

MODUL 11 RECORD

TUJUAN

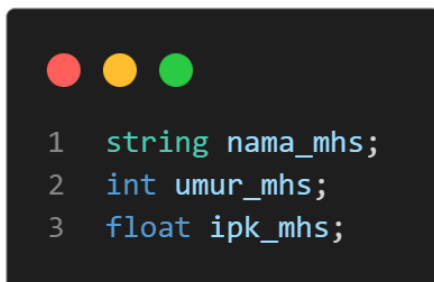
1. Praktikan dapat mengerti dan memahami penggunaan record/struct dalam bahasa pemrograman C.
2. Praktikan dapat mengerti kegunaan record/struct ukan dalam efisiensi pengkodean.
3. Praktikan dapat mengerti cara mengakses dan memodifikasi record/struct.
4. Praktikan dapat memahami relasi antara satu record/struct dan record/struct lain.
5. Praktikan dapat mengerti perbedaan pembuatan suatu record/struct dengan menggunakan typedef dan tanpa menggunakan typedef.

DASAR TEORI

Record atau *struct* merupakan sebuah tipe data bentukan yang berisi kumpulan atribut atau tipe data, baik berbeda-beda maupun sama jenis tipe data, memiliki kaitan antara satu sama lain, dan didalam suatu nama yang sama, berbeda dengan *array* yang hanya dapat diisi oleh satu tipe data yang sama. Dengan menggunakan *record/struct*, kita dapat menyimpan beberapa atribut dengan tipe data yang berbeda di bawah suatu nama.

Dalam penerapannya, masing-masing item di dalam *record* disebut *field* yang berbeda tipe data. Jadi, *record/struct* terdiri dari kumpulan field yang dapat berbeda tipe datanya. Setiap field dapat menyimpan tipe data dasar (int, float, char) maupun tipe data bentukan (*struct/record* lain maupun tipe data yang sudah didefinisikan sebelumnya).


Mengapa kita membutuhkan *record* atau *struct* ?



```
1 string nama_mhs;
2 int umur_mhs;
3 float ipk_mhs;
```

Pada Gambar 1 merupakan cara kita ketika ingin menyimpan sebuah data mahasiswa. Bagaimana jika kita ingin menyimpan data lebih dari 1 mahasiswa?

Gambar 1



```


1  string nama_mhs1;
2  int umur_mhs1;
3  float ipk_mhs1;
4
5  string nama_mhs2;
6  int umur_mhs2;
7  float ipk_mhs2;
8
9  string nama_mhs3;
10 int umur_mhs3;
11 float ipk_mhs3;

```

Gambar 2

Solusinya adalah menambahkan angka pada akhir setiap tipe data (Gambar 2). Cara ini memang dapat digunakan, namun semakin banyak data dan jumlah mahasiswa yang dibutuhkan, maka akan semakin banyak pula variable yang diperlukan, akibatnya code kita menjadi semakin panjang, tidak efektif, dan tidak efisien.

Bagaimana jika semua data di atas kita satukan menggunakan record/struct?



```


1  typedef struct {
2      string nama;
3      int umur;
4      float ipk;
5  } Mahasiswa;
6
7  void main(){
8      Mahasiswa M1, M2, M3;
9  }

```

Gambar 3

Dari potongan kode pada Gambar 3, kita sudah mengatasi masalah-masalah yang dipaparkan di awal. Kita telah membuat suatu record bernama “Mahasiswa”, di dalamnya berisikan field nama, ipk, dan umur. Kemudian untuk menambahkan 3 orang mahasiswa, kita cukup mendeklarasikan tipe data “Mahasiswa” di dalam fungsi/prosedur yang kita inginkan, kemudian memberi nama variabel tersebut, sama seperti tipe data lain.

Bagaimana cara mengakses struct yang sudah kita buat?



```

1  TipeDataRecord record;
2  record.namaField;

```

Gambar 4

Field-field di dalam *record* dapat kita akses dengan memanggil variabel *record*, diikuti tanda titik ‘.’, diakhiri dengan nama field di dalam *record* tersebut (Gambar 4).

