Modul 11

RECORD

PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ATMA JAYA YOGYAKARTA

Oleh Alexander Wongso

A. Tujuan

1. Praktikan dapat mengerti dan memahami penggunaan record/struct dalam bahasa pemrograman C.
2. Praktikan dapat mengerti kegunaan record/struct 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.

B. 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. Perbedaan antara array dan record adalah setiap data yang tersimpan dalam array harus memiliki tipe data yang sama, sedangkan dalam record, tipe datanya dapat berbeda.

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.

Mengapa kita membutuhkan record atau struct?

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

Gambar 1

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

```
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 seperti pada **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 efisien.**

Bagaimana jika semua data di atas kita satukan menggunakan record atau 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 **string nama, int umur, dan float ipk**. Kemudian untuk menambahkan 3 orang mahasiswa, kita cukup mendeklarasikan tipe data **Mahasiswa** di dalam fungsi/prosedur yang kita inginkan, kemudian memberi nama variabel tersebut, contohnya **M1, M2, dan M3**, sama seperti tipe data lain.

Bagaimana cara mengakses struct yang sudah kita buat?

```
TipeDataRecord record;
record.namaField;
```

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

Gambar 4

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

```
// Mengakses nama dari struct "Mahasiswa" bernama "M1"
printf("Nama Mahasiswa : %s", M1.nama);

// Memodifikasi isi field "nama_mhs2" dari struct "Mahasiswa" yang bernama "M2"
strcpy(M2.nama,"Nama");

// Memodifikasi isi field "umur_mhs2" dari struct "Mahasiswa" yang bernama "M2"
M2.umur = 19;

// Menggunakan if-else berdasarkan field "ipk_mhs3" dari struct "Mahasiswa" yang bernama "M3"
if(M3.ipk > 3.7){
  printf("Cumlaude")
}else{
  printf("Tidak Cumlaude");
}
```

Gambar 5

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

Kita akan membuatkan *record* **Mahasiswa** dan **Dosen Pembimbing**. Dimana setiap mahasiswa hanya memiliki 1 dosen pembimbing pada **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:

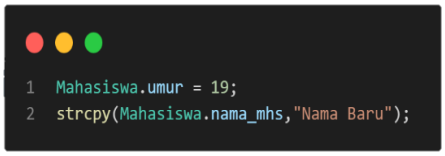
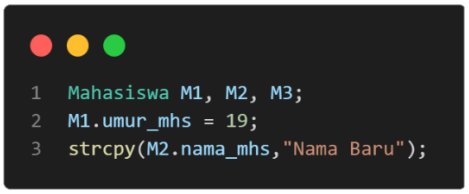
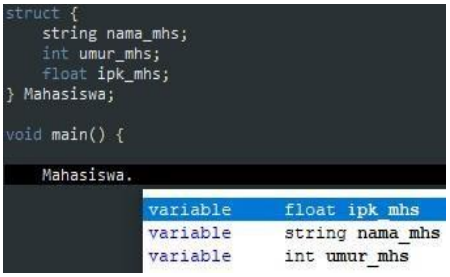
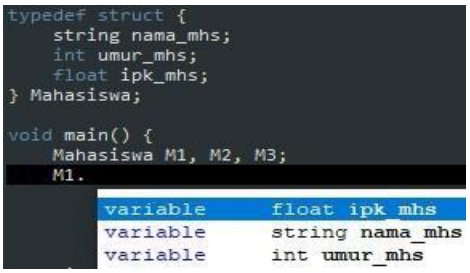
```c
void main() {
    Mahasiswa M1, M2, M3;

    // Memodifikasi isi field "nama_dosen" dan "nip" melalui struct "Mahasiswa" bernama "M1"
    strcpy(M1.dospem.nama, "Nama Dosen");
    strcpy(M1.dospem.nip, "123456789");

    // Mengakses isi field "nama_dosen" melalui struct "Mahasiswa" bernama "M2"
    printf("Nama Dosen : %s", M2.dospem.nama);

