

**LAPORAN PRAKTIKUM STRUKTUR DATA DAN
PEMROGRAGAMAN**

**MODUL 1
TIPE DATA**



Disusun oleh :

Mohammad Harits Tantowi

2311102016

IF – 11 – A

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

BAB II

DASAR TEORI

Tipe data adalah adalah sebuah pengklasifikasian data berdasarkan jenis data tersebut. Tipe data dibutuhkan agar kompiller 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

TIPE DATA PRIMITIF

Tipe data primitif adalah tipe data yang sudah 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 adalah :

- a. Int : adalah tipe data yang digunakan untuk menyimpan bilangan bulat seperti 12, 1, 4, dan sebagainya.
- b. Float : tipe data yang digunakan untuk menyimpan bilangan desimal seperti 1.5, 2.1, 3.14, dan sebagainya.
- c. Char : berfungsi untuk menyimpan data berupa sebuah huruf. Biasanya digunakan untuk simbol seperti A, B, C dan seterusnya
- d. Boolean : tipe data ini digunakan untuk menyimpan nilai boolean yang hanya memiliki dua nilai yaitu true dan false

TIPE DATA ABSTRAK

Tipe data abstrak atau yang biasa disebut Abstrak Data Tipe (ADT) merupakan tipe data yang dibentuk oleh programmer 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 (OOP) 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](https://learn.microsoft.com/en-us/cpp/faq/struct-vs-class) perbedaan antara Struct dan Class adalah pada akses defaultnya dimana Struct bersifat public dan Class bersifat private.

TIPE DATA KOLEKSI

Tipe data koleksi (Collection Data Type) adalah tipe data yang digunakan untuk mengelompokkan dan menyimpan beberapa nilai atau objek secara bersamaan. Tipe data koleksi memungkinkan Anda menyimpan, mengelola, dan mengakses sejumlah besar 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-elemen dengan 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 array yang memiliki kemampuan untuk menyimpan data dalam bentuk elemenelemen 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 Tree khususnya Red-Black Tree.

BAB III

GUIDED

Guided I

SOURCE CODE

```
#include <iostream>

using namespace std;

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

    cout << "masukan angka pertama (+, -, *, /): ";
    cin >> op;

    if (cin.fail() || (op != '+' && op != '-' && op != '*' && op != '/')) {
        cerr << "Kesalahan! Operator tidak valid ." << endl;
        return 1;
    }

    cout << "masukan angka pertama: ";
    cin >> num1;

    if (cin.fail()) {
        cerr << "Kesalahan! Operator tidak valid." << endl;
        return 1;
    }

    cout << "masukan angka ke dua: ";
    cin >> num2;

    if (cin.fail()) {
        cerr << "Kesalahan! Operator tidak valid." << endl;
        return 1;
    }

    switch (op) {
        case '+':
            cout << num1 + num2;
            break;
        case '-':
            cout << num1 - num2;
            break;
        case '*':
            cout << num1 * num2;
            break;
```

```

        case '/':
            if (num2 == 0) {
                cerr << "Kesalahan! Pembagian dengan nol tidak diperbolehkan."
<< endl;
                return 1;
            }
            cout << num1 / num2;
            break;
        default:
            cerr << "Kesalahan! Operator tidak dikenal." << endl;
            return 1;
    }

    return 0;
}

```

SCREENSHOOT PROGRAM

The screenshot shows the Visual Studio Code interface with a C++ file named `strukturdata_2.cpp` open. The code defines a simple calculator with a `main` function that prompts the user for two numbers and an operator. It includes error handling for invalid operators and division by zero. The terminal at the bottom shows the command to compile and run the program, followed by the user input: `masukan angka pertama (+, -, *, /): /` and `masukan angka ke dua: 2`. The program output shows the result of the division: `6`.

```

D:\praktikum> strukturdata teori > strukturdata_2.cpp
5 int main() {
6     float num1, num2;
7
8     cout << "masukan angka pertama (+, -, *, /): ";
9     cin >> op;
10
11     if (cin.fail() || (op != '+' && op != '-' && op != '*' && op != '/')) {
12         cerr << "Kesalahan! Operator tidak valid." << endl;
13         return 1;
14     }
15
16     cout << "masukan angka pertama: ";
17     cin >> num1;
18
19     if (cin.fail()) {
20         cerr << "Kesalahan! Operator tidak valid." << endl;
21         return 1;
22     }
23
24     cout << "masukan angka ke dua: ";
25     cin >> num2;
26
27     if (cin.fail()) {
28         cerr << "Kesalahan! Operator tidak valid." << endl;
29         return 1;
30     }
31
32     switch (op) {
33         case '+':
34             cout << num1 + num2;
35             break;
36         case '-':
37             cout << num1 - num2;
38             break;
39         case '*':
40             cout << num1 * num2;
41             break;
42         case '/':
43             if (num2 == 0) {
44                 cerr << "Kesalahan! Pembagian dengan nol tidak diperbolehkan."
45                 << endl;
46                 return 1;
47             }
48             cout << num1 / num2;
49             break;
50         default:
51             cerr << "Kesalahan! Operator tidak dikenal." << endl;
52             return 1;
53     }
54
55     return 0;
56 }

