LAPORAN PRAKTIKUM

MODUL I
TIPE DATA

Disusun oleh:
Syafanida Khakiki
NIM: 2311102005

Dosen Pengampu:
Muhammad Afrizal Amrustian, S. Kom., M. Kom

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

BAB I

TUJUAN PRAKTIKUM

1. Mahasiswa dapat mempelajari tipe data primitif, abstrak, dan kolektif.
2. Mahasiswa dapat memahami pengaplikasian pada tools yang digunakan.
3. Mahasiswa mengaplikasikan berbagai tipe data pada bahasa pemograman yang telah ditentukan.

BAB II

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 Primitif
2. Tipe data Abstrak
3. Tipe data Koleksi

1. Tipe data primitif

Tipe data primitif adalah tipe data yang telah ditentukan oleh sistem, tipe data primitif ini disediakan oleh banyak bahasa pemrograman, perbedaannya terletak pada jumlah bit yang dialokasikan untuk setiap bit pada tipe data primitif tergantung pada bahasa pemrograman,compiler dan sistem operasinya. Contoh tipe data primitif :

i. Integer : adalah tipe data yang digunakan untuk menyimpan bilangan bulatseperti 12, 1, 4, dan sebagainya.
ii. Float : tipe data yang digunakan untuk menyimpan bilangan desimalseperti 1.5, 2.1, 3.14, dan sebagainya.
iii. Char : berfungsi untuk menyimpan data berupa sebuah huruf. Biasanyadigunakan untuk simbol seperti A, B, C dan seterusnya
iv. Boolean : tipe data ini digunakan untuk menyimpan nilai boolean yanghanya memiliki dua nilai yaitu true dan false.

2. Tipe Data Abstrak

Tipe data abstrak atau yang biasa disebut Abstrak Data Tipe(ADT) merupakan tipe data yang dibentuk oleh programer itu sendiri. Pada tipe data abstrak bisa berisi banyak tipe data, jadi nilainya bisa lebih dari satu dan beragam tipe data. Fitur Class adalah fitur Object Oriented Program(OPP) pada bahasa C++ yang mirip dengan fitur data structures Struct pada bahasa C. Keduanya berfungsi untuk membungkus tipe data di dalamnya sebagai anggota. menurut learn.microsoft.com perbedaan antara Struct dan Class adalah pada akses defaultnya dimana Struct bersifat public dan Class bersifat private.

3. Tipe Data Koleksi

Tipe data koleksi (Collection Data Type) adalah tipe data yang digunakan

untuk mengelompokkan dan menyimpan beberapa nilai atau objek secara bersamaan. Tipedata koleksi memungkinkan Anda menyimpan, mengelola, dan mengakses sejumlahbesar data dengan cara yang terstruktur. Ada beberapa tipe data koleksi yang umum digunakan dalam pemrograman, dan di antaranya adalah:

a. Array : Array adalah struktur data statis yang menyimpan elemen-elemendengan tipe data yang sama. Elemen-elemen tersebut dapat diakses dengan menggunakan indeks. Array memiliki ukuran tetap yang ditentukan saat deklarasi.

b. Vector : Vector adalah Standard Template Library (STL) jika di dalam C/C++ memiliki bentuk std::vector . Umumnya, vector mirip seperti arrayyang memiliki kemampuan 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

c. 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 Treekhususnya Red-Black Tree.

