisikan dalam setiap program C ++.

Fungsi adalah sub-program yang bisa digunakan kembali baik di dalam program itu sendiri, maupun di program yang lain. Fungsi dapat menerima input dan menghasilkan output. Dalam pemrograman, fungsi atau prosedur sering digunakan untuk membungkus program menjadi bagian-bagian kecil. Tujuannya agar program tidak menumpuk pada fungsi main() saja. Bayangkan saja, kalau program kita bertambah besar dan kompleks. Jika semua kodenya ditulis di dalam fungsi main() saja, maka kita akan kesulitan membacanya. Fungsi biasanya akan mengembalikan sebuah nilai dari hasil prosesnya. Karena itu, kita harus menentukan tipe data untuk nilai yang akan dikembalikan. Apabila fungsi tersebut tidak memiliki nilai kembalian, maka kita harus menggunakan tipe void untuk menyatakan kalau fungsi tersebut tidak akan mengembalikan nilai apa-apa.

Adapun sintaks untuk membuat sebuah fungsi adalah sebagai berikut:

Sintaks I (fungsi dengan 1 parameter):

```
Tipe_nilai_balik nama_fungsi(Tipe_data nama parameter){
   //Badan_Fungsi
};
```

Sintaks II (fungsi dengan 2 parameter atau lebh):

```
Tipe_nilai_balik nama_fungsi(Tipe_data nama parameter1,
   Tipe_data nama parameter1){
   //Badan_Fungsi
```

```
};
```

Sintaks III (fungsi tanpa parameter):

```
Tipe_nilai_balik nama_fungsi(){
   //Badan_Fungsi
};
```

Keterangan:

- **Tipe_nilai_balik**: merupakan tipe data dari nilai yang akan dikembalikan oleh fungsi tersebut kepada perintah pemanggilnya. Bagian ini harus sesuai dengan tipe data dari nilai atau variable yang akan dikembalikan. Jika suatu fungsi tidak mengembalikan nilai apapun maka diisi dengan *void*.
- **Nama_fungsi**: tempat dimana anda dapat mendifinisikan nama dari fungsi yang akan anda bentuk. Nama fungsi sebaiknya menggambarkan tujuan dari fungsi tersebut.
- **Tipe data nama_parameter**: Pada bagian parameter, tipe data ini ditentukan sesuai dengan tipe data dari nilai yang dikirimkan pada parameter dari perintah pemanggilnya. Nama parameter merupakan variable yang menampung data yang dikirim dari perintah pemanggil. Jika suatu fungsi tidak membutuhkan nilai parameter, maka bagian ini dapat dikosongkan.
- **Badan_fungsi**: Merupakan bagian proses pengolahan data sehingga suatu fungsi dapat menghasilkan output yang diharapkan sesuai dengan tujuan dibentuknya suatu fungsi. Jika fungsi bernilai balik maka harus ada perintah pengembalikan nilai yaitu menggunakan keyword return terhadap variable yang nilainya akan dikembalikan.

b. Mengakses fungsi

Saat mendirikan sebuah fungsi buatan, pernyataan-pernyataan yang ada di dalam fungsi buatan tidak akan dibaca oleh CPU jika nama atau identifier pada fungsi yang kita buat tidak dipanggil di dalam fungsi utama atau dalam fungsi lain yang sudah terjamin CPU akan melewainya.

Untuk memanggil fungsi, kita cukup menuliskan nama dari fungsi tersebut dan memberi sepasang tanda ( ), itu merupakan tempat dimana nilai arguments untuk parameter fungsi tersebut, kita bisa mengkosonginya jika memang fungsi tersebut tidak membutuhkan argument.

Adapun sintaks untuk mengakses sebuah fungsi adalah sebagai berikut:

Sintaks I (Mengakses fungsi tanpa parameter):

```
nama_fungsi();
```

Sintaks II (Mengakses fungsi dengan parameter):

```
nama_fungsi(nilai_argumen);
```

Sintaks III (Mengakses fungsi dengan nilai balik):

```
tipe_data nama_variabel = nama_fungsi(nilai_argumen);
```

Jika parameter lebih dari satu, maka pisahkan tiap nilai argumen dengan koma sesuai susunan parameter pada fungsi yang dipanggil. Selain itu nilai argumen juga bisa merujuk kepada suatu variabel yang berisi sebuah nilai.