```

```

PS D:\praktikum\strukturdata teori> cd "d:\praktikum\strukturdata teori\" ; if ($?) { g++ strukturdata_2.cpp -o strukturdata_2 } ; if ($?) { .\strukturdata_2 }
masukan angka pertama (+, -, *, /): /
masukan angka ke dua: 2
6
PS D:\praktikum\strukturdata teori>

```

DESKRIPSI PROGRAM

Program ini adalah sebuah kalkulator sederhana yang mampu melakukan operasi penjumlahan, pengurangan, perkalian, dan pembagian. Berikut adalah deskripsi setiap bagian dari program:

- Deklarasi variabel: Program ini menggunakan tiga variabel, yaitu `op` (operator), `num1` (angka pertama), dan `num2` (angka kedua).

- Input operator: Pengguna diminta untuk memasukkan operator yang ingin digunakan (misalnya +, -, *, atau /). Jika input tidak valid (bukan salah satu dari operator yang diperbolehkan), program akan menampilkan pesan kesalahan dan berhenti.
- Input angka pertama: Pengguna diminta untuk memasukkan angka pertama yang akan digunakan dalam operasi. Jika input tidak valid (bukan angka), program akan menampilkan pesan kesalahan dan berhenti.
- Input angka kedua: Pengguna diminta untuk memasukkan angka kedua yang akan digunakan dalam operasi. Jika input tidak valid (bukan angka), program akan menampilkan pesan kesalahan dan berhenti.
- Proses operasi: Program akan menjalankan operasi yang dipilih berdasarkan operator yang diinputkan oleh pengguna. Jika operator adalah / dan angka kedua adalah 0, program akan menampilkan pesan kesalahan dan berhenti.
- Output hasil: Program akan menampilkan hasil dari operasi yang telah dilakukan.

Untuk menjalankan program, pengguna hanya perlu mengkompilasi dan menjalankan file yang berisi kode program ini. Program akan menampilkan pesan yang menginstruksikan pengguna untuk memasukkan operator, angka pertama, dan angka kedua, dan setelah itu akan menampilkan hasil dari operasi yang telah dilakukan.

GUIDED II

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 = "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);
return 0;
}

```

SCREENSHOOT PROGRAM

The screenshot shows a Visual Studio IDE with a C++ project named 'strukturdata_2.cpp'. The source code defines a struct 'Mahasiswa' with members 'name', 'address', and 'age'. It then declares two variables, 'mhs1' and 'mhs2', and assigns them values. The program uses 'printf' to display the details of both students. The terminal output shows the execution results, displaying the name, address, and age for both 'Mahasiswa 1' and 'Mahasiswa 2'.

```

D:\praktikum> strukturdata teori > strukturdata_2.cpp
4 {
5     int age;
6 };
7
8 int main ()
9 {
10 // menggunakan struct
11 struct Mahasiswa mhs1,mhs2;
12 // mengisi nilai ke struct
13 mhs1.name = "Dian";
14 mhs1.address = "Mataram";
15 mhs1.age = 22;
16 mhs2.name = "Bambang";
17 mhs2.address = "Surabaya";
18 mhs2.age = 23;
19 // mencetak isi struct
20 printf("## Mahasiswa 1 ##\n");
21 printf("Nama: %s\n", mhs1.name);
22 printf("Alamat: %s\n", mhs1.address);
23 printf("Umur: %d\n", mhs1.age);
24 printf("## Mahasiswa 2 ##\n");
25 printf("Nama: %s\n", mhs2.name);
26 printf("Alamat: %s\n", mhs2.address);
27 printf("Umur: %d\n", mhs2.age);
28 return 0;
29 }
30 }