BAB III

GUIDED

1. **Guided 1**

Source code :

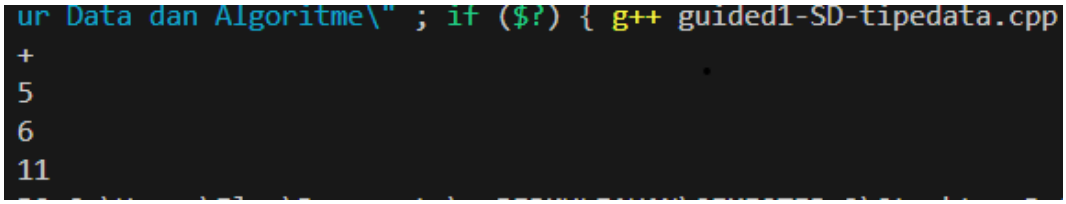
```
#include <iostream>
using namespace std;

int main()
{
    char op;
    float num1, num2;
    cin >> op;
    cin >> num1 >> num2;

switch (op)
{
case '+':
cout << num1 + num2;
break;
case '-':
cout << num1 - num2;
break;
case '*':
cout << num1 * num2;
break;
case '/':
cout << num1 / num2;
break;
default:
cout << "Error! operator is not correct";
}
return 0;
}
```

Screenshoot program :

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

int main()
{
   char op;
   float num1, num2;
   cin >> op;
   cin >> num1 >> num2;
```

```
switch (op)
{

case '+':
cout << num1 + num2;
break;

case '-':
cout << num1 - num2;
break;

case '*':
cout << num1 * num2;
break;

case '/':
cout << num1 / num2;
break;
default:
cout << "Error! operator is not correct";
}
return 0;
}
```

Output :



Deskripsi program :

- Program dimulai dengan mengimpor library iostream untuk mengaktifkan fungsi input-output standar C++.
- Deklarasi variabel op sebagai char untuk menyimpan operator matematika.
- Deklarasi variabel num1 dan num2 sebagai float untuk menyimpan dua bilangan yang akan dioperasikan.
- Membaca input operator matematika (op), num1, dan num2 dari pengguna menggunakan cin.
- Program menggunakan struktur switch-case untuk memeriksa nilai op.
- Jika op adalah '+', program akan menampilkan hasil penjumlahan num1 dan num2.

- Jika op adalah '-', program akan menampilkan hasil pengurangan num1 dan num2.
- Jika op adalah '*', program akan menampilkan hasil perkalian num1 dan num2.
- Jika op adalah '/', program akan menampilkan hasil pembagian num1 dan num2.
- Jika op tidak sama dengan salah satu operator yang valid, program akan menampilkan pesan kesalahan "Error! operator is not correct".

2. Guided 2

Source code :

```
#include <stdio.h>

struct Mahasiswa

{
    const char *name;
    const char *address;
    int age;
};


int main()
{

struct Mahasiswa mhs1, mhs2;
mhs1.name = "Dian";
mhs1.address = " Mataram";
mhs1.age = 22;
mhs2.name = "Bambang";
mhs2.address = "Surabaya";
mhs2.age = 23;


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;
}
```

Screenshoot program :

```
#include <iostream>
#include <string>

using namespace std;

// Definisi struct data menu
struct Menu {
    string nama;
    int harga;
};

int main() {
    // Deklarasi array menu
    Menu menu_burger[4] = {
        {"Beef Burger", 30000},
        {"Chicken Burger", 25000},
        {"Egg Burger", 20000},
        {"Cheese Burger", 15000}
    };

    // Menampilkan menu
    cout << "Menu Burger:" << endl;
    for (int i = 0; i < 4; ++i) {
        cout << i+1 << ". " << menu_burger[i].nama << " - Rp " <<
menu_burger[i].harga << endl;
    }

    // Meminta input untuk pilihan menu
    cout << "Masukkan nomor menu yang dipilih (1-4): ";
    int pilihan;
    cin >> pilihan;

    // Validasi pilihan menu
    if (pilihan < 1 || pilihan > 4) {
        cout << "Pilihan menu tidak valid!" << endl;
        return 1; // Keluar dari program dengan kode error
```

```
    }

    // Menghitung total harga
    int total_harga = menu_burger[pilihan - 1].harga;

    // Menampilkan total harga
    cout << "Total harga: Rp " << total_harga << endl;

    return 0;
}
```

Output :