    // Menggunakan if-else berdasarkan field "nama_dosen" melalui struct "Mahasiswa" bernama "M3"
    if (strcmp(M3.dospem.nama,"Nama Dosen")==0){
        printf("Dosen Dasra Pemrograman");
    }else{
        printf("Nama dosen belum diinput !");
    }
}
```

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 { string nama_mhs; int umur_mhs; float ipk_mhs; } Mahasiswa;` Membuat variabel bernama "Mahasiswa" dengan tipe data struct.	`typedef struct { string nama_mhs; int umur_mhs; float ipk_mhs; } Mahasiswa;` Membuat tipe data buatan bernama "Mahasiswa" bertipe struct/record.

Pendeklarasian	 Setiap variabel struct hanya dapat dipakai sekali saja.	 Dikarenakan Mahasiswa merupakan tipe data buatan, kita dapat menggunakannya lebih dari satu kali.
Pemanggilan	 Dikarenakan Mahasiswa merupakan sebuah variabel, sehingga kita dapat langsung menggunakannya. Variabel tersebut sudah terdeklarasi di bagian header program.	 Dikarenakan Mahasiswa merupakan sebuah tipe data, maka kita harus mendeklarasikan variabel M1, M2, dan/atau M3 terlebih dahulu.

Dari perbedaan diatas kita dapat mengetahui bahwa kata kunci typedef adalah kata kunci untuk membuat tipe data baru di C. Saat kita menggunakan typedef untuk struct, maka struct tersebut akan dikenali sebagai tipe data. Sehingga saat menggunakan struct, kita tidak perlu lagi pakai kata kunci struct. 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

Sebuah perusahaan sedang membuka lowongan pekerjaan, jadi perusahaan tersebut ingin membuat program pengelolaan karyawan. Anda sebagai programmer yang handal diminta untuk membuat program tersebut dengan ketentuan sebagai berikut ini:

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

Karyawan
nama : string jabatan : string gaji : float

2. Inisialisasikanlah struct tersebut dengan data kosong sebelum memasuki menu program.
3. Buatlah 4 menu dengan ketentuan sebagai berikut :

   a. Menu 1 : **Input Data**

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

   b. Menu 2 : **Show Data**

   Menu ini hanya bisa diakses jika data karyawan sudah diinput. Tampilkan seluruh data yang sudah diinputkan pada menu 1. Tambahkan total gaji dimana total gaji didapatkan setelah gaji dikurangi dengan pajak. Apabila gaji diatas Rp 100.000 maka akan dikenakan pajak 2%

   c. Menu 3 : **Delete Data**

   Menu ini hanya bisa diakses jika data karyawan sudah diinput. Kemudian program akan melakukan konfirmasi sebelum menghapus data. Apabila inputan “Yes” maka data karyawan akan dihapus. Sebaliknya, penghapusan akan dibatalkan.

d. Menu 4 : **Update Data**

Menu ini hanya bisa diakses jika data karyawan sudah diinput. Kemudian program akan melakukan konfirmasi sebelum mengganti data. Apabila inputan “Yes” maka user akan diminta untuk mengisi data karyawan yang baru seperti menu 1. Sebaliknya, update akan dibatalkan.

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

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

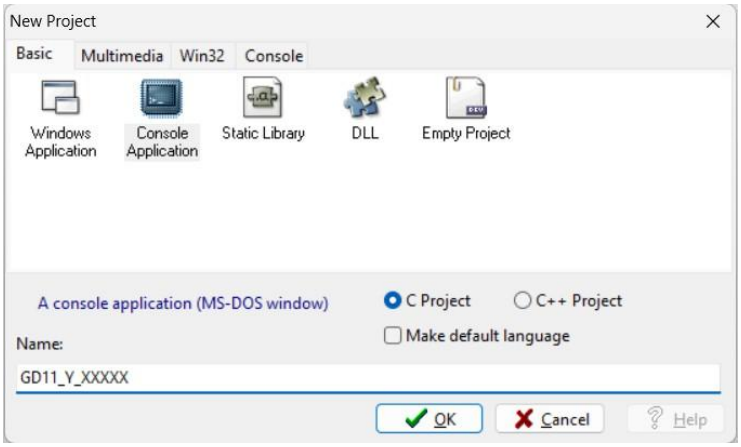
   Contoh : **Alexander Wongso | 230712519 | D**

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

