

**LAPORAN PRAKTIKUM STRUKTUR DATA DAN
ALGORITMA
MODUL 1
TIPE DATA**



DISUSUN OLEH:

PRIESTY AMELIANA MAULIDAH

2311102175

S1 IF-11-E

DOSEN:

Muhammad Afrizal Amrustian, S. Kom.

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

INSTITUT TEKNOLOGI TELKOM PURWOKERTO PURWOKERTO

A.DASAR TEORI

Tipe data adalah sebuah pengklasifikasi 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

Tipe data Primitif

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

a.Int: adalah tipe data yang digunakan untuk menyimpan bilangan bulat seperti 12, 1, dan 4, dan sebagainya.

B.Float: Tipe data digunakan untuk menyimpan angka desimal seperti 1.5, 2.1, 3.14, dst.

c.Char: digunakan untuk menyimpan data dalam format karakter. biasanya digunakan untuk simbol seperti A, B, dan C dan sebagainya.

d.Boolean: Tipe data ini digunakan untuk menyimpan nilai Boolean yang hanya memiliki dua nilai: benar dan salah.

Tipe Data Abstrak

Tipe data abstrak, atau disebut tipe data abstrak (ADT), adalah tipe data yang dibuat oleh pemrogram sendiri. Hasil situs gratis ramalan psikologis oleh hewan Kelas Fungsi adalah fungsi program berorientasi objek (OPP) dalam bahasa C++, dan mirip dengan fungsi struktur struktur data dalam bahasa C. Keduanya dirancang untuk menyertakan tipe data sebagai anggota. Menurut learn.microsoft.com, perbedaan antara Struct dan Class adalah akses default, Struct bersifat publik dan Class bersifat pribadi.

Tipe Data Koleksi

Tipe data koleksi adalah tipe data yang digunakan untuk mengelompokkan dan menyimpan beberapa nilai atau objek secara bersamaan.

Tipe data koleksi memungkinkan Anda menyimpan, mengelola, dan mengakses data dalam jumlah besar dengan cara yang terstruktur. Ada beberapa jenis pengambilan data yang biasa digunakan dalam pemrograman:

a. Array: Array adalah struktur data statis yang menyimpan elemen dengan tipe data yang sama. Elemen-elemen ini dapat diakses melalui index. Ukuran array tetap dan pada saat dideklarasikan.

B. Vector : Vector adalah Standard Template Library (STL) jika berbentuk `std::vector` dalam C/C++. Secara umum vektor mirip dengan array dan dapat menyimpan data dalam bentuk elemen, dan alokasi memori otomatis dan bersebelahan. Kekuatan vektor melampaui jumlah elemen dinamis; vektor di C/C++ juga memiliki fitur pelengkap seperti akses elemen, iterator, kapasitas, dan pengubah.

c. map: map mirip dengan array, tetapi dapat menggunakan tipe data non-integer dengan indeks .

Indeks disebut sebagai "kunci" pada peta.

Peta `std::map` menggunakan pohon self-balancing , khususnya pohon merah-hitam.

B.Guided

Guided 1

```
#include <iostream>

using namespace std;

int main(){
    // deklarasi variabel
    char konversi;
    float suhu;

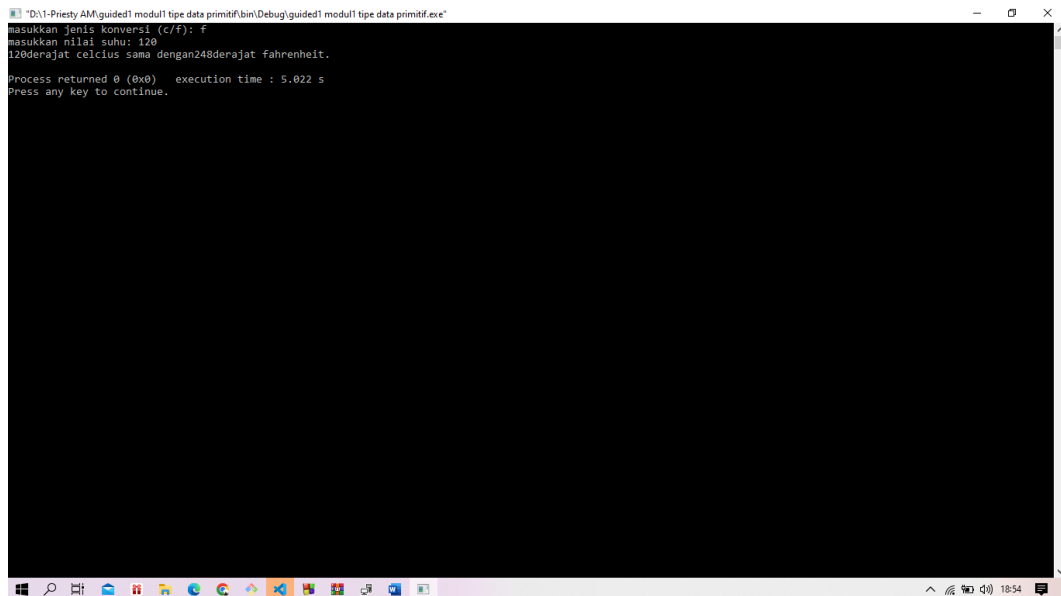
    // input jenis konversi dan nilai suhu
    cout << "masukkan jenis konversi (c/f): ";
    cin >> konversi;

    cout << "masukkan nilai suhu: ";
    cin >> suhu;

    // melakukan konversi suhu
    float hasil_konversi;
    if ( toupper(konversi)=='F' ) {
        hasil_konversi = (suhu * 9.0 / 5.0)+ 32;
        cout << suhu << "derajat celcius sama dengan" << hasil_konversi << "derajat fahrenheit." <<endl;
    }else if (toupper(konversi)=='C') {
        hasil_konversi = (suhu -32)* 5.0 / 9.0;
        cout << suhu << "derajat fahrenheit sama dengan " << hasil_konversi << "derajat celcius." <<endl;
    }
```

```
}else{  
    cout << "jenis konversi tidak valid." <<endl;  
}  
  
return 0;  
}
```