```
Menu Burger:
1. Beef Burger - Rp 30000
2. Chicken Burger - Rp 25000
3. Egg Burger - Rp 20000
4. Cheese Burger - Rp 15000
Masukkan nomor menu yang dipilih (1-4): 1
Total harga: Rp 30000
```

Deskripsi program :

3. **Guided 3**

Source code :

```
#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;
}
```

Screenshoot program :

```
#include <stdio.h>

struct Mahasiswa

{
    const char *name;
    const char *address;
    int age;
};


int main()
{

struct Mahasiswa mhs1, mhs2;
mhs1.name = "Dian";
mhs1.address = " Mataram";
mhs1.age = 22;
mhs2.name = "Bambang";
mhs2.address = "Surabaya";
mhs2.age = 23;


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;
}
```

Output :

```
Isi array pertama :23
Isi array kedua:50
Isi array ketiga:34
Isi array keempat:78
Isi array kelima:90
```

Deskripsi program :

- Deklarasi array bernama nilai yang dapat menyimpan 5 elemen bertipe integer.
- Menginisialisasi nilai-nilai pada masing-masing elemen array nilai dengan menggunakan indeks array. Pada contoh ini, nilai-nilai yang disimpan adalah 23, 50, 34, 78, dan 90.
- Menampilkan isi dari setiap elemen array menggunakan operator indeks [] dengan mengakses elemen array secara langsung. Setiap elemen array di-print satu per satu ke layar dengan menggunakan objek cout dan operator <<.
- Setelah semua elemen array ditampilkan, program mengembalikan nilai 0 untuk menandakan bahwa program berakhir dengan sukses.

LATIHAN KELAS - UNGUIDED

1. Unguided 1

Source code :

```
#include <iostream>
using namespace std;
int a, t1, t2;
int luas_alas (int a, int t1){
 return 0.5*a*t1;
}
int keliling_alas (int a){
 return 3*a;
}
int luas_prisma (int a, int t1, int t2){
 return 2*luas_alas(a,t1) + keliling_alas(a)*t2;
}
int main()
{
 cout << "Masukkan Panjang Sisi Alas Prisma (cm): " ;
 cin>> a;
 cout << "Masukkan Tinggi Segitiga Alas Prisma (cm): " ;
 cin>> t1;
 cout << "Masukkan Tinggi Prisma (cm): " ;
 cin>> t2;
 cout << "Luas Alas : " << luas_alas(a,t1) << " cm2" <<endl;
 cout << "Keliling : " << keliling_alas(a) << " cm2" <<endl;
 for (int i = 0; i < 5; i++) {
 cout << "LUAS PRISMA dengan tinggi " <<t2 <<" : " <<
luas_prisma(a,t1,t2) << " cm2" <<endl;
 t2 += 3;
 }
 return 0;
}
```

Screenshoot program :

```
#include <iostream> //Syafanida Khakiki, 2311102005
using namespace std;
int a, t1, t2;
int luas_alas (int a, int t1){
 return 0.5*a*t1;
}
int keliling_alas (int a){
```

```
 return 3*a;
}
int luas_prisma (int a, int t1, int t2){
 return 2*luas_alas(a,t1) + keliling_alas(a)*t2;
}
int main()
{
 cout << "Masukkan Panjang Sisi Alas Prisma (cm): " ;
 cin>> a;
 cout << "Masukkan Tinggi Segitiga Alas Prisma (cm): " ;
 cin>> t1;
 cout << "Masukkan Tinggi Prisma (cm): " ;
 cin>> t2;
 cout << "Luas Alas : " << luas_alas(a,t1) << " cm2" <<endl;
 cout << "Keliling : " << keliling_alas(a) << " cm2" <<endl;
 for (int i = 0; i < 5; i++) {
 cout << "LUAS PRISMA dengan tinggi " <<t2 <<" : " <<
luas_prisma(a,t1,t2) << " cm2" <<endl;
 t2 += 3;
 }
 return 0;
}
```

Output :