Pada Gambar 5 ada beberapa pengaksesan, modifikasi, dan operasi yang dapat dilakukan dalam *record* “Mahasiswa” yang sebelumnya telah kita buat :

```
1 // Mengakses nama dari struct "Mahasiswa" bernama "M1"
2 printf("Nama : %s", M1.nama);
3
4 // Memodifikasi isi field "nama_mhs" dari struct "Mahasiswa" bernama "M2"
5 strcpy(M2.nama, "Nama Baru");
6
7 // Memodifikasi isi field "umur_mhs" dari struct "Mahasiswa" bernama "M2"
8 M2.umur = 19;
9
10 // Menggunakan if-else berdasarkan field "nama_mhs" dari struct "Mahasiswa" bernama "M3"
11 if (strcmp(M3.nama, "Panji")==0){
12     printf("Pemegang Modul 11");
13 }else{
14     printf("Nama belum diinput !");
15 }
16
17 // Menggunakan if-else berdasarkan field "ipk_mhs" dari struct "Mahasiswa" bernama "M3"
18 if (M3.ipk>3.7){
19     printf("Kamu cumlaude");
20 }else{
21     printf("Tidak cumlaude");
22 }
```

Gambar 5

Bagaimana kalau ada *struct* di dalam *struct*?

Contohnya: kita akan membuat record Mahasiswa dan DosenPembimbing. Dimana setiap mahasiswa hanya memiliki 1 dosen pembimbing (Gambar 6).

```
1 //Buatlah struct "DosenPembimbing" sebelum "Mahasiswa" agar dikenali complier
2 typedef struct{
3     string nama;
4     string nip;
5 } DosenPembimbing;
6
7 // Jangan lupa memasukan struct "DosenPembimbing" didalam "Mahasiswa"
8 typedef struct {
9     string nama;
10    int umur;
11    float ipk;
12    DosenPembimbing dospem;
13 } Mahasiswa;
```

Gambar 6

Pada Gambar 7 ada beberapa pengaksesan, modifikasi, dan operasi yang dapat dilakukan dalam *record* “DosenPembimbing” yang ada didalam *record* “Mahasiswa” yang telah kita buat :

```

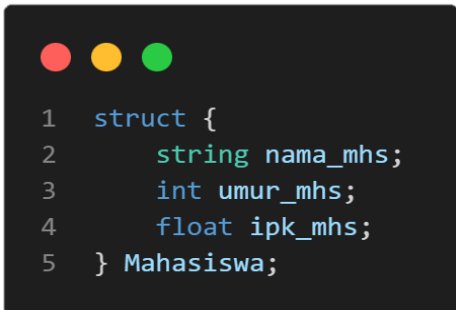
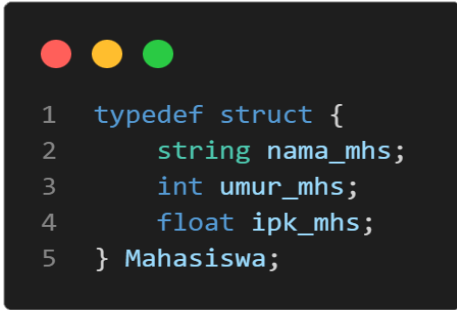
1 void main() {
2     Mahasiswa M1, M2, M3;
3
4     // Memodifikasi isi field "nama_dosen" dan "nip" melalui struct "Mahasiswa" bernama "M1"
5     strcpy(M1.dospem.nama, "Nama Dosen");
6     strcpy(M1.dospem.nip, "123456789");
7
8     // Mengakses isi field "nama_dosen" melalui struct "Mahasiswa" bernama "M2"
9     printf("Nama Dosen : %s", M2.dospem.nama);
10
11    // Menggunakan if-else berdasarkan field "nama_dosen" melalui struct "Mahasiswa" bernama "M3"
12    if (strcmp(M3.dospem.nama, "Nama Dosen")==0){
13        printf("Dosen Dasra Pemrograman");
14    }else{
15        printf("Nama dosen belum diinput !");
16    }
17 }

```

Gambar 7

Kenapa kita menggunakan **typedef**? Apakah bisa kita membuat *record*/struct tanpa **typedef**?