Screenshots output



Deskripsi:

1. Namespace:

- `# include <iostream>` : menyertakan pustaka `iostream` yang diperlukan untuk operasi input dan output.

- Using namespace std :mengimpor semua nama dari namespace std untuk kemudahan penggunaan.

2. Deklarasi variabel

- Char konversi : menyimpan jenis konversi yang dipilih pengguna (c/f).
- Float suhu : menyimpan nilai suhu yang dimasukkan pengguna.
- Float hasil_konversi: menyimpan hasil konversi suhu.

3. Input data

- Cout << "masukkan jenis konversi (c/f): "; : menampilkan pesan untuk meminta pengguna memasukkan jenis konversi.
- Cin >> konversi; : membaca input jenis konversi dari pengguna dan menyimpannya dalam variabel konversi.
- Cout << "masukkan nilai suhu: "; : menampilkan pesan untuk meminta pengguna memasukkan nilai suhu.
- Cin >> suhu ; : membaca input nilai suhu dari pengguna dan menyimpannya dalam variabel suhu.

4. Konversi suhu

- If (toupper(konversi)=='f') { : memeriksa apakah jenis konversi yang dipilih adalah "f"
 - Jika, rumus hasil_konversi = (suhu * 9.0 / 5.0) + 32; digunakan untuk mengonversi celcius ke fahrenheit.
 - Hasil konversi kemudian ditampilkan dengan pesan cout << suhu << "derajat celcius sama dengan " << hasil_konversi << "derajat fahrenheit." <<endl; .
- else if (toupper(konversi)=='F') {:
Memeriksa apakah jenis konversi yang dipilih adalah "C".
 - Jika ya, rumus hasil_konversi = (suhu - 32) * 5.0 / 9.0; digunakan untuk mengonversi Fahrenheit ke Celcius.
 - Hasil konversi kemudian ditampilkan dengan pesan cout << suhu << "derajat fahrenheit sama dengan " << hasil_konversi << "derajat celcius." <<endl;.

- else: Jika jenis konversi tidak valid, pesan `cout << "jenis konversi tidak valid." << endl;` ditampilkan.

5. Mengembalikan Nilai:

- `return 0;` Menandakan bahwa program telah selesai dengan sukses.

Guided 2

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

printf("## Mahasiswa 2 ##\n");

printf("Nama: %s\n", mhs2.name);

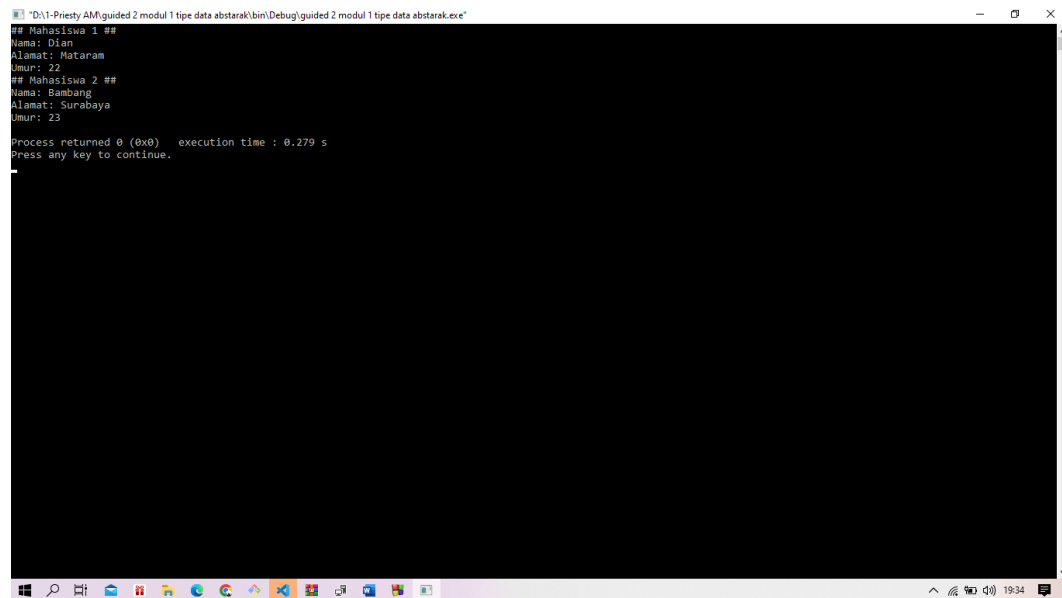
printf("Alamat: %s\n", mhs2.address);

