



Struct dan Tipe Buatan Lain part 2

TIM DOSEN

Your best quote that reflects your approach... “It’s one small step for man, one giant leap for mankind.”

- NEIL ARMSTRONG

Outline

➤ Struktur (Struct)

➤ Union

➤ Bit-field

➤ Enum

➤ typedef

Struktur (Struct)

PART 2

Pointers to Structures

- Pointer selain dapat digunakan ke berbagai tipe data seperti pointer ke int, char dan tipe data lainnya, bisa juga memiliki pointer yang menunjuk ke struktur. Pointer ini disebut pointer struktur.
- Pengaksesan variable struct bertipe pointer reference menggunakan tanda (→)
- Pengaksesan variable struct bertipe pointer reference **Tanpa penggunaan dot (.)**

Pointers to Structures - lanjut

➤ Bentuk umum :

```
struct structure_name
{
    data-type member-1;
    data-type member-1;
    data-type member-1;
    data-type member-1;
};
int main()
{
    struct structure_name *ptr;
}
```

Contoh

Pointer to Struct

```
#include <iostream>

using namespace std;

struct mahasiswa
{
    char nama[50];
    int stambuk;
};

int main() {
    struct mahasiswa stud = {"Ali Tahir", 1};
    struct mahasiswa *ptr;
    ptr = &stud; // penugasan dari stud ke ptr

    std::cout << stud.nama << stud.stambuk ;
    std::cout << "\n" << ptr->nama << ptr->stambuk ;
    return 0;
}
```

Struktur dan Fungsi

Ada dua metode dalam melewati struktur ke fungsi.

- Melewati Nilai (*Passing by Value*)
- Melewati dengan Referensi (*Passing by Reference*)

Passing by Value

❑ Pada passing by value berarti melewatkan variabel struktur sebagai argumen ke suatu fungsi

❑ Contoh :

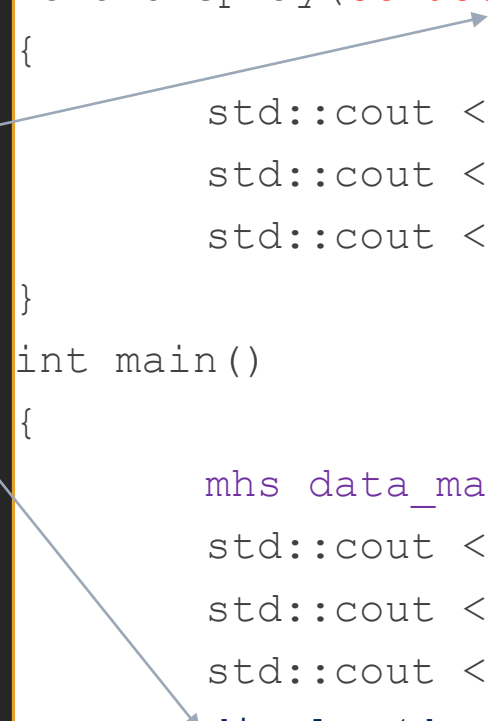
```
struct mhs { //deklarasi tipe struct
    char nama[30];
    int stambuk;
    float nilai;
};
void display(struct mhs st)
{
    std::cout << "Stambuk : " << st.stambuk ;
    std::cout << "\nNama   :" << st.nama ;
    std::cout << "\nNilai  :" << st.nilai ;
}
```

← passing by value
meneruskan variabel
lain ke suatu fungsi,
melewatkan variabel
struktur 'ke' fungsi
display.

Contoh Program

Passing by Value

```
#include <iostream>
//deklarasi struct
struct mhs { //deklarasi tipe struct
    char nama[30];
    int stambuk;
    float nilai;
} ;
void display(struct mhs st)
{
    std::cout << "Stambuk : " << st.stambuk ;
    std::cout << "\nNama      : " << st.nama ;
    std::cout << "\nNilai     : " << st.nilai ;
}
int main()
{
    mhs data_mahasiswa;
    std::cout <<"Stambuk : "; std::cin>>data_mahasiswa.stambuk;
    std::cout <<"Nama   : "; std::cin>>data_mahasiswa.nama;
    std::cout <<"Nilai  : ";std::cin>>data_mahasiswa.nilai;
    display(data_mahasiswa);
    return 0;
}
```



Passing by Reference

- Penggunaan pointer reference (*) dan pointer dereference (&)
- Dalam melewati dengan referensi, alamat variabel struktur menggunakan operator reference dan dilewatkan ke fungsi.
- Jika mengubah variabel struktur yang ada di dalam fungsi, variabel struktur asli yang digunakan untuk memanggil fungsi berubah.
- Untuk mengakses variable bertipe struct yang berpointer reference menggunakan tanda “ → “ diikuti variable struct
- Untuk mengakses nilai dari struktur berpointer reference menggunakan operator deference.