Tentu bisa, mari perhatikan beberapa contoh di bawah:

Tindakan	Tanpa typedef	Dengan typedef
Pendefinisian struct	 <pre> 1 struct { 2 string nama_mhs; 3 int umur_mhs; 4 float ipk_mhs; 5 } Mahasiswa; </pre> <p>Membuat variabel bernama “Mahasiswa” dengan tipe data struct.</p>	 <pre> 1 typedef struct { 2 string nama_mhs; 3 int umur_mhs; 4 float ipk_mhs; 5 } Mahasiswa; </pre> <p>Membuat tipe data buatan bernama “Mahasiswa” bertipe struct/record.</p>

<p>Pendeklarasian struct</p>	<div><pre>1 Mahasiswa.umur = 19; 2 strcpy(Mahasiswa.nama_mhs, "Nama Baru");</pre></div> <p>Setiap variabel struct hanya dapat dipakai sekali saja.</p>	<div><pre>1 Mahasiswa M1, M2, M3; 2 M1.umur_mhs = 19; 3 strcpy(M2.nama_mhs, "Nama Baru");</pre></div> <p>Dikarenakan Mahasiswa merupakan tipe data buatan, kita dapat menggunakannya lebih dari satu kali.</p>												
<p>Pemanggilan struct</p>	<div><pre>struct { string nama_mhs; int umur_mhs; float ipk_mhs; } Mahasiswa; void main() { Mahasiswa.</pre></div> <table><tr><td>variable</td><td>float ipk mhs</td></tr><tr><td>variable</td><td>string nama_mhs</td></tr><tr><td>variable</td><td>int umur_mhs</td></tr></table> <p>Dikarenakan Mahasiswa merupakan sebuah variabel, sehingga kita dapat langsung menggunakannya. Variabel tersebut sudah terdeklarasi di bagian header program.</p>	variable	float ipk mhs	variable	string nama_mhs	variable	int umur_mhs	<div><pre>typedef struct { string nama_mhs; int umur_mhs; float ipk_mhs; } Mahasiswa; void main() { Mahasiswa M1, M2, M3; M1.</pre></div> <table><tr><td>variable</td><td>float ipk mhs</td></tr><tr><td>variable</td><td>string nama_mhs</td></tr><tr><td>variable</td><td>int umur_mhs</td></tr></table> <p>Dikarenakan Mahasiswa merupakan sebuah tipe data, maka kita harus mendeklarasikan variabel M1, M2, dan/atau M3 terlebih dahulu.</p>	variable	float ipk mhs	variable	string nama_mhs	variable	int umur_mhs
variable	float ipk mhs													
variable	string nama_mhs													
variable	int umur_mhs													
variable	float ipk mhs													
variable	string nama_mhs													
variable	int umur_mhs													

Sebuah *record* hanya dapat menampung 1 data *struct* saja. Jika ingin menyimpan *record* sebagai *array*, maka kita dapat menggunakan *array of record*. Di modul ini, kita belum bisa menggunakan *array of record*, karena akan dipelajari di modul 12.

GUIDED 1

Suatu perguruan tinggi ingin membuat program pengelolaan data dosen. Anda diminta untuk membuat program tersebut, sesuai dengan permintaan klien. Program tersebut dibuat dengan ketentuan berikut ini :

1. Buatlah satu buah tipe data bentukan (record) dengan ketentuan sebagai berikut:

Dosen
nama : string
mataKuliah : string
gaji : float

2. Inisialisasilah struct tersebut dengan data kosong sebelum memasuki menu program.
3. Buatlah 2 menu dengan ketentuan sebagai berikut :

- a. Menu 1 : Input Data

Buatlah menu 1 yang berfungsi untuk menginput data dosen. Menu ini hanya bisa diakses jika data dari dosen kosong. Sebaliknya, jika data dosen telah terisi maka akan menampilkan error handling.

- b. Menu 2 : Show Data

Menu ini hanya bisa diakses jika data dosen sudah diisi. Tampilkan seluruh data yang sudah diinputkan pada menu 1. Tambahkan informasi total gaji dosen.

Rumus total gaji dosen : $\text{gaji} - (\text{gaji} * \text{pajak } (11\%))$

4. Tambahkan menu keluar yang berfungsi untuk keluar program. Menu ini dapat diakses kapanpun. Tambahkan identitas kamu dengan format :

“<Nama Praktikan> | <NPM lengkap> | <Kelas>”