3. Array

a. Membentuk array

Array atau larik adalah sebuah variabel yang memiliki serangkaian elemen dari jenis tipe data yang sama. Elemen-elemen tersebut dirangkai di dalam memori yang berdekatan dengan elemen lainya. konsep ini mirip seperti struct dan class yang dapat merupakan kumpulan dari variabel, perbedaan dengan mereka adalah array merupakan kumpulan variabel dengan satu indentifier dan satu tipe data yang sama, dan dalam istilah-istilah array kumpulan dari variabel tersebut adalah elemen. Array adalah kumpulan item yang disimpan di lokasi memori berkelanjutan. Ide array adalah untuk merepresentasikan banyak instance dalam satu variabel.

Array adalah sebuah variabel yang menyimpan lebih dari satu buah data yang memiliki tipe data yang sama. Array juga dapat didefenisikan sebagai tipe terstruktur yang terdiri dari sejumlah

komponen-komponen yang mempunyai tipe yang sama. Suatu array mempunyai jumlah komponen yang banyaknya tetap. Banyaknya komponen dalam suatu array ditunjuk oleh suatu indek untuk membedakan variabel yang satu dengan variabel lainnya.

Setiap data yang terdapat dalam array tersebut menempati alamat memori yang berbeda disebut elemen array. Untuk mengakses nilai dari suatu elemen array, akan digunakan indeks dari array tersebut. Sangat perlu diperhatikan bahwa dalam bahasa C/C++, indeks array selalu dimulai dari angka 0, bukan 1. Hal ini berbeda dengan bahasa pemrograman lainnya misalnya bahasa pascal dimana indeks awal array dapat ditentukan dengan keinginan kita. Untuk mendekalarasikan suatu array dalam bahasa C++ adalah dengan menggunakan tanda [ ] (bracket).

Berikut gambaran dari sebuah array yang dideklarasikan dengan lebar N.

Nilai ke-1	Nilai ke-2	…	Nilai ke-N	←----------Nilai elemen array
Alamat ke-1	Alamat ke-2	…	Alamat ke-N	←----------Alamat elemen array
0	1	…	N	←----------Indeks elemen array

Adapun sintaks untuk mendeklarasikan sebuah array adalah sebgai berikut:

Sintaks I (deklarasi array 1 dimensi)

```
Tipe_data nama_array[ukuran_array];
```

Sintaks II (deklarasi array 2 dimensi)

```
Tipe_data nama_array[lebar_array][tinggi_array];
```

Keterangan:

- **tipe_data**: merupakan tipe data dari array yang akan dibentuk
- **nama_array**: merupakan nama dari array. Pemberian nama sebaiknya berkaitan dengan data yang akan ditampung.
- **ukuran_array**: merupakan banyaknya data yang akan ditampung. Indeks array dimulai dai 0. Jika array merupakan array multi dimensi, maka ukuran pertama merupakan lebar dan ukuran kedua adalah tinggi array.

b. Menginisialisasi nilai array

Saat mendirikan variabel array kita juga dimungkinkan untuk memberi nilai saat deklarasi variabel array secara bersamaan, inisialisasi array disebut sebagai "Initializer list". apa yang kita butuhkan untuk inisialisasi sebuah variabel array adalah sepasang tanda kurung kurawal yang mengapit semua nilai tersebut. Adapun sintkas yang dapat digunakan untuk mengisi nilai terhadap suatu array adalah sebagai berikut:

Sintaks I (inisialisasi nilai secara langsung)

```
Tipe_data nama_array[N]={nilai1, nilai2,…,nilaiN};
```

Sintaks II (inisialisasi nilai pada indeks tertentu)

```
nama_array[indeks]= nilai;
```

Keterangan:

- **tipe_data**: merupakan tipe data dari array yang akan dibentuk
- **nama_array**: merupakan nama dari array. Pemberian nama sebaiknya berkaitan dengan data yang akan ditampung.
- **N**: merupakan banyaknya data yang akan ditampung. Indeks array dimulai dai 0. Jika array merupakan array multi dimensi, maka ukuran pertama merupakan lebar dan ukuran kedua adalah tinggi array.
- **indeks**: merupakan alamat elemen dalam array atau posisi elemen di dalam array. Indeks selalu dimulai dari 0.
- **Nilai**: merupakan nilai yang akan ditugaskan masuk ke dalam array pada indeks tertentu.

