

LAPORAN PRAKTIKUM

MODUL I TIPE DATA



Disusun oleh:
Geranada Saputra Priambudi
NIM: 2311102008

Dosen Pengampu:
Wahyu Andi Saputra, S.Pd., M,Eng.

**PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO
PURWOKERTO
2023**

BAB I

TUJUAN PRAKTIKUM

1. Memahami tipe data dan macam-macam tipe data.
2. Menggunakan tipe data primitif, abstrak dan kolektif.
3. Memahami perbedaan tipe data primitif, abstrak dan kolektif.

BAB II

DASAR TEORI

Tipe data dalam pemrograman bisa dikelompokkan menjadi beberapa kategori, antara lain:

1. Tipe Data Primitif : Tipe data primitif adalah tipe data yang tersedia secara langsung dalam bahasa pemrograman dan biasanya direpresentasikan oleh tipe data bawaan dari komputer. Contoh tipe data primitif umum meliputi integer, float, double, char, dan boolean.
2. Tipe Data Abstrak : Tipe data abstrak adalah tipe data yang didefinisikan oleh pengguna atau programmer berdasarkan kebutuhan dan tidak tergantung pada representasi fisik di dalam komputer. Contoh tipe data abstrak meliputi array, list, stack, queue, dan tree.
3. Koleksi : Koleksi adalah kumpulan dari beberapa elemen yang terdiri dari satu atau lebih tipe data. Koleksi bisa terdiri dari tipe data primitif atau tipe data abstrak. Contoh koleksi meliputi array, vector, set, map, dan hash table.

Berikut adalah penjelasan singkat tentang masing-masing kategori tipe data:

1. Tipe Data Primitif : Tipe data primitif adalah tipe data dasar yang disediakan oleh bahasa pemrograman untuk merepresentasikan data dasar seperti angka, karakter, atau nilai kebenaran (true/false). Contoh tipe data primitif meliputi:
 - a. int: Untuk menyimpan bilangan bulat.
 - b. float dan double: Untuk menyimpan bilangan pecahan.
 - c. char: Untuk menyimpan karakter.
 - d. bool: Untuk menyimpan nilai kebenaran (true atau false).
2. Tipe Data Abstrak: Tipe data abstrak adalah tipe data yang didefinisikan oleh pengguna atau programmer dan biasanya tidak memiliki implementasi langsung di dalam bahasa pemrograman. Tipe data abstrak ini biasanya digunakan untuk menyembunyikan kompleksitas implementasi dari pengguna. Contoh tipe data abstrak meliputi:
 - a. Array: Kumpulan elemen dengan tipe data yang sama yang disusun dalam urutan tertentu.
 - b. List: Kumpulan elemen yang disusun secara linier dan memungkinkan penambahan dan penghapusan elemen.

c. Stack dan Queue: Struktur data yang memungkinkan penambahan atau penghapusan elemen sesuai dengan aturan tertentu (Stack menggunakan prinsip LIFO, sedangkan Queue menggunakan prinsip FIFO).

d. Tree: Struktur data yang terdiri dari simpul-simpul yang terhubung dan membentuk hirarki.

3. Koleksi: Koleksi adalah tipe data yang dapat menyimpan banyak elemen sekaligus. Koleksi ini bisa terdiri dari tipe data primitif atau tipe data abstrak lainnya. Contoh koleksi meliputi:

a. Array: Koleksi dari elemen-elemen dengan tipe data yang sama yang disusun dalam urutan tertentu.

b. Vector: Koleksi dinamis yang dapat menyesuaikan ukurannya saat elemen-elemen ditambahkan atau dihapus.

c. Set: Koleksi yang hanya menyimpan elemen-elemen unik tanpa adanya urutan tertentu.

d. Map atau Dictionary: Koleksi yang terdiri dari pasangan kuncinilai yang memungkinkan pengaksesan nilai berdasarkan kunci tertentu.