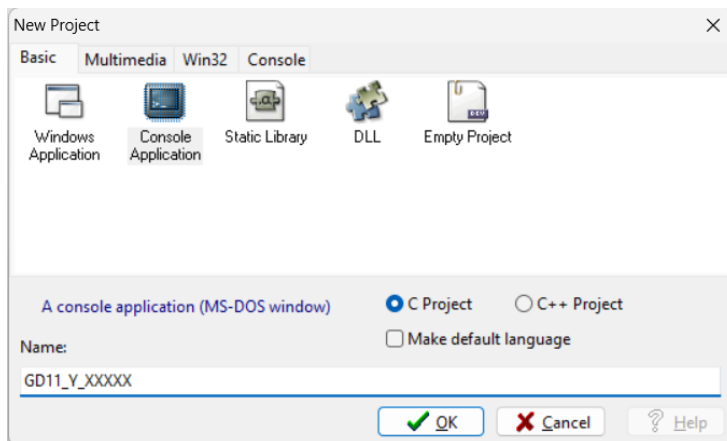
Contoh : “Alphonsus Dio Yuan Ananda | 220711671 | A”

Beberapa prosedur dan fungsi yang digunakan di jawaban Guided (optional):

```
void showMenu();
void initData(Dosen *D);
void inputData(Dosen *D, string nama,
string mataKuliah, float gaji);
bool isEmpty(Dosen D);
void tampilData(Dosen D);
float hitungGaji(Dosen D);
```

KETENTUAN Pengerjaan

Dikerjakan secara mandiri, tanpa copy-paste dari teman, karena pemahaman akan guided akan sangat berguna di unguided, tugas, dan untuk kedepannya.



Penamaan folder dan archive : **GD11_Y_XXXXX** Keterangan

format penamaan:

Y = kelas

X = 5 digit terakhir NPM

Nilai GD -5 apabila format penamaan salah dan ketentuan lain dalam spreadsheet berlaku.

JAWBAN GUIDED

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdbool.h>

#define pajak 0.11

typedef char string[50];

typedef struct{
    string nama;
    string mataKuliah;
    float gaji;
}Dosen;

void showMenu();
void initData(Dosen *D);
void inputData(Dosen *D, string nama, string mataKuliah, float gaji);
bool isEmpty(Dosen D);
void tampilData(Dosen D);
float hitungGaji(Dosen D);

int main(int argc, char *argv[]) {
    Dosen D;
    string nama, mataKuliah;
    float gaji;
    int menu;

    //Inisialisasi Data Dosen
    initData(&D);

    do{
        system("cls");
        showMenu();
        printf("\n>>> ");scanf("%d", &menu);

        switch(menu){
            case 1:
                if(isEmpty(D)){
                    printf("\n\t---[ Input Data Dosen ]---\n");
                    /*
                     * Menu ini dapat diakses jika data masih kosong / belum terisi
                     */
                    //Meminta inputan nama dosen
                    printf("\nMasukkan nama Dosen\t\t\t: ");fflush(stdin);gets(nama);
                    //Kalian boleh menambahkan error handling inputan tidak boleh kosong (jika mau)

                    //Meminta inputan mata kuliah yang diampu
                    printf("\nMasukkan Mata Kuliah yang diampu\t: ");fflush(stdin);gets(mataKuliah);
                    //Kalian boleh menambahkan error handling inputan tidak boleh kosong (jika mau)

                    //Meminta inputan gaji dosen
                    printf("\nMasukkan gaji Dosen\t\t\t: ");scanf("%f", &gaji);
                    //Kalian boleh menambahkan error handling inputan tidak boleh kurang dari 1 (jika mau)

                    inputData(&D, nama, mataKuliah, gaji);
                    printf("\n\tBerhasil memasukkan data!");
                }else{
                    printf("\n\t[!] Data Sudah Di Isi [!]\n");
                }
                break;

            case 2:
                if(!isEmpty(D)){
                    printf("\n\t===[ Show Data Dosen ]===\n");
                    /*
                     * Menu ini dapat diakses ketika data tidak kosong
                     * Untuk show data dosen menggunakan prosedur dibawah ini
                     */
                    tampilData(D);
                }else{
                    printf("\n\t[!] Anda Belum Mengisi Data [!]\n");
                }
                break;
        }
    }while(1);
}
```