PS D:\praktikum\strukturdata teori> cd "d:\praktikum\strukturdata teori\" ; if ($?) { g++ strukturdata_2.cpp -o strukturdata_2 } ; if ($?) { .\strukturdata_2 }
## Mahasiswa 1 ##
Nama: Dian
Alamat:Mataram
Umur: 22
## Mahasiswa 2 ##
Nama: Bambang
Alamat: Surabaya
Umur: 23
PS D:\praktikum\strukturdata teori>

```

DESKRIPSI

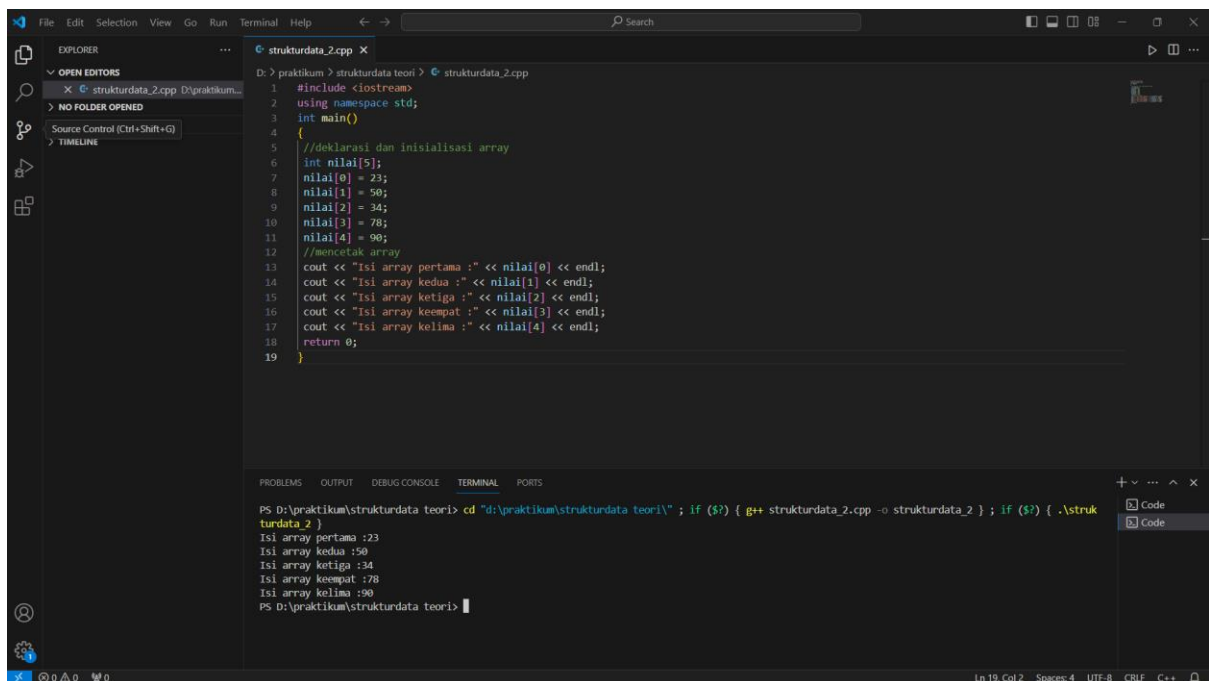
Baris `#include <stdio.h>` menyertakan pustaka input/output standar yang diperlukan untuk operasi seperti `printf()` dan `scanf()`. Program kemudian mendefinisikan sebuah struct yang disebut "Mahasiswa" dengan tiga anggota: nama, penunjuk ke string yang berisi alamat siswa, alamat, penunjuk ke string yang berisi alamat siswa, dan umur, tipe data integer untuk menyimpan usia siswa. Struktur Mahasiswa `mhs1`, `mhs2`; baris mendeklarasikan dua variabel `mhs1` dan `mhs2` bertipe "Mahasiswa". Program kemudian akan mengisi nilai anggota struct `mhs1` dan `mhs2` dengan nama, alamat, dan umur masing-masing siswa. Baris `printf()` digunakan untuk mencetak anggota struct `mhs1` dan `mhs2` ke layar.

GUIDED III

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;
}
```

SCREENSHOOT CODE



DESKRIPSI PROGRAM

Program ini mendeklarasikan array nilai dengan 5 elemen dan menginisialisasi setiap elemen dengan nilai integer yang berbeda. Kemudian, program menggunakan loop for untuk mencetak nilai setiap elemen array ke layar. Program ini menunjukkan cara sederhana untuk bekerja dengan array dalam bahasa C++ dan dapat dimodifikasi untuk kebutuhan yang lebih kompleks, seperti menambahkan lebih banyak elemen, membaca data dari input pengguna, atau melakukan operasi lain pada elemen array.

