

## MODUL 12

### ARRAY OF RECORD



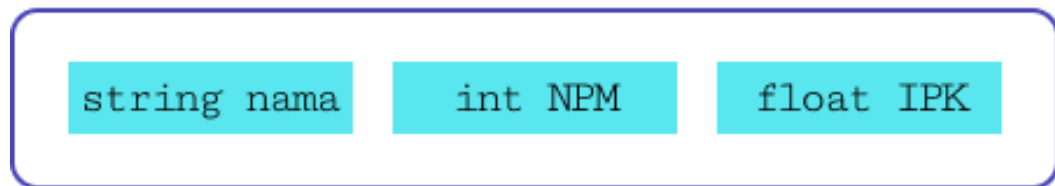
#### I. Tujuan

1. Praktikan dapat memahami dan mengimplementasikan konsep Array of Record (AoR) dalam bahasa pemrograman C.
2. Praktikan dapat menyelesaikan studi kasus dengan menggunakan konsep Array of Record.

## II. Pengantar

Pada modul-modul sebelumnya, kalian telah mempelajari mengenai Array dan Record dalam bahasa C. **Array** merupakan sekumpulan variabel dengan tipe data yang sama yang disimpan secara berurutan dan diakses dengan menggunakan index, yang merupakan angka yang menunjukkan posisi elemen tersebut dalam urutan penyimpanan. Sementara itu, **Record** merupakan sebuah tipe data bentukan yang berisi kumpulan item atau atribut dengan tipe data yang mungkin berbeda-beda. Misalnya, sebuah Record Mahasiswa bisa menyimpan data-data seperti `string nama`, `int NPM`, `float IPK`, dan lain sebagainya, seperti gambar di bawah ini.

### Mahasiswa




Gambar 1. Visualisasi Struct Mahasiswa

Tapi, dua konsep ini kan hal lumayan berbeda ya? Trus nyambungnya di mana sampai akhirnya jadi Array of Record?

```
1  typedef struct{
2      string nama;
3      int NPM;
4      float ipk;
5  } Mahasiswa;
6
7
8  int main(int argc, char *argv[]) {
9      Mahasiswa mahasiswa1, mahasiswa2, mahasiswa3;
10
11     return 0;
12 }
```

Gambar 2. Pembuatan struct Mahasiswa dan pendeklarasian 3 variabel bertipe data Mahasiswa

Perhatikan potongan code di atas. Kita telah memiliki sebuah tipe data bentukan (Record) Mahasiswa dengan atribut nama, NPM, dan IPK. Kemudian, kita membuat 3 buah variabel bertipe-data Mahasiswa untuk menampung data 3 orang mahasiswa. Tapi, bagaimana jika kita mau menyimpan lebih dari 3 mahasiswa? Apa jadinya kalau kita mau membuat 100 data Mahasiswa?

A screenshot of a code editor with a dark background and light-colored text. The code is written in C and shows a function `main` that declares 100 variables of type `Mahasiswa`, from `mahasiswa1` to `mahasiswa100`. The code is as follows:

```
1 int main(int argc, char *argv[]) {  
2     Mahasiswa mahasiswa1, mahasiswa2, mahasiswa3, mahasiswa4, mahasiswa5,  
3     mahasiswa6, mahasiswa7, mahasiswa8.... mahasiswa100;  
4  
5     return 0;  
6 }
```

Gambar 3. Mendeklarasikan 100 variabel bertipe data Mahasiswa secara manual

Blenger kalau kita buat variabelnya satu per satu seperti di atas. Maka dari itu, kita akan menggunakan sebuah konsep baru yaitu **Array of Record (AoR)** untuk memudahkan persoalan tersebut. Di modul ini kita akan belajar mengenai konsep penggunaan AoR mulai dari deklarasi hingga modifikasi data.

### III. Apa Itu Array of Record?

Pada dasarnya, Array of Record ya... *array dari tipe data bentukan (record)*, cuma penggabungan dari dua modul sebelumnya kok. Karena record bisa diperlakukan seperti tipe data biasa pada umumnya (int, float, string, dll), maka kita juga bisa menerapkan array pada record, di mana akan ada sekumpulan variabel dengan tipe data bentukan yang disimpan secara berurutan.

## IV. Penggunaan Array of Record

### a. Deklarasi

Seperti yang sudah dipelajari, perlakuan kita dalam membuat array bertipe data bentukan itu sama dengan membuat array dengan tipe data lainnya, yaitu dengan rumus:

```
TipeData namaVariabel[isiArray];
```

Karena perilaku yang sama terhadap tipe data bentukan, kita dapat mendeklarasikan 100 variabel Mahasiswa seperti mendeklarasikan array pada umumnya, seperti contoh di bawah ini.