```

        case 0:
            printf("\n\tNama Praktikan | NPM | Kelas");
            break;

        default:
            printf("\n\t[!] Menu Tidak Tersedia [!];");
            break;
    }
    getch();
}while(menu!=0);

return 0;
}

void showMenu(){
    /*
        Prosedur ini akan menampilkan menu-menu yang ada pada program ini
    */
    printf("\n\t[ GUIDED RECORD ]\n");
    printf("[1] Input Data\n");
    printf("[2] Tampil Data\n");
    printf("[0] Exit");
}

void initData(Dosen *D){
    /*
        Prosedur ini akan melakukan inisialisasi semua field yang ada di dalam struct
        untuk field string akan diisikan dengan "" atau "-", dan
        untuk field numerik seperti int atau float akan diisikan dengan 0
    */
    strcpy((*D).nama, "-");
    strcpy((*D).mataKuliah, "-");
    (*D).gaji = 0.0;
}

void inputData(Dosen *D, string nama, string mataKuliah, float gaji){
    /*
        Prosedur ini akan melakukan pengisian semua field yang ada di dalam struct
        berdasarkan dengan inputan dari pengguna dari dalam main program
    */
    strcpy((*D).nama, nama);
    strcpy((*D).mataKuliah, mataKuliah);
    (*D).gaji = gaji;
}

bool isEmpty(Dosen D){
    /*
        Fungsi ini akan melakukan pengecekan apakah field dari struct yang ada itu
        masih kosong atau tidak, disini kita menggunakan field nama yang digunakan
        untuk membandingkan, jika nama sama dengan "-" maka akan mereturnkan true,
        sedangkan jika sebaliknya maka akan mereturnkan false
    */
    if(strcmp(D.nama, "-")==0){
        return true;
    }else{
        return false;
    }
}

void tampilData(Dosen D){
    /*
        Prosedur ini akan menampilkan seluruh data yang sudah diinputkan, beserta
        total gaji dosen
    */
    printf("\n\tNama Dosen\t\t: %s", D.nama);
    printf("\n\tMata Kuliah Diampu\t: %s", D.mataKuliah);
    printf("\n\tGaji Dosen\t\t: Rp. %.2f", D.gaji);
    printf("\n\tTotal Gaji Dosen\t: Rp. %.2f", hitungGaji(D));
}

float hitungGaji(Dosen D){
    /*
        Menghitung total gaji setelah dikurangi pajak
    */
    return D.gaji - (D.gaji * pajak);
}

```

GUIDED 2

Tambahkan 2 menu lagi, yaitu:

1. Menu 3 : Hapus Data

Menu ini hanya bisa diakses jika data dari dosen sudah diisi. Akan diminta inputan nama dosen yang akan dihapus, jika nama yang diinputkan sama dengan nama yang sudah diinputkan pada menu 1, maka data akan dihapus. Sebaliknya, jika nama dosen tidak ditemukan maka akan menampilkan error handling.

2. Menu 4: Update Data

Menu ini hanya bisa diakses jika data dari dosen sudah diisi. Akan diminta inputan nama dosen yang akan diupdate, jika nama yang diinputkan sama dengan nama yang sudah diinputkan pada menu 1, maka data akan diupdate (data yang dapat diupdate berupa nama, mata kuliah, dan gaji). Sebaliknya, jika nama dosen tidak ditemukan maka akan menampilkan error handling.

Tampilan Guided 2:

```
          [ GUIDED RECORD ]
[1] Input Data
[2] Tampil Data
[3] Hapus Data
[4] Update Data
[0] Exit
>>>
```

Gambar 8

```
          [ GUIDED RECORD ]
[1] Input Data
[2] Tampil Data
[3] Hapus Data
[4] Update Data
[0] Exit
>>> 4

      ===[ Update Data Dosen ]===

      Masukkan nama Dosen      : Paulus Mudjihartono, ST, MT, PhD

-----
Masukkan nama Dosen            : Joanna Ardhyanti Mita Nugraha, S.Kom., M.Kom
Masukkan Mata Kuliah yang diampu : Dasar Pemrograman
Masukkan gaji Dosen            : 10000000

      [!] Berhasil Update Data [!]
```

Gambar 9

```
[ GUIDED RECORD ]
[1] Input Data
[2] Tampil Data
[3] Hapus Data
[4] Update Data
[0] Exit
>>> 3

===[ Delete Data Dosen ]===

Masukkan nama Dosen      : Joanna Ardhyanti Mita Nugraha, S.Kom., M.Kom

[!] Berhasil Hapus Data [!]
```

Gambar 10

Persiapan UGD :

1. Pelajari kembali fungsi dan prosedur secara umum.
2. Pelajari kembali array.
3. Pelajari *struct* didalam *struct* dan variabel lebih dari 1.