printf("Umur: %d\n", mhs2.age);


return 0;

}
```

Screenshots output



```
"D:\-Priesty AM\guided 2 modul 1 tipe data abstrak\bin\Debug\guided 2 modul 1 tipe data abstrak.exe"
## Mahasiswa 1 ##
Nama: Dian
Alamat: Mataram
Umur: 22
## Mahasiswa 2 ##
Nama: Bambang
Alamat: Surabaya
Umur: 23
Process returned 0 (0x0)   execution time : 0.279 s
Press any key to continue.
```

Deskripsi :

1.Header Inclusion:

- `#include <stdio.h>`: This line includes the standard input/output library, providing functions like `printf` for formatted printing.

2. Struct Definition:

- `struct Mahasiswa { ... }`: This defines a struct named `Mahasiswa` (meaning "Student" in Indonesian).
- It has three member variables:
- `const char *name`: Stores the student's name as a constant character pointer.
- `const char *address`: Stores the student's address as a constant character pointer.
- `int age`: Stores the student's age as an integer.
- `const char *`: Using constant character pointers (`const char *`) helps prevent accidental modification of the data within the struct after it's assigned.

3. Main Function:

- `int main()`: This is the program's entry point.

4. Declaring Struct Variables:

- `struct Mahasiswa mhs1, mhs2;`: Two `Mahasiswa` structs are declared, named `mhs1` and `mhs2`.

5. Assigning Values:

6. The code assigns values to the member variables of mhs1 and mhs2 using the dot (.) operator:

- `mhs1.name = "Dian";`
- `mhs1.address = "Mataram";`
- `mhs1.age = 22;`
- Similarly for mhs2.

5. Printing Information:

○ `printf` statements are used to print formatted output to the console:

- `printf("## Mahasiswa 1 ##\n");` prints a header for student 1.
- `printf("Nama: %s\n", mhs1.name);` prints the name using the format specifier `%s` for strings.
- There's a typo here: `printf("Alamat: %s\n", mhs1.name);` should be `printf("Alamat: %s\n", mhs1.address);` to print the address.
- The rest of the `printf` statements follow the same pattern for student 2.

7. Returning from main:

- return 0;: This indicates successful program termination.

Guided 3

```
#include <iostream>

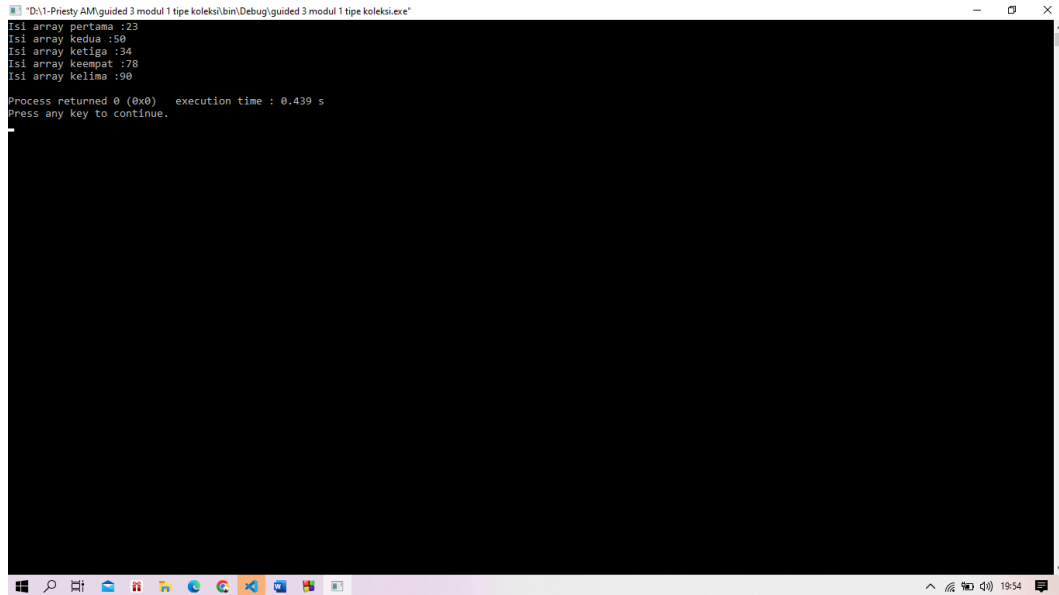
using namespace std;