A screenshot of a code editor with a dark background and light-colored text. The code is written in C and defines a struct for a student record, then declares an array of 100 such records in the main function. The code is as follows:

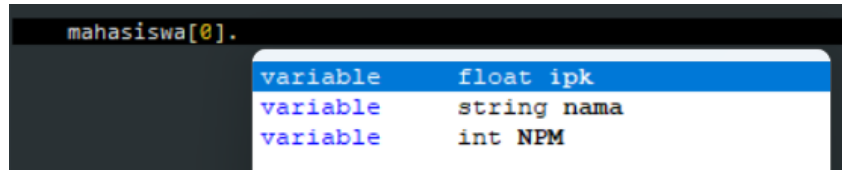
```
1 typedef struct{
2     string nama;
3     int NPM;
4     float ipk;
5 } Mahasiswa;
6
7 int main(int argc, char *argv[]) {
8     Mahasiswa mahasiswa[100];
9     // Tipe data --> Mahasiswa
10    // Nama variabel --> mahasiswa
11    // Isi array --> 100
12
13    return 0;
14 }
```

Gambar 4. Mendeklarasikan 100 variabel bertipe data Mahasiswa dengan konsep Array of Record

### b. Mengakses Index Pada Array of Record

Pengaksesan index pada Array of Record sama seperti array pada umumnya. Yang menjadi pembeda dengan array bertipe data primitif adalah adanya tambahan field yang akan diakses, seperti mengakses field dalam Record yang menggunakan **tanda titik (.)**. Mengakses index pada Array of Record menggunakan rumus

```
namaVariabel[index].field;
```



Gambar 5. Mengakses index & field pada Array of Record

Perhatikan cara mengakses array of record di atas. Setelah pemanggilan nama variabel array, harus diikuti dengan **kurung siku ( [ ] )** beserta **index dari array yang mau diakses (0 ... n)**. Kemudian, diikuti dengan **tanda titik ( . )** yang otomatis akan menampilkan field yang dapat di akses. Kalian dapat melihat cara mengakses field pada Array of Record pada gambar di bawah.

```

1  int main(int argc, char *argv[]) {
2      Mahasiswa mahasiswa[100];
3
4      int i = 0;
5
6      // Mengakses Array of Record index ke-0
7      strcpy(mahasiswa[i].nama, "Dhiaz");
8      printf("\nNama mahasiswa pada index ke-%d adalah %s", i, mahasiswa[i].nama);
9
10     i++; // i = i + 1 --> i = 1
11     // Mengakses Array of Record index ke-1
12     strcpy(mahasiswa[i].nama, "Pieter");
13     printf("\nNama mahasiswa pada index ke-%d adalah %s", i, mahasiswa[i].nama);
14
15     i++; // i = i + 1 --> i = 2
16     // Mengakses Array of Record index ke-2
17     strcpy(mahasiswa[i].nama, "Hary");
18     printf("\nNama mahasiswa pada index ke-%d adalah %s", i, mahasiswa[i].nama);
19
20     return 0;
21 }
22

```

```

D:\Folder\Info Kuliah\Asisten x + v

Nama mahasiswa pada index ke-0 adalah Dhiaz
Nama mahasiswa pada index ke-1 adalah Pieter
Nama mahasiswa pada index ke-2 adalah Hary
-----
Process exited after 0.1281 seconds with return value 0
Press any key to continue . . .

```

Gambar 6 & 7. Contoh pengaksesan Array of Record dan hasilnya

Mari kita analisa potongan code di atas. Pertama, kita melakukan deklarasi struct Mahasiswa dengan field string nama, int NPM, dan float IPK. Kemudian dideklarasikan sebuah Array of Record mahasiswa[100]. Setelah Array of Record mahasiswa dideklarasikan, dilakukan deklarasi variabel int i yang diassign dengan nilai 0. Nantinya, variabel i ini akan menunjukkan index dari mahasiswa yang akan diakses dan di-increment agar kita dapat mengakses variabel pada index berikutnya.

### c. Pengisian dan Pengubahan Data

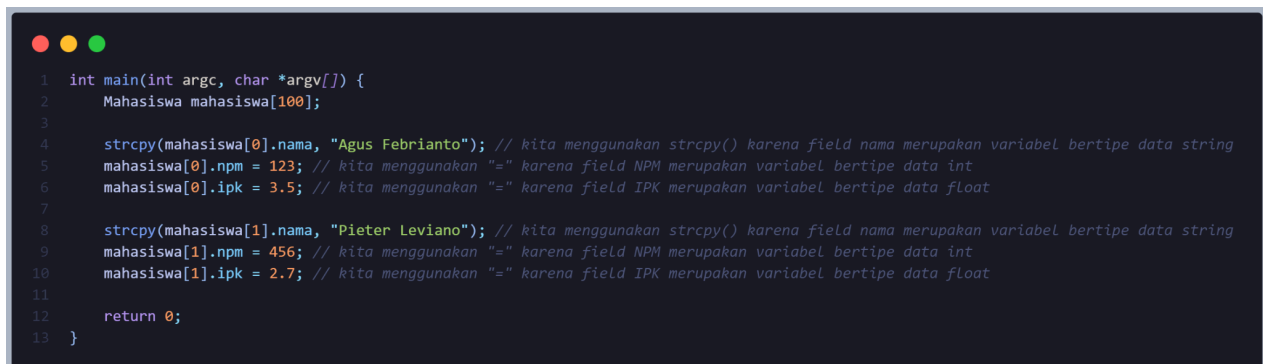
Sama seperti cara mengakses index di poin sebelumnya, mengisi nilai atau mengubah nilai yang ada pada Array of Record bisa dengan menggunakan rumus:

```
namaVariabel[index].field = nilaiBaru;
```