c. Mengakses array

Adapun sintaks untuk mengakses elemen array adalah sebagai berikut:

Sintaks I

```
nama_array[indeks];
```

Keterangan:

- **nama_array**: merupakan nama dari array yang akan diakses nilainya.
- **indeks**: merupakan alamat elemen dalam array atau posisi elemen di dalam array. Indeks selalu dimulai dari 0.

4. Pointer

Pointer adalah representasi simbolik dari alamat memory. Pointer memungkinkan program untuk mensimulasikan call-by-reference serta untuk membuat dan memanipulasi struktur data secara dinamis. Deklarasi umum dalam C / C ++ memiliki format:

Sintaks I

```
Tipe_data *nama_pointer;
```

Keterangan:

- **tipe_data**: merupakan tipe data dari variable pointer yang dideklarasikan.
- *: merupakan penanda bahwa variable yang melekat adalah variable pointer (variable yang menyimpan alamat memory)
- **nama_pointer**: merupakan nama variable yang dideklarasikan sebagai variable pointer.

5. Reference

Ketika sebuah variabel dinyatakan sebagai referensi, itu menjadi nama alternatif untuk variabel yang ada. Pertama-tama variable harus dideklarasikan seperti biasa. Variabel dapat dideklarasikan sebagai referensi dengan menambahkan '&' di depan variable tersebut.

Sintaks I

```
tipe_data &nama_alias = variabel;
```

Keterangan:

- tipe_data: merupakan tipe data dari variable reference yang dideklarasikan.
- **&**: merupakan penanda bahwa variable yang melekat adalah variable reference.
- **nama_alias**: merupakan nama variable yang dideklarasikan sebagai variable reference/alias dari variable lain yang direferensikan.

- **Variable**: merupakan variable yang tipe datanya sama dengan tipe data variable reference. Alamat memory dari variable ini akan disimpan oleh variable reference.

E. PRAKTIKUM

Praktikum 2.1. Fungsi

```
#include <iostream>
using namespace std;
int pilihan;

float cariLuasalasbalok(float panjangBalok, float lebarBalok) {
    float luasAlasbalok = panjangBalok * lebarBalok;
    return luasAlasbalok;
}

float cariVolumebalok(float panjangBalok, float lebarBalok, float
tinggiBalok){
    float luasAlasbalok = cariLuasalasbalok(panjangBalok, lebarBalok);
    float volumeBalok = luasAlasbalok * tinggiBalok;
    return volumeBalok;
}

float cariVolumekubus(float panjangSisi){
    float volumeKubus = panjangSisi*panjangSisi*panjangSisi;
    return volumeKubus;
}

int main() {
    cout <<"===Program Menghitung Volume Balok dan Kubus"<<endl;
    cout << "1. Cari volume Balok "<<endl;
    cout << "2. Cari volume Kubus "<<endl;
    cout << "Masukkan angka menu volume yang akan dihitung: "<<endl;
    cin >> pilihan;

    if (pilihan == 1){
        float panjangBalok;
        float lebarBalok;
        float tinggiBalok;
        cout << "Anda memilih menghitung volume balok!" << endl;
        cout << "Masukkan panjang balok (cm)!" << endl;
        cin >> panjangBalok;
        cout << "Masukkan lebar balok (cm)!" << endl;
        cin >> lebarBalok;
        cout << "Masukkan tinggi balok (cm)!" << endl;
        cin >> tinggiBalok;
```

```
        float volumeBalok = cariVolumebalok (panjangBalok, lebarBalok,
tinggiBalok);
        cout << "Volume balok adalah "<< volumeBalok << " cm3" << endl;
    } else if (pilihan == 2){
        float panjangSisikubus;
        cout << "Anda memilih menghitung volume kubus!" << endl;
        cout << "Masukkan panjang sisi kubus (cm)!" << endl;
        cin >> panjangSisikubus;
        float volumeKubus = cariVolumekubus (panjangSisikubus);
        cout << "Volume kubus adalah "<< volumeKubus << " cm3" <<endl;
    } else {
        cout << "Anda salah menginput pilihan!" << endl;
    }

return 0;
}
```