Passing by Reference - lanjut

➤ Contoh

```
struct mhs { //deklarasi tipe struct
    char nama[30];
    int stambuk;
    float nilai;

};
void display(struct mhs *st)
{
    std::cout << "Stambuk : " << st->stambuk ;
    std::cout << "\nNama    : " << st->nama ;
    std::cout << "\nNilai   : " << st->nilai ;
}
```

Contoh Program

Passing by Reference

Ketika mendeklarasikan fungsinya, melewati pointer copy 'st' dari variabel struktur 'data_mahasiswa' di parameternya. Karena pointer adalah variabel tipe struktur bernama mhs, 'struct mhs' sebelum nama pointer dalam argumen fungsi. Dalam fungsi tersebut, mengakses anggota pointer menggunakan tanda ->.

```
#include <iostream>
//deklarasi struct
struct mhs { //deklarasi tipe struct
    char nama[30];
    int stambuk;
    float nilai;
} ;
void display(struct mhs *st)
{
    std::cout << "Stambuk : " << st->stambuk ;
    std::cout << "\nNama      : " << st->nama ;
    std::cout << "\nNilai     : " << st->nilai ;
}
int main()
{
    mhs data_mahasiswa;
    std::cout <<"Stambuk : "; std::cin>>data_mahasiswa.stambuk;
    std::cout <<"Nama   : "; std::cin>>data_mahasiswa.nama;
    std::cout <<"Nilai  : ";std::cin>>data_mahasiswa.nilai;
    display(&data_mahasiswa);
    return 0;
}
```

Array dan struktur (*Array of Struct*)

- ✓ Variable array menggunakan tipe data struktur.
- ✓ Misal Jika perlu menyimpan data 100 mahasiswa tersebut, maka harus mendeklarasikan 100 variabel yang terpisah dari struktur jelas tidak efisien. Untuk itu, diperlukan membuat array dengan tipe struktur.
- ✓ Bentuk umum : `struct nama_struct variable_struct [ukuran];`
- ✓ Contoh : `struct mahasiswa mhs[100];`
- ✓ Pengaksesan variable array dengan tipe struct dengan menggunakan `indeks array` dan menggunakan tanda `dot (.)` kemudian diikuti `field dari struct`.
- ✓ Misal indeks (i) dan variable array `mhs[100]` dan field (stambuk), untuk mengisinya dengan perintah : `mhs[i].stambuk`

Contoh

```
#include <iostream>
#include <string.h>
using namespace std;
struct mahasiswa
{   char nama[50];   int stambuk;   } ;

int main() {
    struct mahasiswa mhs[2];   int i;
    for(i=0; i<2; i++) {           //input data
        cout << "Mahasiswa " << i + 1 ;
        cout << "\nMasukkan stambuk : " ;
        cin >> mhs[i].stambuk;
        cout << "Masukkan nama : ";   cin >> mhs[i].nama;
    } cout << endl;
    for(i=0; i<2; i++) {           //tampil data
        cout << "\nData Mahasiswa ke-" << i + 1;
        cout << "\nStambuk : " << mhs[i].stambuk ;
        cout << "\nNama      : " << mhs[i].nama<<endl ;
    }
    return 0;
}
```

Union

Union



Dideklarasikan dan digunakan dengan cara yang sama dengan struktur. Seperti struktur, ini juga digunakan untuk menyimpan berbagai jenis data.



Anggotanya menggunakan secara bersama ruang penyimpanan memori yang sama, berbeda dengan struktur yang masing-masing variabel menempati lokasi memori yang berbeda.



Jumlah bytes yang digunakan untuk menyimpan union cukup untuk menyimpan data terbesar yang ditangani



Tipe union umumnya digunakan untuk menangani satu, dua, atau tiga variabel dengan tipe yang mirip

Kegunaan

- Menghemat pemakaian sistem memory.
- Berguna untuk low level programming.

Struktur Umum Union

```
union union_name  
{  
  data-type member-1;  
  data-type member-2;  
  data-type member-3;  
  data-type member-4;  
};
```

Contoh

```
union mahasiswa  
{  
  int stambuk;  
  char nama[30];  
};
```

Deklarasi Variable Union

- Deklarasi variable bertipe Union dan akses variable sama dengan struktur.
- Akses variable menggunakan tanda dot (.)
- Contoh :

```
union student  
{  
    int kelas;  
    int stambuk;;  
}s1, s2, s3;
```

Contoh

```
#include <iostream>

using namespace std;

union mahasiswa
{
    int kelas;
    int stambuk;
};

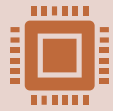
int main() {

    union mahasiswa p1;
    p1.stambuk = 1;
    p1.kelas = 1 ;
    cout<<"Stambuk : "<<p1.stambuk<< "\nKelas : "
<<p1.kelas ;

    return 0;
}
```

Bit-field

Bit-field



Satu bit atau beberapa bit dalam sebuah data berukuran suatu byte atau dua byte dapat diakses dengan mudah melalui bit-field.