BAB IV

UNGUIDED

UNGUIDED I

SOURCE CODE

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

int main() {
    int bilangan1, bilangan2, hasilTambah, hasilKurang, hasilKali, hasilBagi;

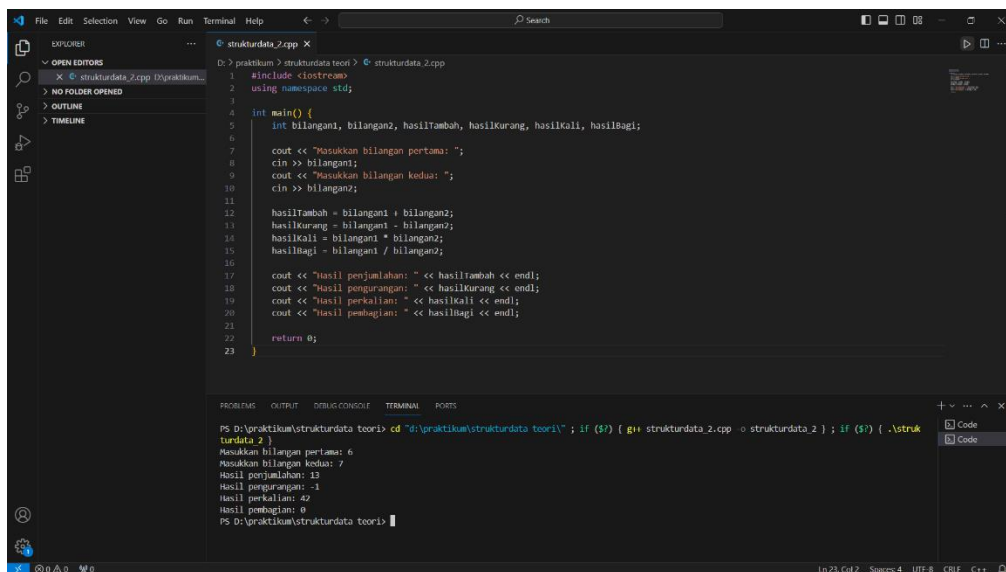
    cout << "Masukkan bilangan pertama: ";
    cin >> bilangan1;
    cout << "Masukkan bilangan kedua: ";
    cin >> bilangan2;

    hasilTambah = bilangan1 + bilangan2;
    hasilKurang = bilangan1 - bilangan2;
    hasilKali = bilangan1 * bilangan2;
    hasilBagi = bilangan1 / bilangan2;

    cout << "Hasil penjumlahan: " << hasilTambah << endl;
    cout << "Hasil pengurangan: " << hasilKurang << endl;
    cout << "Hasil perkalian: " << hasilKali << endl;
    cout << "Hasil pembagian: " << hasilBagi << endl;

    return 0;
}
```

SREENSHOOT CODE



The screenshot displays the Visual Studio Code interface. The Explorer pane on the left shows the project structure with 'strukturdata_2.cpp' selected. The Editor pane shows the source code, which is identical to the one provided in the 'SOURCE CODE' section. The Output pane at the bottom shows the execution results of the program, demonstrating the arithmetic operations performed on the input values 6 and 7.