```
Masukkan Panjang Sisi Alas Prisma (cm): 15
Masukkan Tinggi Segitiga Alas Prisma (cm): 20
Masukkan Tinggi Prisma (cm): 25
Luas Alas : 150 cm2
Keliling : 45 cm2
LUAS PRISMA dengan tinggi 25 : 1425 cm2
LUAS PRISMA dengan tinggi 28 : 1560 cm2
LUAS PRISMA dengan tinggi 31 : 1695 cm2
LUAS PRISMA dengan tinggi 34 : 1830 cm2
LUAS PRISMA dengan tinggi 37 : 1965 cm2
```

Deskripsi program :

- Mendeklarasikan variabel global a, t1, dan t2 yang digunakan untuk menyimpan panjang sisi alas prisma tinggi segitiga alas prisma, dan tinggi prisma.

- Mendefinisikan fungsi luas_alas yang mengambil dua parameter a dan t1 yang merupakan panjang sisi alas dan tinggi segitiga, dan mengembalikan nilai luas alas prisma segitiga dengan rumus (1/2) * a * t1.
- Mendefinisikan fungsi keliling_alas yang mengambil satu parameter a yang merupakan panjang sisi alas, dan mengembalikan nilai keliling alas prisma segitiga dengan rumus 3 * a.
- Mendefinisikan fungsi luas_prisma yang mengambil tiga parameter a, t1, dan t2 yang merupakan panjang sisi alas, tinggi segitiga, dan tinggi prisma, dan mengembalikan nilai luas prisma dengan rumus 2 * luas_alas(a, t1) + keliling_alas(a) * t2.
- Di dalam fungsi main, program meminta pengguna untuk memasukkan panjang sisi alas prisma, tinggi segitiga alas prisma, dan tinggi prisma menggunakan cin.

- Program mencetak luas dan keliling alas prisma segitiga yang dihitung menggunakan fungsi luas_alas dan keliling_alas.
- Program kemudian melakukan perulangan for sebanyak lima kali dengan variabel iterasi i dari 0 sampai 4.
- Di setiap iterasi, program mencetak luas prisma dengan tinggi yang berbeda-beda. Nilai tinggi prisma (t2) diinkremen setiap iterasi dengan nilai 3.
- Program mengembalikan nilai 0 untuk menandakan bahwa program berakhir dengan sukses.

2. Unguided 2

Kelas (class) dan struktur (struct) dalam pemrograman berorientasi objek (OOP) C++ memiliki fungsi utama yang sama: **mendefinisikan struktur data** dan **mengelompokkan data terkait**.

Perbedaan :

- **Class**: Lebih kompleks, mendukung enkapsulasi, pewarisan, dan polimorfisme. Cocok untuk situasi yang membutuhkan keamanan, modularitas, dan kode yang fleksibel.
- **Struct**: Lebih sederhana, akses data langsung, tidak mendukung pewarisan dan polimorfisme. Cocok untuk situasi yang membutuhkan struktur data sederhana dan efisiensi memori.

1). Contoh Program menggunakan Class :

Source code :

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

class Kucing {
public:

    string jenis;
    string warna;
    int berat;

    // Metode (fungsi) dari class Mobil
    void info() {
        cout << "Kucing " << jenis << " warna " << warna << " berat
badan " << berat << "kg" <<endl;
    }
};

int main() {

    Kucing kucing1;

    // Mengatur nilai atribut objek
    kucing1.jenis = "British Short Hair";
    kucing1.warna = "Abu abu";
    kucing1.berat = 4;

    // Memanggil metode objek
    kucing1.info();

    return 0;
}
```

Screenshoot program :

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

class Kucing {
public:

    string jenis;
    string warna;
```

```
    int berat;