Dengan cara ini suatu bit atau beberapa bit dapat diakses tanpa melibatkan operator manipulasi bit (seperti `&` dan `||`).



Selain itu satu atau dua byte dapat dipakai untuk menyimpan sejumlah informasi.

Contoh

```
#include <iostream>
using namespace std;
//deklarasi bit field
struct bit_data
{
    unsigned b0 : 1; unsigned b1 : 1;
    // three-bit unsigned field,
    // allowed values are 0...7
};
int main(){
    //deklarasi union
    union ubyte
    {
        unsigned char byte; bit_data bit;
    };
    ubyte ascii;
    int nilai;
    cout<<"Contoh Penggunaan Bit-field"<<endl;
    cout<<"-----"<<endl;
    cout<<"Masukkan ascii antara 0 s/d 255 : ";
    cin>>nilai;
    ascii.byte = nilai;
    cout<<"\1 0 << Posisi bit"<<endl;
    cout<<ascii.bit.b1<<" "<<ascii.bit.b0<<" "<<endl;
    return 0;
}
```


Enum

Enum



Pada C++, **enum** biasa digunakan saat suatu data yang sudah kita ketahui jumlahnya, dan tidak banyak. Seperti halnya untuk menyatakan nama hari, nama bulan, jenis kelamin, dan lainnya.



Enumerasi adalah suatu set Konstanta Integer yang masing-masing konstanta akan memiliki nama dan nilai yang berbeda.



Enumerasi lebih ditujukan untuk penanganan kesalahan proses input, output maupun proses pengolahan data dalam internal CPU.



Dalam pemrograman C atau C++ tiap-tiap jenis kesalahan akan diidentifikasi oleh nama konstanta hal ini dikarenakan lebih mudah mengingat nama konstanta dibanding nilai atau angka.

Struktur Umum Enum

Bentuk Dasar :

```
enum nama_tipe {  
    nilai_1, nilai_2, ...  
}; // nilai yang diinginkan //contoh dari enum
```

CONTOH :

```
enum warna { COKELAT, MERAH, JINGGA, KUNING, HIJAU, BIRU, HITAM};
```

```
enum pilihan {SATU=1, DUA, TIGA, EMPAT, LIMA};
```

Contoh

```
#include <iostream>
using namespace std;
//pendeklarasian tipe enum berupa 'bulan'
enum bulan{Januari, Februari, Maret, April, Mei, Juni,
           Juli, Agustus, September, Oktober, November, Desember
};
int main() {
    /*pendefinisian variabel bernama b1 dan b2 yang bertipe enum 'bulan'*/
    bulan b1,b2;

    //pemberian nilai pada variabel enum
    b1 = Juli;
    b2 = Juni;

    cout<<"Tipe ENUM C++"<<endl;
    cout<<"-----"<<endl;
    cout<<"nilai b1 = "<< b1<<endl; // 11
    cout<<"nilai b2 = "<< b2<<endl; // 1
    cout<<"\nSelisih kedua bulan di atas adalah "<<b1-b2<<endl;
    return 0;
}
```

typedef

typedef



Typedef adalah perintah atau *keyword* bahasa C atau c++ yang dipakai untuk memberikan nama lain atau **alias** dari tipe data.



Cara Penggunaan Typedef

Perintah **typedef** ditulis sebelum kode program utama, yakni sebelum `main()`.

Struktur Kode

Bentuk Umum :

```
typedef tipeDataAsal namaAlias
```

Contoh Penggunaan Typedef

```
#include <iostream>

typedef unsigned int bulatPositif;
typedef float angkaPecahan;

int main()
{
    bulatPositif a;
    angkaPecahan b;

    a = 123456;
    std::cout<<"Isi variabel a:"<<a<<"\n";

    b = 23.4513;

    std::cout<<"Isi variabel b:"<<b<<"\n";

    return 0;
}
```


Contoh Penggunaan Typedef pada struct

```
#include <iostream>
using namespace std;
struct koordinat
{
    int x;
    int y;
};
typedef struct koordinat titik;
int main ()
{
    titik posisi1 = { 35, 256};

    cout<<"Nilai isi variable struct \n\n";
    cout<<"\t\t x ==>" << posisi1.x;
    cout <<"\t\t y ==>"<< posisi1.y;

    return 0;
}
```