```
PS D:\praktikum\strukturdata teori> cd "d:\praktikum\strukturdata teori"; if ($?) { g++ strukturdata_2.cpp -o strukturdata_2 ; if ($?) { .\strukturdata_2 }
Masukkan bilangan pertama: 6
Masukkan bilangan kedua: 7
Hasil penjumlahan: 13
Hasil pengurangan: -1
Hasil perkalian: 42
Hasil pembagian: 0
PS D:\praktikum\strukturdata teori>
```

DESKRIPSI

meminta pengguna untuk memasukkan dua bilangan, kemudian melakukan operasi tambah, kurang, kali, dan bagi pada bilangan-bilangan tersebut. Hasil dari setiap operasi kemudian ditampilkan ke layar. Kode ini termasuk library iostream dan menggunakan namespace std. Variabel bilangan1 dan bilangan2 digunakan untuk menyimpan bilangan-bilangan yang dimasukkan, sedangkan variabel hasilTambah, hasilKurang, hasilKali, dan hasilBagi digunakan untuk menyimpan hasil dari setiap operasi.

UNGUIDED II

SOURCE CODE “STRUCT”

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

struct Warga
{
    string name;
    string address;
    int age;
};

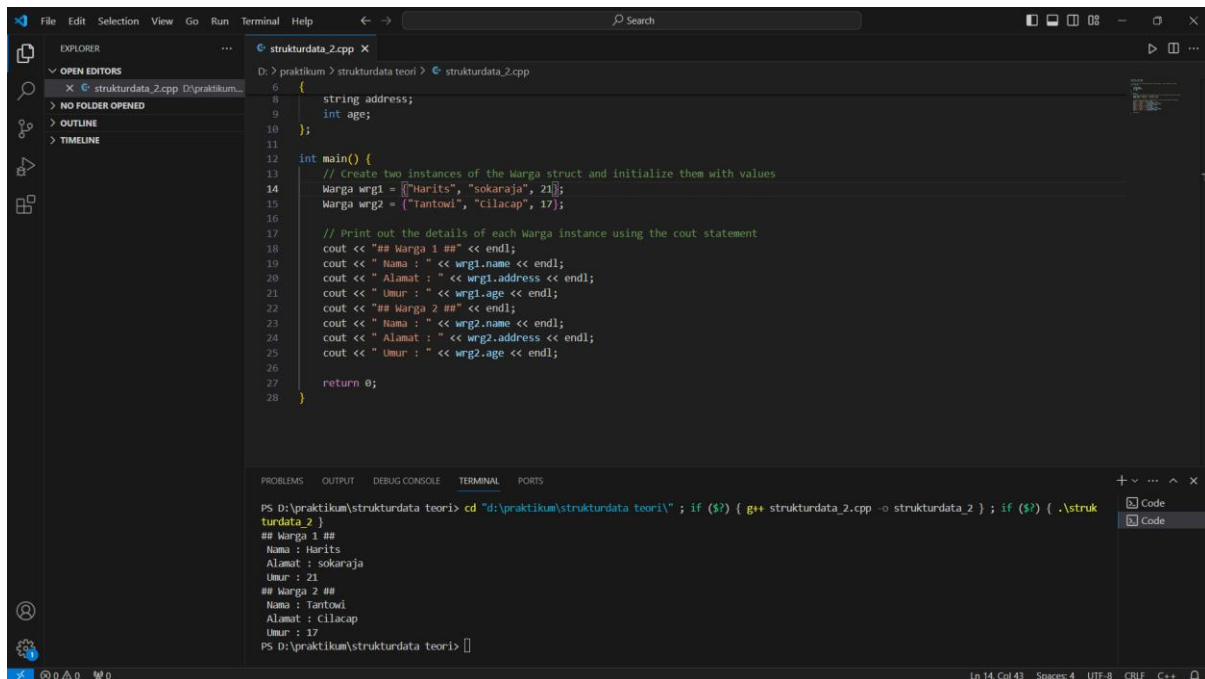
int main() {

    Warga wrng1 = {"Harits", "sokaraja", 21};
    Warga wrng2 = {"Tantowi", "Cilacap", 17};

    cout << "## Warga 1 ##" << endl;
    cout << " Nama : " << wrng1.name << endl;
    cout << " Alamat : " << wrng1.address << endl;
    cout << " Umur : " << wrng1.age << endl;
    cout << "## Warga 2 ##" << endl;
    cout << " Nama : " << wrng2.name << endl;
    cout << " Alamat : " << wrng2.address << endl;
    cout << " Umur : " << wrng2.age << endl;

    return 0;
}
```

SCREENSHOOT PROGRAM



```
6 {
7     string address;
8     int age;
9 };
10
11
12 int main() {
13     // Create two instances of the Warga struct and initialize them with values
14     Warga wrg1 = {"Harits", "sokaraja", 21};
15     Warga wrg2 = {"rantow", "cilacap", 17};
16
17     // Print out the details of each Warga instance using the cout statement
18     cout << "## Warga 1 ##" << endl;
19     cout << "Nama : " << wrg1.name << endl;
20     cout << "Alamat : " << wrg1.address << endl;
21     cout << "Umur : " << wrg1.age << endl;
22     cout << "## Warga 2 ##" << endl;
23     cout << "Nama : " << wrg2.name << endl;
24     cout << "Alamat : " << wrg2.address << endl;
25     cout << "Umur : " << wrg2.age << endl;
26
27     return 0;
28 }
```

```
PS D:\praktikum\strukturdata teori> cd "d:\praktikum\strukturdata teori\" ; if ($?) { g++ strukturdata_2.cpp -o strukturdata_2 } ; if ($?) { .\strukturdata_2 }
## Warga 1 ##
Nama : Harits
Alamat : sokaraja
Umur : 21
## Warga 2 ##
Nama : rantow
Alamat : cilacap
Umur : 17
PS D:\praktikum\strukturdata teori>
```