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;
}
```

Screenshots output



```
"D:\1-Priesty AM\guided 3 modul 1 tipe koleksi\bin\Debug\guided 3 modul 1 tipe koleksi.exe"
isi array pertama :23
isi array kedua :50
isi array ketiga :34
isi array keempat :78
isi array kelima :90
Process returned 0 (0x0) execution time : 0.439 s
Press any key to continue.
```

Deksripsi :

1. Pustaka dan Namespace:

- `#include <iostream>`: Menyertakan pustaka `iostream` yang diperlukan untuk operasi input dan output.
- `using namespace std`: Mengimpor semua nama dari namespace `std` untuk kemudahan penggunaan.

2. Deklarasi dan Inisialisasi Array:

- `int nilai[5]`: Mendeklarasikan array `nilai` dengan 5 elemen bertipe `int`.

- `nilai[0] = 23`: Menginisialisasi elemen pertama array nilai dengan nilai 23.
- Inisialisasi serupa dilakukan untuk elemen array lainnya dengan nilai berbeda.

3. Mencetak Array:

- `cout << "Isi array pertama :" << nilai[0] << endl;`: Mencetak nilai elemen pertama array nilai dengan pesan "Isi array pertama :".
- Perintah `cout` serupa digunakan untuk mencetak nilai elemen array lainnya.

4. Mengembalikan Nilai:

- `return 0;`: Menandakan bahwa program telah selesai dengan sukses.

c. unguided/tugas

unguided 1

```
#include <iostream>

using namespace std;

int main() {
    // Deklarasi variabel
    float jari_jari, luas, keliling;

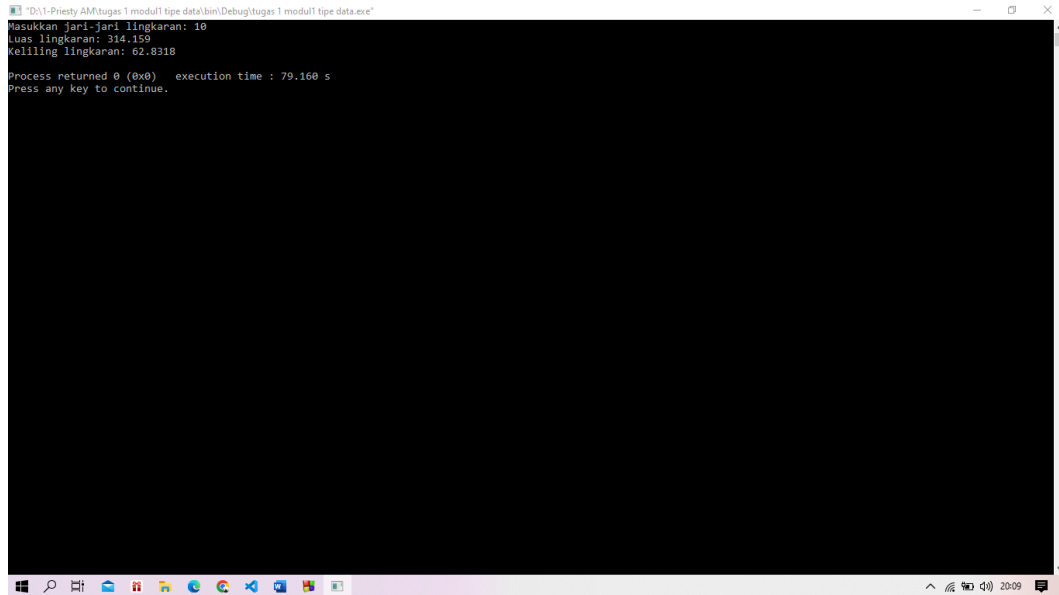
    // Input jari-jari
    cout << "Masukkan jari-jari lingkaran: ";
    cin >> jari_jari;

    // Menghitung luas dan keliling
    luas = 3.14159 * jari_jari * jari_jari;
    keliling = 2 * 3.14159 * jari_jari;

    // Menampilkan hasil
    cout << "Luas lingkaran: " << luas << endl;
    cout << "Keliling lingkaran: " << keliling << endl;

    return 0;
}
```


Screenshots output

A screenshot of a Windows command prompt window. The title bar reads "D:\1-Priesty AM\tugas 1 modul1 tipe data\bin\Debug\tugas 1 modul1 tipe data.exe". The command prompt shows the following text: "Masukkan jari_jari lingkaran: 10", "Luas lingkaran: 314.159", and "Keliling lingkaran: 62.8318". Below this, it says "Process returned 0 (0x0) execution time : 79.160 s" and "Press any key to continue.". The Windows taskbar is visible at the bottom with various icons and a system clock showing 20:09.