```
void showmenu();
void initData(Karyawan *K);
void KaryawanBuilder(Karyawan *K, str nama, str jabatan, float gaji);
bool isEmpty(Karyawan K);
void tampilData(Karyawan K);
float totalGaji(Karyawan K);
```

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 arcive : **GD11_Y_XXXXX**

Keterangan format penamaan:

Y = kelas

X = 5 digit terakhir NPM

JAWABAN GUIDED

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

typedef char str[100];

typedef struct{
    str nama;
    str jabatan;
    float gaji;
}Karyawan;

void showmenu();
void initData(Karyawan *K);
void KaryawanBuilder(Karyawan *K, str nama, str jabatan, float gaji);
bool isEmpty(Karyawan K);
void tampilData(Karyawan K);
float totalGaji(Karyawan K);

int main(int argc, char *argv[]) {
    Karyawan K;
    char menu;
    str YesorNo, nama, jabatan;
    float gaji;

    //Inisialisasi Data Karyawan
    initData(&K);

    do
    {
        system("cls");
        showmenu();menu = getche();
        getch();
        switch(menu)
        {
            case '1':
                if(isEmpty(K))
                {
                    /*
                        Menu ini dapat diakses jika data masih kosong/belum diisi
                        Untuk input data dosen menggunakan prosedur dibawah ini
                    */
                    printf("\n\t\t=== [Input Data] ===\n");
                    while(1)
                    {
                        printf("\n\tMasukan nama karyawan : ");fflush(stdin);gets(nama);
                        if(strcmp(nama,"-")==0 || strlen(nama)==0)
                        {
                            printf("\n\t[!] Nama Karyawan tidak boleh - atau kosong [!]\n");
                        }else break;
                    }
                    while(1)
                    {
                        printf("\n\tMasukan jabatan karyawan : ");fflush(stdin);gets(jabatan);
                        if(strlen(jabatan)==0)
                        {
                            printf("\n\t[!] Jabatan Karyawan tidak boleh kosong [!]\n");
                        }else break;
                    }
                    while(1)
                    {
                        printf("\n\tMasukan gaji karyawan : ");scanf("%f", &gaji);
                        if(gaji <= 0)
                        {
                            printf("\n\t[!] Gaji tidak boleh kurang atau sama dengan 0 [!]\n");
                        }else break;
                    }
                    KaryawanBuilder(&K, nama, jabatan, gaji);
                    printf("\n\t[+] Berhasil Input Data [+]");
                }else
                {
                    printf("\n\t[!] Data sudah terisi [!]");
                }
            break;
```

```c
case '2':
    if(!isEmpty(K))
    {
        /*
            Menu ini dapat diakses ketika data tidak kosong
            Untuk show data dosen menggunakan prosedur dibawah ini
        */
        printf("\n\t\t=== [Show Data] ===\n");
        tampilData(K);
    }else
    {
        printf("\n\t[!] Data masih kosong [!]");
    }
break;

case '3':
    if(!isEmpty(K))
    {
        /*
            Menu ini dapat diakses ketika data tidak kosong
            Untuk delete data dosen menggunakan prosedur dibawah ini
        */
        printf("\n\t\t=== [Delete Data] ===\n");
        printf("\n\tApakah kamu yakin ingin menghapus Data Karyawan? [YES/NO]");
        printf("\n\t>>> ");fflush(stdin);gets(YesorNo);
        if(strcmpi(YesorNo,"Yes")==0)
        {
            initData(&K);
            printf("\n\t[+] Berhasil hapus data [+]");
        }else
        {
            printf("\n\t[!] Penghapusan dibatalkan [!]");
        }
    }else
    {
        printf("\n\t[!] Data masih kosong [!]");
    }
break;

case '4':
    if(!isEmpty(K))
    {
        /*
            Menu ini dapat diakses ketika data tidak kosong
            Untuk update data dosen menggunakan prosedur dibawah ini
        */
        printf("\n\t\t=== [Update Data] ===\n");
        printf("\n\tApakah kamu yakin ingin mengganti Data Karyawan? [YES/NO]");