    // Metode (fungsi) dari class Mobil
    void info() {
        cout << "Kucing " << jenis << " warna " << warna << " berat
badan " << berat << "kg" <<endl;
    }
};

int main() {

    Kucing kucing1;

    // Mengatur nilai atribut objek
    kucing1.jenis = "British Short Hair";
    kucing1.warna = "Abu abu";
    kucing1.berat = 4;

    // Memanggil metode objek
    kucing1.info();

    return 0;
}
```

Output :

```
Kucing British Short Hair warna Abu abu berat badan 4kg
```

Deskripsi program :

- Deklarasi Kelas Kucing (class Kucing): Pada bagian ini, sebuah class bernama Kucing dideklarasikan. Kucing memiliki tiga atribut: jenis, warna, dan berat. Atribut-atribut ini mendefinisikan karakteristik dari objek kucing yang akan dibuat berdasarkan class Kucing.
- Definisi Metode info(): Di dalam class Kucing, terdapat sebuah metode bernama info(). Metode ini bertugas untuk menampilkan informasi tentang kucing, termasuk jenis, warna, dan berat badannya.
- Fungsi main(): Pada fungsi main(), objek kucing (kucing1) dibuat berdasarkan class Kucing. Objek ini memiliki atribut yang ditentukan, yaitu jenis kucing, warna, dan berat badannya.

- Inisialisasi Atribut Objek: Nilai-nilai atribut dari objek kucing1 diatur menggunakan operator titik (.). Ini dilakukan dengan cara menyediakan nilai untuk setiap atribut dari objek kucing.
- Memanggil Metode info(): Setelah atribut objek diatur, metode info() dipanggil pada objek kucing1. Hal ini menyebabkan program mencetak informasi tentang kucing tersebut ke layar.

2). Contoh Program menggunakan Struct :

Source code :

```
#include <iostream>
#include <string>

using namespace std;

// Definisi struct data menu
struct Menu {
    string nama;
    int harga;
};

int main() {
    // Deklarasi array menu
    Menu menu_burger[4] = {
        {"Beef Burger", 30000},
        {"Chicken Burger", 25000},
        {"Egg Burger", 20000},
        {"Cheese Burger", 15000}
    };

    // Menampilkan menu
    cout << "Menu Burger:" << endl;
    for (int i = 0; i < 4; ++i) {
        cout << i+1 << ". " << menu_burger[i].nama << " - Rp " <<
menu_burger[i].harga << endl;
    }

    // Meminta input untuk pilihan menu
    cout << "Masukkan nomor menu yang dipilih (1-4): ";
    int pilihan;
    cin >> pilihan;

    // Validasi pilihan menu
    if (pilihan < 1 || pilihan > 4) {
        cout << "Pilihan menu tidak valid!" << endl;
        return 1; // Keluar dari program dengan kode error
    }
```

```
    // Menghitung total harga
    int total_harga = menu_burger[pilihan - 1].harga;

    // Menampilkan total harga
    cout << "Total harga: Rp " << total_harga << endl;

    return 0;
}
```

Screenshoot program :

```
#include <iostream>
#include <string>

using namespace std;

// Definisi struct data menu
struct Menu {
    string nama;
    int harga;
};

int main() {
    // Deklarasi array menu
    Menu menu_burger[4] = {
        {"Beef Burger", 30000},
        {"Chicken Burger", 25000},
        {"Egg Burger", 20000},
        {"Cheese Burger", 15000}
    };

    // Menampilkan menu
    cout << "Menu Burger:" << endl;
    for (int i = 0; i < 4; ++i) {
        cout << i+1 << ". " << menu_burger[i].nama << " - Rp " <<
menu_burger[i].harga << endl;
    }

    // Meminta input untuk pilihan menu
    cout << "Masukkan nomor menu yang dipilih (1-4): ";
    int pilihan;
```

```
    cin >> pilihan;