```
"D:\1-Priesty AM\tugas 1 modul1 tipe data\bin\Debug\tugas 1 modul1 tipe data.exe"
Masukkan jari_jari lingkaran: 10
Luas lingkaran: 314.159
Keliling lingkaran: 62.8318

Process returned 0 (0x0) execution time : 79.160 s
Press any key to continue.
```

Deskripsi :

1. Pustaka dan Namespace:

- `#include <iostream>`: Menyertakan pustaka `iostream` yang diperlukan untuk operasi input dan output.
- `using namespace std`: Mengimpor semua nama dari namespace `std` untuk kemudahan penggunaan.

2. Deklarasi Variabel:

- `float jari_jari`: Menyimpan nilai jari-jari lingkaran.

- float luas: Menyimpan hasil perhitungan luas lingkaran.
- float keliling: Menyimpan hasil perhitungan keliling lingkaran.

3. `cout << "Masukkan jari-jari lingkaran: ";`
Menampilkan pesan untuk meminta Input Jari-jari:

- pengguna memasukkan nilai jari-jari.
- `cin >> jari_jari;` Membaca input nilai jari-jari dari pengguna dan menyimpannya dalam variabel `jari_jari`.

4. Menghitung Luas dan Keliling:

- `luas = 3.14159 * jari_jari * jari_jari;`
Menghitung luas lingkaran dengan rumus πr^2 .
- `keliling = 2 * 3.14159 * jari_jari;`
Menghitung keliling lingkaran dengan rumus $2\pi r$.

5. Menampilkan Hasil:

- `cout << "Luas lingkaran: " << luas << endl;`
Menampilkan nilai luas lingkaran yang telah dihitung.

- `cout << "Keliling lingkaran: " << keliling << endl;`; Menampilkan nilai keliling lingkaran yang telah dihitung.

6. Mengembalikan Nilai:

- `return 0;`; Menandakan bahwa program telah selesai dengan sukses.

Unguided 2

- Class dan struct merupakan dua cara untuk mendefinisikan tipe data terstruktur dalam C++. Tipe data terstruktur memungkinkan Anda mengelompokkan data terkait dan fungsi yang bekerja pada data tersebut.

Fungsi Class dan Struct:

1. Mendefinisikan Struktur Data:

- Class dan struct memungkinkan Anda untuk membuat struktur data kustom yang terdiri dari atribut (variabel) dan metode (fungsi).
- Atribut dapat berupa tipe data dasar (int, float, dll.) atau tipe data terstruktur lainnya.
- Metode adalah fungsi yang bekerja pada data dalam struktur.

2. Enkapsulasi:

- Class dan struct memungkinkan Anda untuk menyembunyikan detail implementasi data dan metode.
- Hanya bagian public dari class/struct yang dapat diakses dari luar.
- Enkapsulasi membantu meningkatkan modularitas dan keamanan kode.

3. Pewarisan:

- Class dapat mewarisi dari class lain, memungkinkan reuse kode dan fungsionalitas.
- Class turunan dapat menambahkan atribut dan metode baru ke class dasar.
- Pewarisan membantu meningkatkan fleksibilitas dan skalabilitas kode.

4. Polimorfisme:

- Class dan struct memungkinkan Anda untuk mendefinisikan fungsi dengan nama yang sama tetapi dengan parameter berbeda.
- Fungsi mana yang dipanggil tergantung pada jenis objek yang digunakan.
- Polimorfisme membantu meningkatkan fleksibilitas dan reusability kode.

Code program c++ (struct)

```
#include <iostream>

using namespace std;

// Struct untuk mendefinisikan data Mahasiswa
struct Mahasiswa {
    string nama;
    int nim;
    float ipk;

    // Menampilkan informasi mahasiswa
    void tampilkanInfo() {
        cout << "Nama: " << nama << endl;
        cout << "NIM: " << nim << endl;
        cout << "IPK: " << ipk << endl;
    }
};

int main() {
    // Deklarasi objek Mahasiswa
    Mahasiswa mhs1;
```

```
// Mengisi data untuk mhs1

mhs1.nama = "dian ";

mhs1.nim = "123456789";

mhs1.ipk = "3.9";