BAB III

GUIDED

1. Guided 1

Source code

```
#include <iostream>
using namespace std;
// Main program
int 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;
```

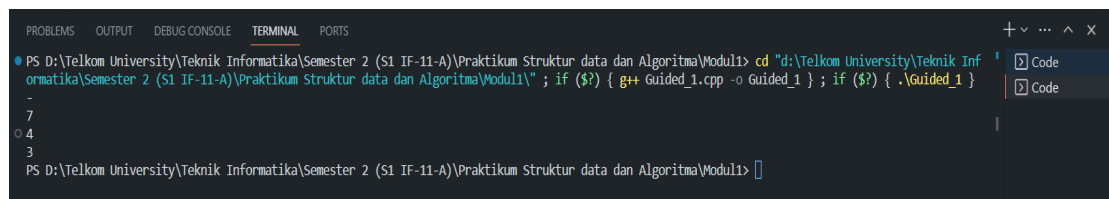
```

        break;

        // If the operator is other than +, -, * or /,
        // error message will display
        default:
            cout << "Error! operator is not correct";
        } // switch statement ends
        return 0;
    }

```

Screenshoot program



```

PS D:\Telkom University\Teknik Informatika\Semester 2 (S1 IF-11-A)\Praktikum Struktur data dan Algoritma\Modul1> cd "d:\Telkom University\Teknik Inf
ormatika\Semester 2 (S1 IF-11-A)\Praktikum Struktur data dan Algoritma\Modul1" ; if ($?) { g++ Guided_1.cpp -o Guided_1 } ; if ($?) { .\Guided_1 }
-
7
4
3
PS D:\Telkom University\Teknik Informatika\Semester 2 (S1 IF-11-A)\Praktikum Struktur data dan Algoritma\Modul1>

```

Deskripsi program

Program diatas merupakan program menghitung 2 bilangan. Pengguna diminta memasukkan salah satu operator matematika yaitu : +,-,*,/. Jika yang dimasukkan +(Penjumlahan) maka **num1 + num2**. Jika yang dimasukkan -(Pengurangan) maka **num1 – num2**. Jika yang dimasukkan *(Perkalian) maka **num1 * num2**. Jika yang dimasukkan /(Pembagian) maka **num1 / num2**. Jika pengguna memasukkan operator lain selain 4 operator tersebut maka akan error.

2. Guided 2

Source Code

```

#include <stdio.h>
// Struct
struct Mahasiswa
{
    const char *name;
    const char *address;
    int age;
};
int main()
{
    // menggunakan struct

```

```

struct Mahasiswa mhs1, mhs2;
// mengisi nilai ke struct
mhs1.name = "Gery";
mhs1.address = "Bandung";
mhs1.age = 19;
mhs2.name = "Alex";
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);

return 0;
}

```

Screenshot Program

```

PS D:\Telkom University\Teknik Informatika\Semester 2 (S1 IF-11-A)\Praktikum Struktur data dan Algoritma\Modul1> cd "d:\Telkom University\Teknik Informatika\Semester 2 (S1 IF-11-A)\Praktikum Struktur data dan Algoritma\Modul1\"; if ($?) { g++ Guided_2.cpp -o Guided_2 }; if ($?) { .\Guided_2 }
## Mahasiswa 1 ##
Nama: Gery
Alamat: Bandung
Umur: 19
## Mahasiswa 2 ##
Nama: Alex
Alamat: Surabaya
Umur: 23
PS D:\Telkom University\Teknik Informatika\Semester 2 (S1 IF-11-A)\Praktikum Struktur data dan Algoritma\Modul1>

```

Deskripsi program

Program diatas merupakan program menampilkan data mahasiswa. Pertama program membuat struct Bernama **mahasiswa** yang berisi **name**, **address** dan **age**. Lalu mendeklarasikan **mhs1** dan **mhs2** dengan tipe **mahasiswa**.

3. Guided 3

Source code

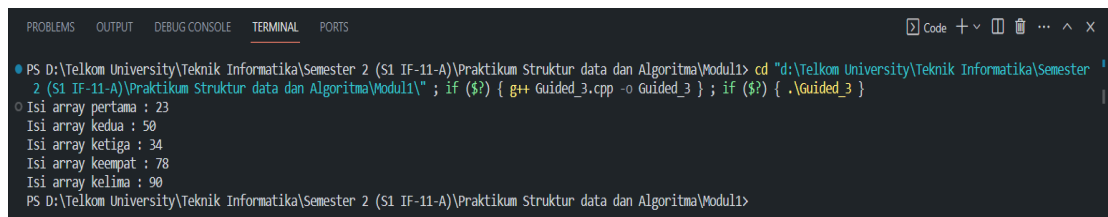
```

#include <iostream>
using namespace std;
int main()
{
    // deklarasi dan inisialisasi array
    int nilai[5];
    nilai[0] = 23;
    nilai[1] = 50;
    nilai[2] = 34;
    nilai[3] = 78;
}

```

```
nilai[4] = 90;
// mencetak array
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;
}
```