DESKRIPSI PROGRAM

Program ini adalah sebuah program C++ yang mendefinisikan sebuah struct bernama Warga dengan tiga anggota: name, address, dan age. Di dalam fungsi main(), program membuat dua instance dari struct Warga dan menginisialisasinya dengan nilai. Program kemudian mencetak detail dari setiap instance Warga menggunakan pernyataan cout. Program ini digunakan untuk menunjukkan cara membuat struct dan mengakses anggota-anggota struct menggunakan notasi titik (.).

SOURCE “CLASS”

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

class Warga {
public:
    string nama;
    string alamat;
    int usia;

    // Constructor untuk inisialisasi objek Warga
    Warga(string n, string addr, int a) : nama(n), alamat(addr), usia(a) {}

    // Metode untuk menampilkan informasi warga
    void tampilkanInformasi() {
```

```

        cout << "Nama : " << nama << endl;
        cout << "Alamat : " << alamat << endl;
        cout << "Usia : " << usia << endl;
    }
};

int main() {
    // Membuat objek Warga menggunakan constructor
    Warga wrg1("hartis", "sokaraja", 21);
    Warga wrg2("Tantowi", "Cilacap", 17);

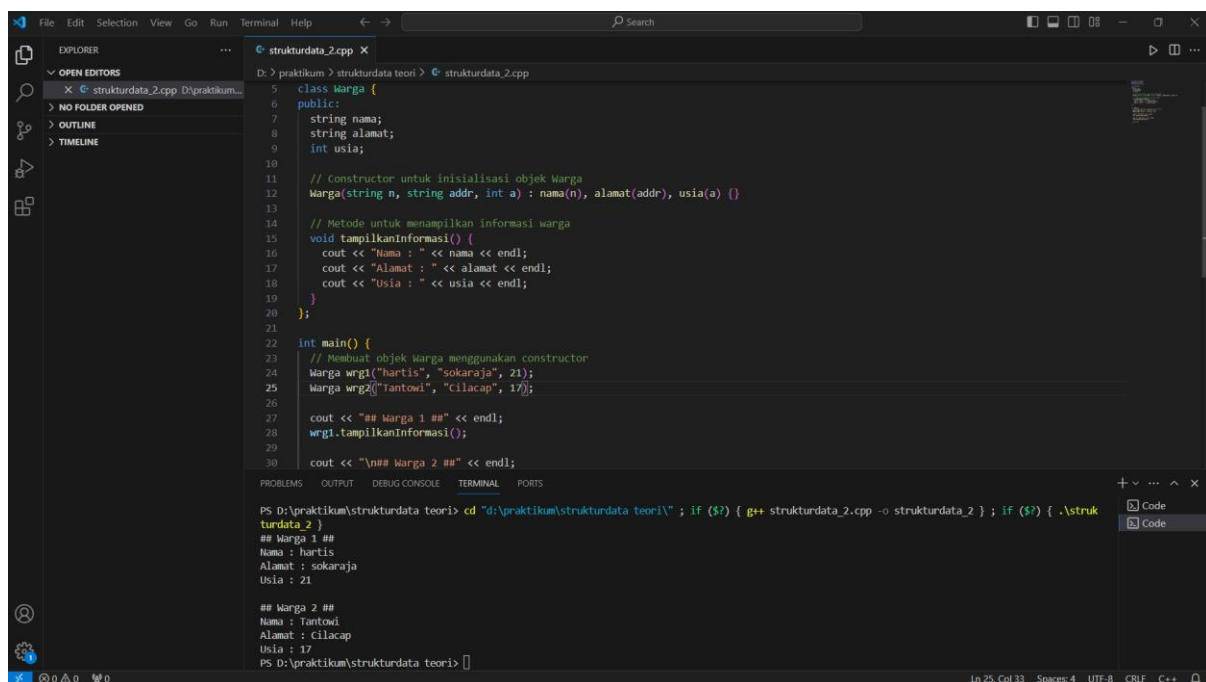
    cout << "## Warga 1 ##" << endl;
    wrg1.tampilkanInformasi();