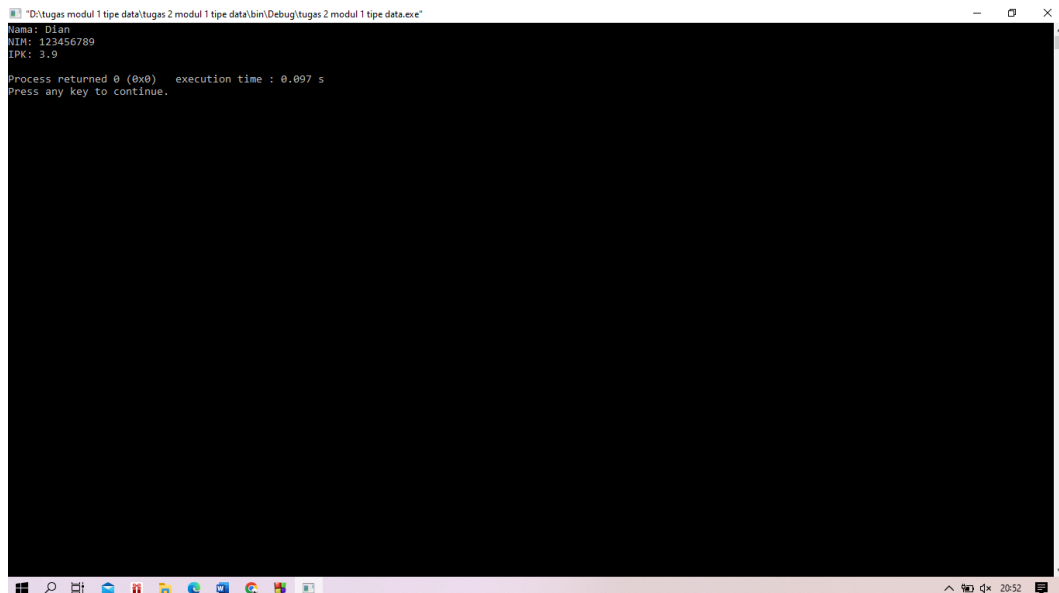

// Menampilkan informasi mhs1

mhs1.tampilkanInfo();


return 0;

}
```

Screenshots output



```
D:\tugas modul 1 tipe data\tugas 2 modul 1 tipe data\bin\Debug\tugas 2 modul 1 tipe data.exe
Nama: Dian
NIM: 123456789
IPK: 3.9

Process returned 0 (0x0)   execution time : 0.097 s
Press any key to continue.
```

Deskripsi:

1. Pustaka dan Namespace:

- `#include <iostream>`: Menyertakan pustaka `iostream` untuk operasi input dan output.
- `using namespace std`: Mengimpor semua nama dari namespace `std` untuk kemudahan penggunaan.

2. Definisi Struct Mahasiswa:

- `struct Mahasiswa`: Mendefinisikan struct bernama `Mahasiswa`.
- Struct ini memiliki tiga **atribut**:
 - `nama`: Menyimpan nama mahasiswa sebagai string.
 - `nim`: Menyimpan NIM mahasiswa sebagai integer.
 - `ipk`: Menyimpan IPK mahasiswa sebagai float.
- Struct ini memiliki satu **metode**:
 - `tampilkanInfo()`: Menampilkan informasi mahasiswa (`nama`, `NIM`, dan `IPK`) ke konsol.

3. Penggunaan Struct Mahasiswa:

- `Mahasiswa mhs1`: Mendeklarasikan objek `mhs1` dari struct `Mahasiswa`.

- `mhs1.nama = "Dian";` Mengisi nilai atribut nama untuk `mhs1` dengan "Dian".
- `mhs1.nim = 123456789;` Mengisi nilai atribut nim untuk `mhs1` dengan 123456789.
- `mhs1.ipk = 3.9;` Mengisi nilai atribut ipk untuk `mhs1` dengan 3.9.
- `mhs1.tampilkanInfo();` Memanggil metode `tampilkanInfo()` pada objek `mhs1` untuk menampilkan informasinya.

4. Output Program:

Nama: Dian

NIM: 123456789

IPK: 3.9.

Code c++ (class)