Screenshot program



```
PS D:\Telkom University\Teknik Informatika\Semester 2 (S1 IF-11-A)\Praktikum Struktur data dan Algoritma\Modul1> cd "d:\Telkom University\Teknik Informatika\Semester 2 (S1 IF-11-A)\Praktikum Struktur data dan Algoritma\Modul1" ; if ($?) { g++ Guided_3.cpp -o Guided_3 } ; if ($?) { .\Guided_3 }
Isi array pertama : 23
Isi array kedua : 50
Isi array ketiga : 34
Isi array keempat : 78
Isi array kelima : 90
PS D:\Telkom University\Teknik Informatika\Semester 2 (S1 IF-11-A)\Praktikum Struktur data dan Algoritma\Modul1>
```

Deskripsi program

Program diatas merupakan program menampilkan isi array pertama sampai kelima. Pertama deklarasikan nilai sebagai array yang berjumlah 5. Isikan array nilai tersebut dengan angka. Lalu menggunakan cout untuk menampilkan nilai tersebut.

LATIHAN KELAS - UNGUIDED

1. Unguided 1

Source code

```
#include <iostream>
#include <iomanip>

using namespace std;

// Fungsi untuk menghitung luas persegi
double hitungLuasPersegi(double sisi) {
    return sisi * sisi;
}

int main() {
    // Deklarasi variabel untuk menyimpan sisi persegi
    double sisiPersegi;

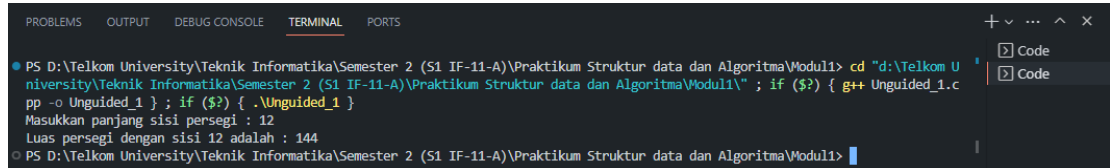
    // Meminta input dari pengguna
    cout << "Masukkan panjang sisi persegi : ";
    cin >> sisiPersegi;

    // Panggil fungsi untuk menghitung luas persegi
    double luasPersegi = hitungLuasPersegi(sisiPersegi);

    // Menampilkan hasil
    cout << "Luas persegi dengan sisi " << sisiPersegi << "
    adalah : " << luasPersegi << endl;

    return 0;
}
```

Screenshoot program



```
PS D:\Telkom University\Teknik Informatika\Semester 2 (S1 IF-11-A)\Praktikum Struktur data dan Algoritma\Modul1> cd "d:\Telkom U
niversity\Teknik Informatika\Semester 2 (S1 IF-11-A)\Praktikum Struktur data dan Algoritma\Modul1\" ; if ($?) { g++ Unguided_1.c
pp -o Unguided_1 } ; if ($?) { .\Unguided_1 }
Masukkan panjang sisi persegi : 12
Luas persegi dengan sisi 12 adalah : 144
PS D:\Telkom University\Teknik Informatika\Semester 2 (S1 IF-11-A)\Praktikum Struktur data dan Algoritma\Modul1>
```

Deskripsi program

Program diatas merupakan program menghitung luas persegi. Pertama membuat fungsi menghitung luas persegi lalu mendeklarasikan variable untuk menyimpan sisi persegi. Pengguna menginputkan nilai untuk menghitung luas persegi.

Kesimpulan

Program ini menunjukkan penggunaan tipe data primitif (double) untuk menyimpan informasi dasar (panjang sisi persegi dan luas persegi) dan melakukan operasi matematis sederhana. Input dari pengguna juga diterima dan divalidasi untuk memastikan keberhasilan program. Tipe data primitif memungkinkan program untuk melakukan perhitungan matematis dan interaksi dengan pengguna secara sederhana dan efisien. Program semacam ini mencerminkan penggunaan tipe data primitif dalam memproses informasi dasar dalam pemrograman.