    cout << "\n## Warga 2 ##" << endl;
    wrg2.tampilkanInformasi();

    return 0;
}

```

SCREENSHOOT CODE



DESKRIPSI

Kode di atas adalah program C++ yang membuat sebuah kelas bernama `Warga` dengan tiga atribut: `nama` (string), `alamat` (string), dan `usia` (integer). Kelas ini memiliki sebuah constructor yang digunakan untuk menginisialisasi objek `Warga` dengan tiga argumen: `n` (string), `addr` (string), dan `a` (integer). Konstruktor ini akan menetapkan nilai dari `nama`,

alamat, dan usia dengan nilai-nilai yang diberikan. Jadi, kode ini akan menampilkan informasi dua orang warga yang telah didefinisikan menggunakan constructor dan method `tampilkanInformasi()`. Hasilnya akan menampilkan:

```
## Warga 1 ##
Nama : hartis
Alamat : sokaraja
Usia : 21

## Warga 2 ##
Nama : Tantowi
Alamat : Cilacap
Usia : 17
```

UNGUIDED III

SOURCE CODE

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

int main() {

    map<string, int> m;

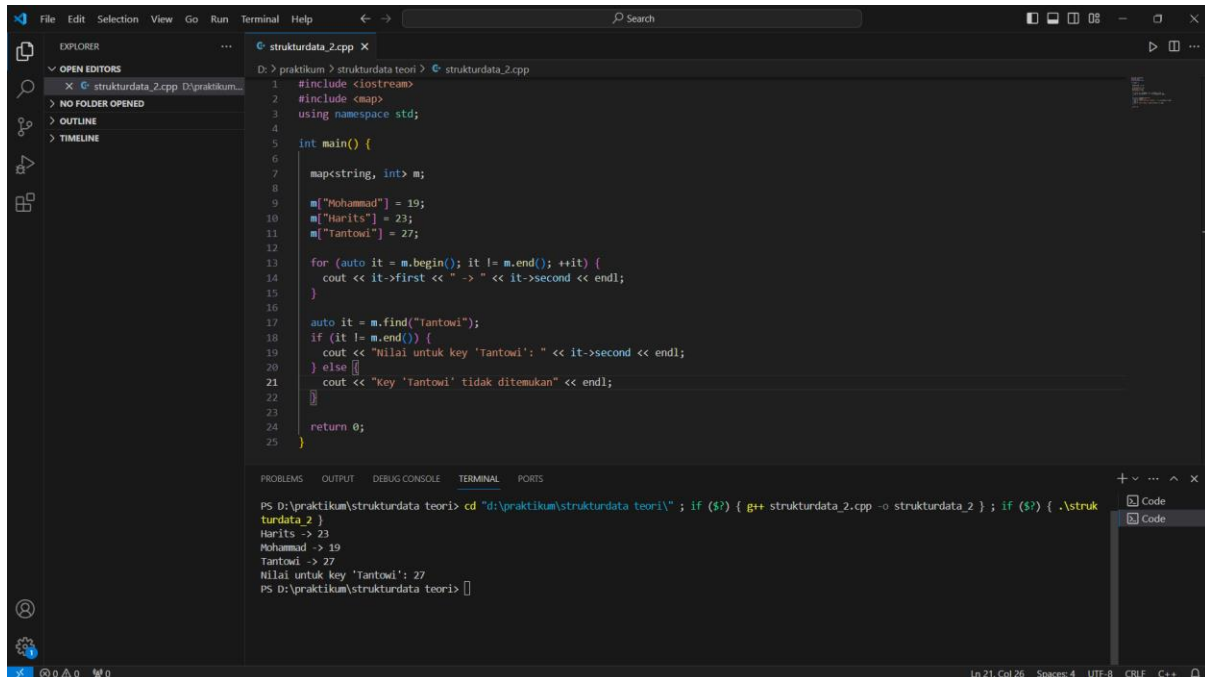
    m["Mohammad"] = 19;
    m["Harits"] = 23;
    m["Tantowi"] = 27;

    for (auto it = m.begin(); it != m.end(); ++it) {
        cout << it->first << " -> " << it->second << endl;
    }

    auto it = m.find("Tantowi");
    if (it != m.end()) {
        cout << "Nilai untuk key 'Tantowi': " << it->second << endl;
    } else {
        cout << "Key 'Tantowi' tidak ditemukan" << endl;
    }

    return 0;
}
```