```
#include <iostream>

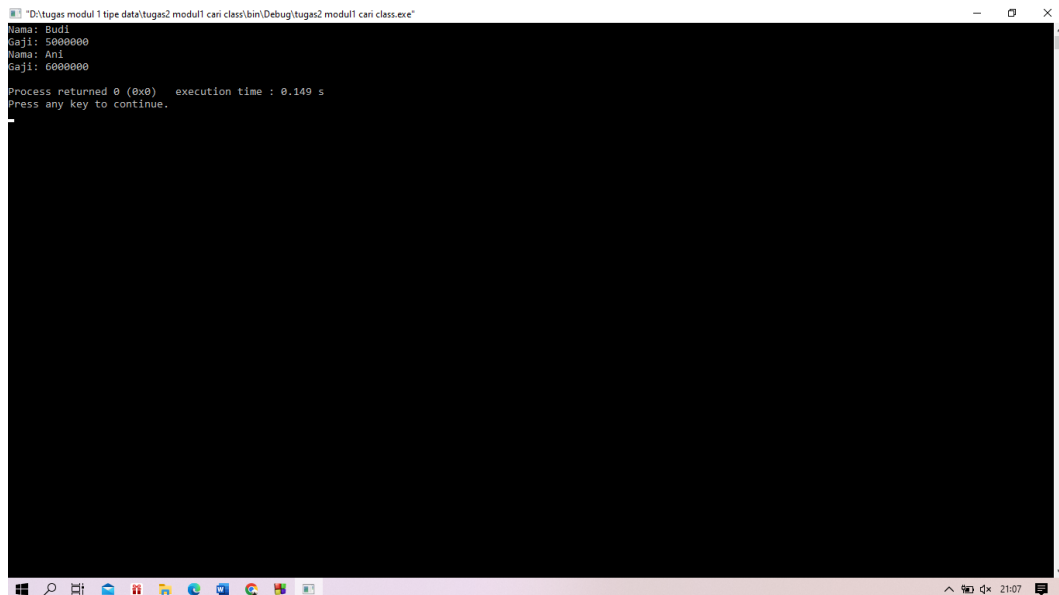
class Karyawan {
private:
    std::string nama;
    int gaji;

public:
    Karyawan(std::string nama, int gaji) {
        this->nama = nama;
        this->gaji = gaji;
    }
}
```



```
void tampilkan_info() {  
    std::cout << "Nama: " << nama << std::endl;  
    std::cout << "Gaji: " << gaji << std::endl;  
}  
};  
  
int main() {  
    Karyawan karyawan1("Budi", 5000000);  
    Karyawan karyawan2("Ani", 6000000);  
  
    karyawan1.tampilkan_info();  
    karyawan2.tampilkan_info();  
  
    return 0;  
}
```

Screenshots output



```
"D:\tugas modul 1 tipe data\tugas2 modul1 cari class\bin\Debug\tugas2 modul1 cari class.exe"
Nama: Budi
Gaji: 5000000
Nama: Ani
Gaji: 6000000
Process returned 0 (0x0) execution time : 0.149 s
Press any key to continue.
```

Deskripsi :

1. Pustaka dan Namespace:

- `#include <iostream>`: Menyertakan pustaka `iostream` untuk operasi input dan output.

2. Definisi Class Karyawan:

- `class Karyawan`: Mendefinisikan class bernama `Karyawan`.
- Class ini memiliki dua **atribut**:
 - `nama`: Menyimpan nama karyawan sebagai string.
 - `gaji`: Menyimpan gaji karyawan sebagai integer.
- Class ini memiliki satu **metode**:

- `tampilkan_info()`: Menampilkan informasi karyawan (nama dan gaji) ke konsol.

3. Penggunaan Class Karyawan:

- `Karyawan karyawan1("Budi", 5000000);`
Membuat objek `karyawan1` dari class `Karyawan` dengan nama "Budi" dan gaji 5000000.
- `Karyawan karyawan2("Ani", 6000000);`
Membuat objek `karyawan2` dari class `Karyawan` dengan nama "Ani" dan gaji 6000000.
- `karyawan1.tampilkan_info();`: Memanggil metode `tampilkan_info()` pada objek `karyawan1` untuk menampilkan informasinya.
- `karyawan2.tampilkan_info();`: Memanggil metode `tampilkan_info()` pada objek `karyawan2` untuk menampilkan informasinya.

4. Output Program:

Nama: Budi

Gaji: 5000000

Nama: Ani

Gaji: 6000000

Unguided 3

```
#include <iostream>

#include <map>

using namespace std;

int main() {
    // Deklarasi map untuk menyimpan data mahasiswa
    map<string, int> mhs;

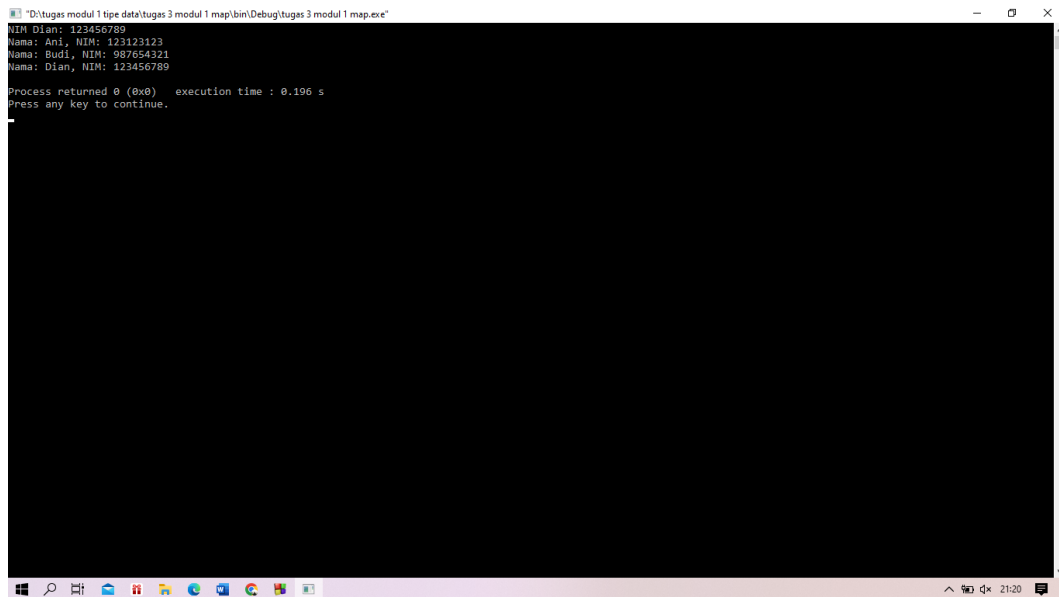
    // Menambahkan data ke map
    mhs["Dian"] = 123456789;
    mhs["Budi"] = 987654321;
    mhs["Ani"] = 123123123;