2. Unguided 2

Source code

```
#include <iostream>

using namespace std;

// Struct
struct Employee
{
    const char *name;
```

```
    const char *address;
    int age;
};

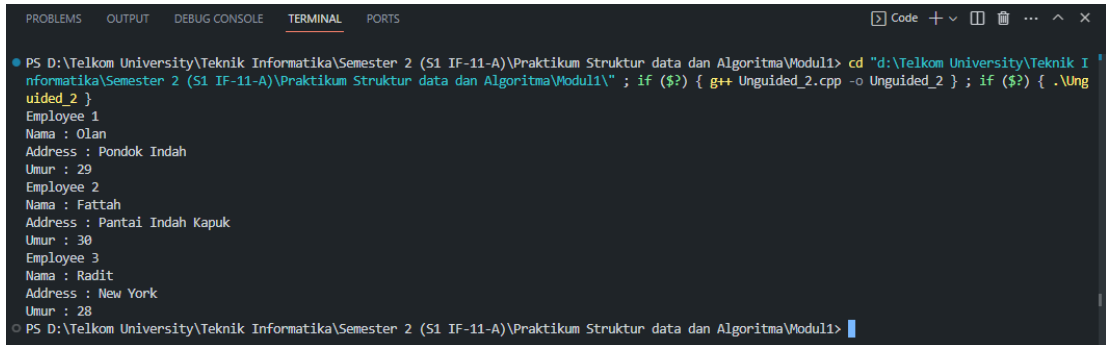
int main()
{
    // menggunakan struct
    struct Employee epl1, epl2, epl3;
    // mengisi nilai ke struct
    epl1.name = "Olan";
    epl1.address = "Pondok Indah";
    epl1.age = 29;
    epl2.name = "Fattah";
    epl2.address = "Pantai Indah Kapuk";
    epl2.age = 30;
    epl3.name = "Radit";
    epl3.address = "New York";
    epl3.age = 28;

    // mencetak isi struct
    cout << "Employee 1" << endl;
    cout << "Nama : " << epl1.name << endl;
    cout << "Address : " << epl1.address << endl;
    cout << "Umur : " << epl1.age << endl;
    cout << "Employee 2" << endl;
    cout << "Nama : " << epl2.name << endl;
    cout << "Address : " << epl2.address << endl;
    cout << "Umur : " << epl2.age << endl;
    cout << "Employee 3" << endl;
    cout << "Nama : " << epl3.name << endl;
    cout << "Address : " << epl3.address << endl;
    cout << "Umur : " << epl3.age << endl;

    return 0;
}
```

```
}
```

Screenshot program



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS D:\Telkom University\Teknik Informatika\Semester 2 (S1 IF-11-A)\Praktikum Struktur data dan Algoritma\Modul1> cd "d:\Telkom University\Teknik I
nformatika\Semester 2 (S1 IF-11-A)\Praktikum Struktur data dan Algoritma\Modul1\" ; if ($?) { g++ Unguided_2.cpp -o Unguided_2 } ; if ($?) { .\Ung
uided_2 }
Employee 1
Nama : Olan
Address : Pondok Indah
Umur : 29
Employee 2
Nama : Fattah
Address : Pantai Indah Kapuk
Umur : 30
Employee 3
Nama : Radit
Address : New York
Umur : 28
PS D:\Telkom University\Teknik Informatika\Semester 2 (S1 IF-11-A)\Praktikum Struktur data dan Algoritma\Modul1>
```

Deskripsi program

Program diatas merupakan program menampilkan data employee. Pertama program membuat struct Bernama **employee** yang berisi **name**, **address** dan **age**. Lalu mendeklarasikan **ep1**, **ep2** dan **ep3** dengan tipe **employee**.

Fungsi class dan struct

Class:

- Encapsulation (enkapsulasi): Class memungkinkan Anda menggabungkan data (atribut) dan fungsi (metode) dalam satu unit yang disebut objek. Dengan demikian, Anda dapat menyembunyikan implementasi internal dari luar dunia objek, meningkatkan abstraksi dan mengurangi kompleksitas.
- Inheritance (pewarisan): Class mendukung pewarisan, yang memungkinkan pembuatan class baru (subclass atau turunan) dengan mewarisi sifat dan perilaku dari class yang sudah ada (superclass atau induk).
- Polymorphism (polimorfisme): Class mendukung polimorfisme, yang memungkinkan objek dari class yang berbeda dapat digunakan dengan cara yang seragam melalui antarmuka yang sama.

Struct:

- Data Aggregation (penggabungan data): Struct mirip dengan class dalam hal menyatukan data, tetapi biasanya digunakan untuk menyimpan data tanpa adanya metode atau perilaku terkait. Struct biasanya digunakan untuk merepresentasikan kelompok data yang sederhana.

3. Unguided 3

Source code

```
#include <iostream>
#include <map>

using namespace std;