SCREENSHOOT PROGRAM



The screenshot shows a Visual Studio IDE with a C++ file named `strukturdata_2.cpp`. The code defines a `map` of strings to integers, inserts three pairs, iterates through them, and searches for a specific key. The terminal shows the successful execution of the program.

```
1 #include <iostream>
2 #include <map>
3 using namespace std;
4
5 int main() {
6
7     map<string, int> m;
8
9     m["Muhammad"] = 19;
10    m["Harits"] = 23;
11    m["Tantowi"] = 27;
12
13    for (auto it = m.begin(); it != m.end(); ++it) {
14        cout << it->first << " -> " << it->second << endl;
15    }
16
17    auto it = m.find("Tantowi");
18    if (it != m.end()) {
19        cout << "Nilai untuk key 'Tantowi': " << it->second << endl;
20    } else {
21        cout << "Key 'Tantowi' tidak ditemukan" << endl;
22    }
23
24    return 0;
25 }
```

```
PS D:\praktikum\strukturdata teori> cd "d:\praktikum\strukturdata teori\"; if ($?) { g++ strukturdata_2.cpp -o strukturdata_2 }; if ($?) { .\strukturdata_2 }
Harits -> 23
Muhammad -> 19
Tantowi -> 27
Nilai untuk key 'Tantowi': 27
PS D:\praktikum\strukturdata teori>
```

DESKRIPSI PROGRAM

Program ini menunjukkan penggunaan kontainer map dalam C++ untuk menyimpan pasangan kunci-nilai. Map merupakan struktur data yang ideal untuk situasi di mana kita perlu mengaitkan informasi dengan identifier unik. Dalam contoh ini, map digunakan untuk menyimpan pasangan string (sebagai kunci) dan bilangan bulat (sebagai nilai). Pada awal program, sebuah map kosong bernama `m` dibuat. Kemudian, beberapa pasangan kunci-nilai ditambahkan ke `m` menggunakan operator `[]`. Operator ini memungkinkan kita untuk mengakses elemen map berdasarkan kuncinya. Selanjutnya, program melakukan iterasi melalui seluruh elemen `m` menggunakan iterator `it`. Iterator adalah objek yang memungkinkan kita untuk menavigasi elemen-elemen dalam kontainer. Dalam contoh ini, iterator digunakan untuk mencetak setiap pasangan kunci-nilai dalam map. Kemudian, program melakukan pencarian untuk key "Tantowi" menggunakan metode `find()`. Metode ini mengembalikan iterator yang menunjuk ke elemen dengan key yang diberikan, atau iterator ke akhir map jika key tidak ditemukan. Terakhir, program menunjukkan cara menggunakan nilai yang dikembalikan oleh `find()`. Jika key ditemukan, program mencetak nilai yang terkait. Jika key tidak ditemukan, program menampilkan pesan yang menunjukkan bahwa key tidak ada dalam map. Program ini mendemonstrasikan beberapa operasi dasar yang dapat dilakukan dengan map dalam C++. Map adalah alat yang ampuh untuk menyimpan dan mengelola data, dan program ini memberikan contoh bagaimana map dapat digunakan dalam aplikasi C++ yang sebenarnya.

PERBEDAAN

Perbedaannya yaitu array cocok untuk situasi di mana indeks berurutan dan jumlah elemen diketahui di awal, sementara map lebih fleksibel dan berguna ketika kita perlu mengaitkan nilai dengan kunci yang dapat bervariasi

BAB V

KESIMPULAN

Menguasai berbagai jenis tipe data, seperti primitif (bool, char, int, float, double, void) untuk menyimpan nilai sederhana, abstrak (kelas, struct) untuk menyembunyikan detail implementasi dan menyediakan operasi pada data, dan koleksi (vector, list, map, array) untuk kumpulan data, merupakan fondasi penting dalam membangun program C++ yang efektif dan efisien. Pemilihan tipe data yang tepat akan memaksimalkan performa, kemudahan akses data, serta menjaga struktur program yang rapi dan terorganisir. Hal ini akan membantu programmer dalam menulis kode yang lebih mudah dibaca, dipelihara, dan diuji, sehingga menghasilkan program yang handal dan berkinerja tinggi.

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>