    // Mencari data di map
    if (mhs.find("Dian") != mhs.end()) {
        cout << "NIM Dian: " << mhs["Dian"] << endl;
    } else {
        cout << "Dian tidak ditemukan" << endl;
    }

    // Menampilkan semua data di map
    for (auto it = mhs.begin(); it != mhs.end(); ++it) {
        cout << "Nama: " << it->first << ", NIM: " << it->second << endl;
    }

    return 0;
}
```

Screenshots ouput

A screenshot of a Windows command prompt window. The title bar reads "D:\tugas modul 1 tipe data\tugas 3 modul 1 map\bin\Debug\tugas 3 modul 1 map.exe". The output text is as follows:

```
NIM: Dian: 123456789  
Nama: Ani, NIM: 123123123  
Nama: Budi, NIM: 987654321  
Nama: Dian, NIM: 123456789  
Process returned 0 (0x0)   execution time : 0.196 s  
Press any key to continue.
```

The rest of the window is black. The Windows taskbar is visible at the bottom.

Deskripsi :

1. Pustaka dan Namespace:

- `#include <iostream>`: Menyertakan pustaka `iostream` untuk operasi input dan output.
- `#include <map>`: Menyertakan pustaka `map` untuk menggunakan fungsi `map`.
- `using namespace std`: Mengimpor semua nama dari namespace `std` untuk kemudahan penggunaan.

2. Deklarasi Map:

- `map<string, int> mhs`: Mendeklarasikan map `mhs` yang memetakan string (nama) ke integer (NIM).

3. Menambahkan Data ke Map:

- `mhs["Dian"] = 123456789;` Menambahkan data dengan key "Dian" dan value 123456789 ke map mhs.

4. Mencari Data di Map:

- `mhs.find("Dian") != mhs.end();` Mengecek apakah key "Dian" ada di map mhs.
 - Jika ada, kode di dalam blok if akan dijalankan.

5. Menampilkan Semua Data di Map:

- `for (auto it = mhs.begin(); it != mhs.end(); ++it):` Perulangan untuk iterating semua elemen di map mhs.
 - `it->first`: Mengakses key dari elemen map.
 - `it->second`: Mengakses value dari elemen map.

D. kesimpulan

Tipe data adalah elemen fundamental dalam pemrograman yang digunakan untuk mendefinisikan jenis data yang dapat disimpan dan dimanipulasi oleh program. Berikut kesimpulan mengenai tipe data:

1. Pengertian:

- Tipe data adalah kategori yang menentukan jenis informasi yang dapat disimpan dalam variabel.
- Setiap tipe data memiliki karakteristik dan aturannya sendiri.

2. Jenis Tipe Data:

- Tipe data dasar:
 - Bilangan bulat (integer): Menyimpan nilai numerik tanpa desimal.
 - Bilangan desimal (floating-point): Menyimpan nilai numerik dengan desimal.
 - Karakter (character): Menyimpan karakter tunggal.
 - Boolean: Menyimpan nilai true atau false.
- Tipe data terstruktur:
 - Array: Menyimpan kumpulan data dengan tipe data sama.

- Struct: Menyimpan kumpulan data dengan tipe data berbeda.
- Class: Menyimpan kumpulan data dan fungsi terkait.

3. Kegunaan Tipe Data:

- Memastikan program dapat memproses data dengan benar.
- Meningkatkan efisiensi dan keamanan program.
- Mempermudah pembacaan dan pemahaman kode.

4. Tips Memilih Tipe Data:

- Pilih tipe data yang sesuai dengan jenis data yang ingin disimpan.
- Pertimbangkan kebutuhan operasi yang akan dilakukan pada data.
- Gunakan tipe data yang efisien untuk menghemat memori.

E. Referensi

<https://osf.io/preprints/osf/esz5m>

<https://osf.io/preprints/osf/jvx3y>