        printf("\n\t>>> ");fflush(stdin);gets(YesorNo);
        if(strcmpi(YesorNo,"Yes")==0)
        {
            while(1)
            {
                printf("\n\tMasukan nama karyawan baru : ");fflush(stdin);gets(nama);
                if(strcmp(nama,"-")==0 || strlen(nama)==0)
                {
                    printf("\n\t[!] Nama Karyawan tidak boleh - atau kosong [!]\n");
                }else break;
            }
            while(1)
            {
                printf("\n\tMasukan jabatan karyawan baru : ");fflush(stdin);gets(jabatan);
                if(strlen(jabatan)==0)
                {
                    printf("\n\t[!] Jabatan Karyawan tidak boleh kosong [!]\n");
                }else break;
            }
            while(1)
            {
                printf("\n\tMasukan gaji karyawan baru : ");scanf("%f", &gaji);
                if(gaji <= 0)
                {
                    printf("\n\t[!] Gaji tidak boleh kurang atau sama dengan 0 [!]\n");
                }else break;
            }
            KaryawanBuilder(&K, nama, jabatan, gaji);
            printf("\n\t[+] Berhasil update data [+]");
        }else
        {
            printf("\n\t[!] Update dibatalkan [!]");
        }
    }else
    {
        printf("\n\t[!] Data masih kosong [!]");
    }
break;
```

```
            case 27:
                // menekan esc untuk keluar dari program
                printf("\n\n\t~Nama Praktikan | NPM | Kelas");
            break;

            default:
                printf("\n[!] Menu Tidak Ada [!]");
            break;
        }getch();
    }while(menu!=27);
    return 0;
}

void showmenu()
{
    // Prosedur ini akan menampilkan menu-menu yang ada pada program ini
    puts("\t  =============");
    puts("\t< Guided Record >");
    puts("\t  =============\n");
    puts("[1] Input Data Karyawan");
    puts("[2] Show Data Karyawan");
    puts("[3] Delete Data Karyawan");
    puts("[4] Update Data Karyawan");
    puts("[0] Exit");
    printf(">>> ");
}

void initData(Karyawan *K)
{
    /*
        Prosedur ini 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((*K).nama, "-");
    strcpy((*K).jabatan,"-");
    (*K).gaji = 0.0;
}

void KaryawanBuilder(Karyawan *K, str nama, str jabatan, float gaji)
{
    /*
        Prosedur ini akan melakukan inputan semua field yang ada di dalam struct
        berdasarkan dengan inputan dari pengguna dari dalam main program
    */
    strcpy((*K).nama, nama);
    strcpy((*K).jabatan, jabatan);
    (*K).gaji = gaji;
}

void tampilData(Karyawan K)
{
    /*
        Prosedur ini akan menampilkan seluruh data yang sudah diinputkan
        dan total gaji karyawan
    */
    printf("\n\tNama Karyawan\t\t: %s", K.nama);
    printf("\n\tJabatan Karyawan\t: %s", K.jabatan);
    printf("\n\tGaji Karyawan\t\t: Rp %.2f", K.gaji);
    printf("\n\tTotal Gaji Karyawan\t: Rp %.2f", totalGaji(K));
}
```

```
bool isEmpty(Karyawan K)
{
    /*
        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 return true,
        sedangkan jika sebaliknya maka akan return false
    */
    if(strcmp(K.nama,"-")==0)
    {
        return true;
    }
    return false;
}

float totalGaji(Karyawan K)
{
    /*
        Fungsi ini akan mengecek apakah gaji lebih besar dari 100000 atau tidak
        Apabila lebih dari 100000 maka pajak = gaji dikali dengan 2%
        Sebaliknya pajak = 0. Kemudian akan mengreturnkan gaji asli dikurangi pajak
    */
    float pajak;

    if(K.gaji > 100000)
    {
        pajak = 0.02 * K.gaji;
    }else
    {
        pajak = 0;
    }
    return K.gaji - pajak;
}
```

Hint Untuk UGD :

1. Pelajari kembali fungsi dan prosedur secara umum.
2. Pelajari *struct* didalam *struct* dan variabel lebih dari 1.
3. Pelajari kode ascii.
4. Pelajari sintaks Sleep( ) dan sleep( ).
5. Pelajari (angka random) = rand()%(max-min+1)+min

PENUTUP

Kontak Asdos

Masih bingung? Punya pertanyaan atau ingin diskusi lebih lanjut?
Jangan ragu untuk kontak saya😊

- Teams **230712519@students.uajy.ac.id**
- Whatsapp **085261259680**

SELAMAT BELAJAR

ubur-ubur ikan lele, semangat belajarnya lee!