    // Validasi pilihan menu
    if (pilihan < 1 || pilihan > 4) {
        cout << "Pilihan menu tidak valid!" << endl;
        return 1; // Keluar dari program dengan kode error
    }

    // Menghitung total harga
    int total_harga = menu_burger[pilihan - 1].harga;

    // Menampilkan total harga
    cout << "Total harga: Rp " << total_harga << endl;

    return 0;
}
```

Output :

```
Menu Burger:
1. Beef Burger - Rp 30000
2. Chicken Burger - Rp 25000
3. Egg Burger - Rp 20000
4. Cheese Burger - Rp 15000
Masukkan nomor menu yang dipilih (1-4): 1
Total harga: Rp 30000
```

Deskripsi program :

- Deklarasi Array Menu Burger: Setelah mendefinisikan struct Menu, program mendeklarasikan sebuah array bernama menu_burger yang berisi empat menu burger. Setiap elemen array menu_burger berupa objek Menu yang menyimpan nama dan harga menu.
- Menampilkan Menu: Program kemudian menampilkan menu burger beserta harganya ke layar. Ini dilakukan dengan menggunakan perulangan for untuk mengakses setiap elemen dalam array menu_burger dan mencetak nama dan harga menu.
- Meminta Input Pengguna: Setelah menampilkan menu, program meminta pengguna untuk memasukkan nomor menu yang dipilih. Nomor menu yang valid adalah angka antara 1 hingga 4, sesuai dengan jumlah menu yang tersedia.

- Validasi Input Pengguna: Program melakukan validasi terhadap input pengguna. Jika pengguna memasukkan nomor menu yang tidak valid, program memberikan pesan kesalahan dan keluar dari program dengan kode error.
- Menghitung Total Harga: Jika nomor menu yang dipilih valid, program mengambil harga menu yang sesuai dari array menu_burger dan menghitung total harga berdasarkan pilihan pengguna.
- Menampilkan Total Harga: Akhirnya, program menampilkan total harga yang harus dibayar pengguna ke layar.

3. Unguided 3

Source code :

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

using namespace std;

int main() {
    // Membuat map dengan key bertipe string dan value bertipe int
    map<string, int> data;

    // Menambahkan elemen ke dalam map
    data["apel"] = 10;
    data["jeruk"] = 20;
    data["pisang"] = 15;

    // Mengakses nilai menggunakan key
    cout << "Jumlah apel: " << data["apel"] << endl;
    cout << "Jumlah jeruk: " << data["jeruk"] << endl;
    cout << "Jumlah pisang: " << data["pisang"] << endl;

    return 0;
}
```

Screenshoot program :

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

using namespace std;

int main() {
```

```
    // Membuat map dengan key bertipe string dan value bertipe int
    map<string, int> data;

    // Menambahkan elemen ke dalam map
    data["apel"] = 10;
    data["jeruk"] = 20;
    data["pisang"] = 15;

    // Mengakses nilai menggunakan key
    cout << "Jumlah apel: " << data["apel"] << endl;
    cout << "Jumlah jeruk: " << data["jeruk"] << endl;
    cout << "Jumlah pisang: " << data["pisang"] << endl;

    return 0;
}
```

Output :