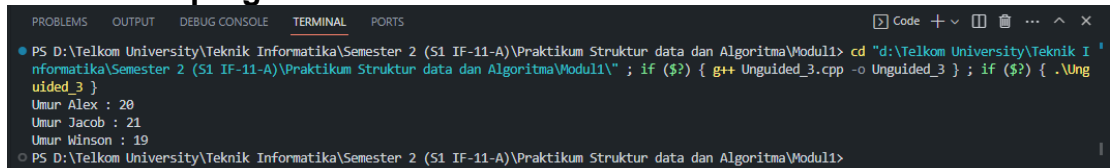
int main() {
    // Membuat map dengan kunci bertipe string dan nilai
    // bertipe int
    map<string, int> dataMap;

    // Menambahkan elemen ke dalam map
    dataMap["Alex"] = 20;
    dataMap["Jacob"] = 21;
    dataMap["Winson"] = 19;

    // Mengakses dan mencetak nilai berdasarkan kunci
    cout << "Umur Alex : " << dataMap["Alex"] << endl;
    cout << "Umur Jacob : " << dataMap["Jacob"] << endl;
    cout << "Umur Winson : " << dataMap["Winson"] << endl;

    return 0;
}
```

Screenshot program



Deskripsi program

Program diatas merupakan program membuat sebah map yang menyimpan umur beberapa orang dengan menggunakan string sebagai kunci dan integer sebagai nilai. Kemudian program mengakses dan mencetak umur masing-masing orang ke output.

Perbedaan utama antara array dan map adalah sebagai berikut:

- Indeks atau Kunci:
 - Array: Menggunakan indeks berupa bilangan bulat untuk mengakses elemen.
 - Map: Menggunakan kunci, yang dapat berupa tipe data apa pun, untuk mengakses nilai.
- Ukuran Dinamis:
 - Array: Memiliki ukuran tetap yang ditentukan pada saat deklarasi.
 - Map: Ukurannya dapat tumbuh dan menyusut secara dinamis saat elemen-elemen ditambahkan atau dihapus.

- Tipe Elemen:
Array: Elemen-elemen memiliki tipe data yang sama.
Map: Kunci dan nilai dapat memiliki tipe data yang berbeda.
- Penyimpanan Elemen:
Array: Elemen-elemen disimpan secara berurutan di dalam memori.
Map: Implementasinya mungkin menggunakan struktur data yang lebih kompleks, seperti pohon merah-hitam atau tabel hash, untuk mengoptimalkan pencarian dan pengaksesan.

BAB IV

KESIMPULAN

Dalam pemrograman, tipe data dapat dikelompokkan menjadi tipe data primitive, abstrak dan koleksi. Tipe data primitive merujuk pada tipe data dasar yang sudah didefinisikan dalam Bahasa pemrograman. Tipe data ini merupakan unit paling sederhana yang dapat digunakan untuk menyimpan nilai. Contoh tipe data primitive umum meliputi integer (bilangan bulat), float (bilangan desimal), karakter dan Boolean. Karakteristik utama dari tipe data primitive adalah bahwa mereka tidak dapat dipecah lagi menjadi tipe data yang lebih sederhana. Tipe data abstrak adalah konsep di Tingkat tinggi yang menyediakan cara untuk menggabungkan data dan fungsi terkait ke dalam suatu entitas Tunggal. Class dalam pemrograman berorientasi objek adalah contoh dari tipe data abstrak. Mereka memungkinkan penggunaan enkapsulasi, pewarisan dan polimorfisme yang membantu meningkatkan abstraksi dan kompleksitas dalam pengembangan perangkat lunak. Tipe data koleksi merujuk pada struktur data yang dapat menyimpan sejumlah elemen. Ini memungkinkan pengelolaan sekumpulan nilai dalam satu wadah. Beberapa contoh tipe data koleksi meliputi array, list, set, map, dll. Tipe data koleksi memungkinkan operasi seperti penambahan, penghapusan dan pencarian elemen.

Praktik pengkodean yang menggabungkan konsep tersebut dapat dilakukan dengan membuat program sederhana yang menggunakan map, struct, class dan tipe data primitif. Contoh, kita membuat program yang mengelola data karyawan. Kita bisa menggunakan struct untuk mempreentasikan data karyawan seperti **nama**, **Alamat** dan **umur**. Class untuk mengelola data karyawan seperti **menambah**, **mengurangkan** dan **pencarian karyawan**. Map untuk menyimpan data karyawan berdasarkan nama sebagai kunci. Dengan menggabungkan konsep ini kita dapat membuat program yang efisien dan mudah diorganisir.