Modul I

Algoritma dan Struktur Data

“Tipe Data Primitif, Tipe Data Abstrak dan Tipe Data Koleksi”

Tujuan Pembelajaran :

- Mahasiswa dapat mempelajari berbagai jenis tipe data dari tipe data primitif, abstrak dan koleksi.
- Mahasiswa dapat memahami pengaplikasian pada tools yang digunakan
- Mahasiswa dapat menggunakan bahasa pemprograman yang telah ditentukan.

Dasar Teori :

Tipe data adalah adalah sebuah pengklasifikasian data berdasarkan jenis data tersebut. Tipe data dibutuhkan agar kompiler dapat mengetahui bagaimana sebuah data akan digunakan. Adapun tipe data yang akan dipelajari, sebagai berikut :

1. Tipe Data Primitive
2. Tipe Data Abstrak
3. Tipe Data Koleksi

1. Tipe Data Primitive

Tipe data primitive hanya dapat menyimpan satu nilai pada satu waktu dan tidak dapat diubah menggunakan cara yang sama seperti tipe data non-primitif. Tipe data Primitif akan dianggap sama jika nilainya sama. Berikut adalah daftar tipe data primitif yang umum digunakan di bahasa pemrograman C++:

1. `int` : tipe data ini digunakan untuk menyimpan bilangan bulat seperti 1, 2, 3, dan seterusnya.
2. `float` : tipe data ini digunakan untuk menyimpan bilangan pecahan seperti 2.5, 3.14, dan seterusnya.
3. `char` : tipe data ini digunakan untuk menyimpan karakter seperti ‘a’, ‘b’, ‘c’, dan seterusnya.
4. `bool` : tipe data ini digunakan untuk menyimpan nilai boolean yang hanya memiliki dua nilai yaitu true dan false.

- **Guided**

Berikut adalah contoh penggunaan tipe data primitif char dan float.

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

// Main program
main()
{
    char op;
    float num1, num2;

    // It allows user to enter operator i.e. +, -, *, /
    cin >> op;

    // It allow user to enter the operands
    cin >> num1 >> num2;

    // Switch statement begins
    switch (op) {

        // If user enter +
        case '+':
            cout << num1 + num2;
            break;

        // If user enter -
        case '-':
            cout << num1 - num2;
            break;

        // If user enter *
        case '*':
            cout << num1 * num2;
            break;

        // If user enter /
        case '/':
            cout << num1 / num2;
            break;

        // If the operator is other than +, -, * or /,
```

```
        // error message will display
        default:
            cout << "Error! operator is not correct";

    } // switch statement ends

    return 0;
}
```

2. Tipe Data Abstrak

Abstract Data Type (ADT) adalah tipe data yang dibuat untuk menggambarkan karakter/kondisi (state) dan perilaku (behaviour) dari sebuah object. Kita dapat membuat tipe data sendiri (user defined data type), misalnya string dalam C++. Tipe data abstract String dibuat dalam C++ class.

Fitur *Class* `class` adalah fitur OOP pada C++ mirip seperti FItur *Data Structures* `struct` pada C, keduanya dapat menampung *variabel* sebagai anggota.

A. *Class* adalah salah satu dari konsep OOP yang digunakan untuk membungkus data abstraksi *procedural* sebagai deskripsi tergeneralisir atau rancangan dari sebuah *object* untuk mendefinisikan atau menggambarkan isi dan tingkah laku sebagai entitas dari *object*.

B. Structure atau struct adalah kumpulan dari beberapa variabel dengan beragam tipe data yang dibungkus dalam satu varabel. Struct juga dikenal dengan *records* dalam bahasa pemrograman lain seperti Pascal.

- **Guided**

1. Berikut adalah contoh program class

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

class Rectangle {
    int width, height;
  public:
    void set_values (int,int);
    int area() {return width*height;}
};
```

```
void Rectangle::set_values (int x, int y) {
  width = x;
  height = y;
}

int main () {
  Rectangle rect;
  rect.set_values (3,4);
  cout << "area: " << rect.area();
  return 0;
}
```

2. Berikut contoh program dari fungsi struct

```
#include <stdio.h>

// membuat struct
struct Mahasiswa {
    char *name;
    char *address;
    int age;
};

void main(){

    // menggunakan struct
    struct Mahasiswa mhs1, mhs2;

    // mengisi nilai ke struct
    mhs1.name = "Dian";
    mhs1.address = "Mataram";
    mhs1.age = 22;

    mhs2.name = "Bambang";
    mhs2.address = "Surabaya";
    mhs2.age = 23;

    // mencetak isi struct
    printf("## Mahasiswa 1 ##\n");
    printf("Nama: %s\n", mhs1.name);
    printf("Alamat: %s\n", mhs1.address);
```

```
    printf("Umur: %d\n", mhs1.age);

    printf("## Mahasiswa 2 ##\n");
    printf("Nama: %s\n", mhs2.name);
    printf("Alamat: %s\n", mhs2.address);
    printf("Umur: %d\n", mhs2.age);

}
```