```
Jumlah apel: 10
Jumlah jeruk: 20
Jumlah pisang: 15
```

Deskripsi program :

- Program membuat sebuah map dengan tipe data key bertipe string dan tipe data value bertipe int. Map ini disebut data.
- Tiga elemen ditambahkan ke dalam map menggunakan operator []. Setiap elemen memiliki key yang unik (nama buah) dan value yang menyatakan jumlah buah tersebut.
- Kemudian, program mengakses nilai dari map menggunakan key. Dalam hal ini, program mencetak jumlah apel, jeruk, dan pisang yang disimpan dalam map.
- Hasil keluaran dari program tersebut akan menampilkan jumlah buah apel, jeruk, dan pisang yang telah ditambahkan ke dalam map sebelumnya.

Perbedaan Map dengan Array :

Array:

- **Penyimpanan berdasarkan indeks**: Elemen dalam array disimpan secara berurutan berdasarkan indeks numerik (biasanya dimulai dari 0). Akses ke elemen dilakukan menggunakan indeks ini.
- **Ukuran tetap**: Ukuran array biasanya ditentukan saat pembuatan dan tidak dapat diubah secara dinamis selama program berjalan.
- **Akses cepat berdasarkan indeks**: Mengakses elemen dalam array berdasarkan indeks umumnya lebih cepat dibandingkan struktur data lain, terutama untuk elemen yang berdekatan dalam urutan penyimpanan.
- **Cocok untuk data homogen**: Array ideal untuk menyimpan data dengan tipe yang sama dan diakses secara berurutan.

Map:

- **Penyimpanan berdasarkan key**: Elemen dalam map disimpan berdasarkan key yang unik (biasanya berupa string atau tipe data lain yang dapat berfungsi sebagai pengenal). Akses ke elemen dilakukan menggunakan key ini.
- **Ukuran dinamis**: Map dapat secara otomatis bertambah atau berkurang ukurannya saat elemen ditambahkan atau dihapus.
- **Akses berdasarkan key**: Akses ke elemen dalam map dilakukan menggunakan key, bukan indeks numerik. Ini memungkinkan pencarian elemen yang lebih fleksibel dan efisien, terutama jika Anda tidak mengetahui indeksnya sebelumnya.
- **Cocok untuk data heterogen**: Map cocok untuk menyimpan data dengan tipe yang berbeda-beda, di mana setiap elemen diidentifikasi dan diakses menggunakan key unik.

BAB IV

KESIMPULAN

- **Tipe Data Primitif:**

Definisi: Tipe data primitif adalah tipe data bawaan yang didefinisikan oleh bahasa pemrograman dan tidak dapat dipecah lagi menjadi tipe data yang lebih sederhana.

Contoh: Contoh tipe data primitif meliputi integer, floating point numbers, character, boolean, dan sebagainya.

Karakteristik:

Representasi langsung dari nilai-nilai dasar dalam pemrograman.

Digunakan untuk menyimpan data sederhana seperti bilangan, karakter, atau keadaan logis.

- **Tipe Data Abstrak:**

Definisi: Tipe data abstrak (ADT) adalah tipe data yang didefinisikan oleh pengguna berdasarkan perilaku atau operasi-operasi tertentu, tanpa mengungkapkan implementasi internalnya.

Contoh: Stack, queue, linked list, tree, dan graph adalah contoh tipe data abstrak.

Karakteristik:

Didefinisikan oleh operasi-operasi yang dapat dilakukan padanya.

Implementasi internalnya tidak terlihat oleh pengguna.

Membantu dalam mengorganisir dan memanipulasi data dengan cara yang terstruktur.

- **Tipe Data Koleksi:**

Definisi: Tipe data koleksi adalah tipe data yang digunakan untuk menyimpan kumpulan elemen, baik dalam urutan tertentu (seperti array, vector, linked list) atau dalam bentuk pasangan key-value (seperti map).

Contoh: Array, vector, map, struct, class, dan sebagainya adalah contoh tipe data koleksi.

Karakteristik:

Memiliki berbagai struktur dan perilaku yang sesuai dengan kebutuhan pemrograman.

Memungkinkan akses, penambahan, dan penghapusan elemen-elemen dari kumpulan data.

DAFTAR PUSTAKA

Karumanchi, N. (2016). *Data Structures and algorithms made easy: Concepts,problems, Interview Questions*. CareerMonk Publications.

TylerMSFT. (n.d.). Collections (C++/CX). diakses dari https://learn.microsoft.com/en-us/cpp/cppcx/collections-c-cx?view=msvc-170

Asisten Praktikum (2024). Tipe Data. Learning Management System