Praktikum 2.2. Array

```
#include <iostream>
using namespace std;

float cariLuassegitigask (float alas, float tinggi){
    float luas = (alas*tinggi)/2;
    return luas;
}

int main() {
    float alasSegitigask[3];
    float tinggiSegitigask[3];

    cout << "===Program menghitung luas 3 segitiga siku-siku==="<<endl;
    for (int i = 0; i<3; i++){
        cout<<"Masukkan panjang alas segitiga (cm) ke-"<< i+1 << endl;
        cin >> alasSegitigask[i];
    };

    for (int i = 0; i<3; i++){
        cout<<"Masukkan tinggi segitiga (cm) ke-"<< i+1 << endl;
        cin >> tinggiSegitigask[i];
    };

    for (int i = 0; i<3; i++){
        cout<<"Luas segitiga siku-siku ke-"<< i+1 << " adalah:" << endl;
        float segiTigasikusiku = cariLuassegitigask(alasSegitigask[i],
tinggiSegitigask[i]);
```

```
        cout<<segiTigasikusiku<< " cm2" << endl;
    };

return 0;
}
```

Praktikum 2.3. Pointer dan Reference

```
#include <iostream>
using namespace std;

int main() {

  int *P;
  int X;
  P = &X;
  X = 10;

  cout<<"Nilai X  : "<<X<<endl;
  cout<<"Nilai *P : "<<*P<<endl;
  cout<<"Nilai P  : "<<P<<endl;
  cout<<"Nilai &X : "<<&X<<endl;

  *P = 200;

  cout<<"Nilai X  : "<<X<<endl;
  cout<<"Nilai *P : "<<*P<<endl;
  cout<<"Nilai P  : "<<P<<endl;
  cout<<"Nilai &X : "<<&X<<endl;

  return 0;
}
```

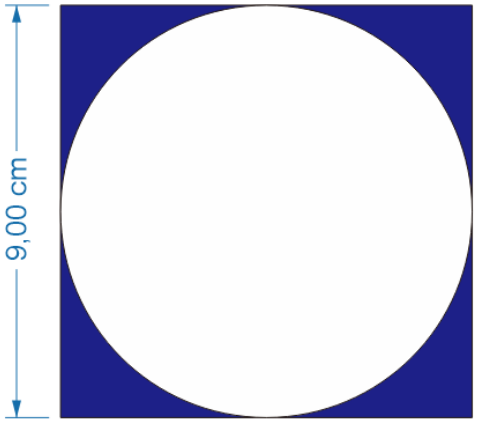
Praktikum 2.4. Pointer dan Reference (Pass-by-reference)

```
#include <iostream>
using namespace std;

void penjumlahan(int &nilai1, int &nilai2, int penambah){
     nilai1 = nilai1 + penambah;
     nilai2 = nilai2 - penambah;
```

```
}

int main(){

    int nilaiPertama=5, nilaiKedua=5, nilaiPenambah=5;
    int *aliasNilai1;
    int *aliasNilai2;

    aliasNilai1 = &nilaiPertama;
    aliasNilai2 = &nilaiKedua;

    *aliasNilai1 = 10;
    *aliasNilai2 = 20;

    penjumlahan(nilaiPertama, nilaiKedua, nilaiPenambah);

    cout<<"Penjumlahan 1 = "<<nilaiPertama<<endl;
    cout<<"Penjumlahan 2 = "<<nilaiKedua<<endl;

    return 0;
}
```

F. LATIHAN

1. Perhatikan Gambar berikut ini



Terdapat sebuah persegi dengan lebar sisi 9 cm. Ditengah persegi tersebut terdapat sebuah lingkaran seperti pada gambar di atas. Berapakah luas area yang berwarna biru?

Catatan!

1. Buat algoritma untuk menghitung luas area berwarna biru!

2. Buat program berdasarkan algoritma yang telah anda buat untuk menghitung luas area yang berwarna biru tersebut!
3. Program wajib menggunakan fungsi dan argumen menggunakan teknik pass-by-reference!

CATATAN