3. Tipe Data Koleksi

Koleksi adalah tipe data yang berupa rangkaian atau kumpulan data ataupun objek yang berindeks. Dalam program C++, kita dapat menggunakan kontainer Standard Template Library (STL) secara gratis, atau jenis koleksi lain yang ditentukan pengguna. Terdapat tiga tipe dasar koleksi di C++ yaitu:

1. Array
   Tipe data array adalah tipe data yang terdiri dari kumpulan tipe data lain. Anggota atau isi dari array itu sendiri harus satu jenis tipe data, misalkan terdiri dari kumpulan angka bulat saja (*integer*), kumpulan karakter saja (*char*), maupun kumpulan angka pecahan saja (*double*).

   - Guided

```
#include <iostream>

using namespace std;

int main()
{
  int nilai[5];

  nilai[0] = 23;
  nilai[1] = 50;
  nilai[2] = 34;
  nilai[3] = 78;
  nilai[4] = 90;

  cout << "Isi array pertama :" << nilai[0] <<endl;
  cout << "Isi array kedua   :" << nilai[1] <<endl;
  cout << "Isi array ketiga  :" << nilai[2] <<endl;
  cout << "Isi array keempat :" << nilai[3] <<endl;
  cout << "Isi array kelima  :" << nilai[4] <<endl;

  return 0;
}
```

2. Map

Map terasa mirip dengan array namun dengan index yang memungkinkan untuk berupa tipe data selain integer. Pada map, indeks tersebut diberi nama "key". Pada `std::map` digunakan Self-Balancing Tree khususnya Red-Black Tree.
Beberapa fungsi dasar yang terkait dengan Map:

begin() – Mengembalikan iterator ke elemen pertama di Map.
end() – Mengembalikan iterator ke elemen teoretis yang mengikuti elemen terakhir di Map.
size() – Mengembalikan jumlah elemen di map.
max_size() – Mengembalikan jumlah maksimum elemen yang dapat ditampung map.
kosong() – Mengembalikan apakah map kosong.
pair insert(keyvalue, mapvalue) – Menambahkan elemen baru ke map.
erase(iterator position) – Menghapus elemen pada posisi yang ditunjuk oleh iterator.
erase(const g)– Menghapus nilai kunci 'g' dari map.
clear() – Menghapus semua elemen dari map.

- Guided

```
#include <collection.h>
#include <algorithm>
using namespace Platform;
using namespace Platform::Collections;
using namespace Windows::Foundation::Collections;


void Class1::Test()
{
    Vector<int>^ vec = ref new Vector<int>();
    vec->Append(1);
    vec->Append(2);
    vec->Append(3);
    vec->Append(4);
    vec->Append(5);


    auto it =
        std::find(begin(vec), end(vec), 3);

    int j = *it; //j = 3
    int k = *(it + 1); //or it[1]

    // Find a specified value.
    unsigned int n;
    bool found = vec->IndexOf(4, &n); //n = 3

    // Get the value at the specified index.
```

```
    n = vec->GetAt(4); // n = 3

    // Insert an item.
    // vec = 0, 1, 2, 3, 4, 5
    vec->InsertAt(0, 0);

    // Modify an item.
    // vec = 0, 1, 2, 12, 4, 5,
    vec->SetAt(3, 12);

    // Remove an item.
    //vec = 1, 2, 12, 4, 5
    vec->RemoveAt(0);

    // vec = 1, 2, 12, 4
    vec->RemoveAtEnd();

    // Get a read-only view into the vector.
    IVectorView<int>^ view = vec->GetView();
}
```

3. Vector

   Vector adalah *Standard Template Library* (STL) jika di dalam C/C++ memiliki bentuk std::vector . Umumnya, vector mirip seperti *array* yang memiliki kemapuan untuk menyimpan data dalam bentuk elemen-elemen yang alokasi memorinya dilakukan otomatis dan bersebelahan. Kemampuan *vector* bukan hanya pada jumlah elemen yang dinamis, *vector* pada C/C++ juga dilengkapi dengan fitur-fitur pelengkap seperti *element access*, *iterators*, *capacity*, *modifiers*.

- Guided

```
#include <vector>
#include <iostream>

using namespace std;

int main(){
    vector<int> nomer = {1,2,3};
    int no_elemen = 1;

    //memodifikasi nilai
    nomer[no_elemen] = 10;

    //melihat nilai
```

```
    cout<<"nilai dari elemen "<<no_elemen<<" adalah
"<<nomer[no_elemen];

    return 0;
}
```

UNGUIDED

1. Buatlah program menggunakan tipe data primitif minimal dua fungsi dan bebas. Menampilkan program, jelaskan program tersebut dan ambil kesimpulan dari materi tipe data primitif!
2. Jelaskan fungsi dari class dan structur secara detail dan berikan contoh programnya
3. Buat dan jelaskan program menggunakan fungsi map dan jelaskan perbedaan dari array dengan map.