Atau

```
strcpy(namaVariabel[index].field, nilaiStringBaru);
```

Operasi pengisian data pada Array of Record harus memperhatikan tipe data yang dimiliki oleh field yang ingin diisi atau dimodifikasi. Tipe data yang dimiliki field sendiri sudah kita bentuk ketika kita membuat tipe data bentukan Mahasiswa di bagian header. Berikut ini adalah potongan code yang berisi pengisian nilai pada setiap field Mahasiswa dengan mengimplementasikan Array of Record.

A screenshot of a code editor with a dark background and light-colored text. The code is in C and defines a main function that initializes an array of student records. The array is named 'mahasiswa' and has 100 elements. The first two elements are initialized with specific data: the first student is 'Agus Febrianto' with NPM 123 and IPK 3.5; the second student is 'Pieter Leviano' with NPM 456 and IPK 2.7. Comments in Indonesian explain the use of 'strcpy' for string fields and '=' for integer and float fields. The main function returns 0.

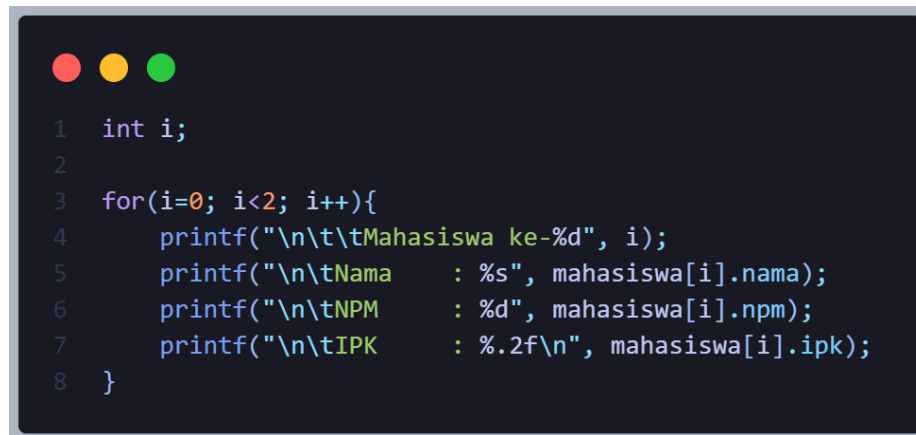
```
1 int main(int argc, char *argv[]) {
2     Mahasiswa mahasiswa[100];
3
4     strcpy(mahasiswa[0].nama, "Agus Febrianto"); // kita menggunakan strcpy() karena field nama merupakan variabel bertipe data string
5     mahasiswa[0].npm = 123; // kita menggunakan "=" karena field NPM merupakan variabel bertipe data int
6     mahasiswa[0].ipk = 3.5; // kita menggunakan "=" karena field IPK merupakan variabel bertipe data float
7
8     strcpy(mahasiswa[1].nama, "Pieter Leviano"); // kita menggunakan strcpy() karena field nama merupakan variabel bertipe data string
9     mahasiswa[1].npm = 456; // kita menggunakan "=" karena field NPM merupakan variabel bertipe data int
10    mahasiswa[1].ipk = 2.7; // kita menggunakan "=" karena field IPK merupakan variabel bertipe data float
11
12    return 0;
13 }
```

Gambar 8. Pengisian nilai pada setiap field Mahasiswa

Ingat, apabila field bertipe data **string**, maka kita akan menggunakan **strcpy()** untuk set nilai dari field tersebut. Sementara untuk tipe data lain seperti **int**, **float**, **bool**, kita dapat menggunakan simbol **sama dengan (=)** untuk set nilai dari field tersebut.

#### d. Penggunaan Perulangan dalam Array of Record

Kita juga bisa menggunakan perulangan untuk memudahkan kita memodifikasi atau menampilkan nilai field pada Array of Record, seperti contoh code di bawah ini.



```
1  int i;
2
3  for(i=0; i<2; i++){
4      printf("\n\t\tMahasiswa ke-%d", i);
5      printf("\n\tNama      : %s", mahasiswa[i].nama);
6      printf("\n\tNPM       : %d", mahasiswa[i].npm);
7      printf("\n\tIPK        : %.2f\n", mahasiswa[i].ipk);
8  }
```

Gambar 9. Penampilan nilai pada setiap field Mahasiswa

Kita menggunakan perulangan yang dimulai dari index ke-0 hingga index ke-2. Nilai index dalam kurung siku [ ] diganti dengan variabel *i*, sehingga program akan menampilkan nilai mahasiswa sesuai dengan nilai index *i* dan bertambah seiring berjalannya perulangan.



```

      Nama      : Agus Febrianto
      NPM       : 123
      IPK        : 3.50

      Nama      : Pieter Leviano
      NPM       : 456
      IPK        : 2.70
```

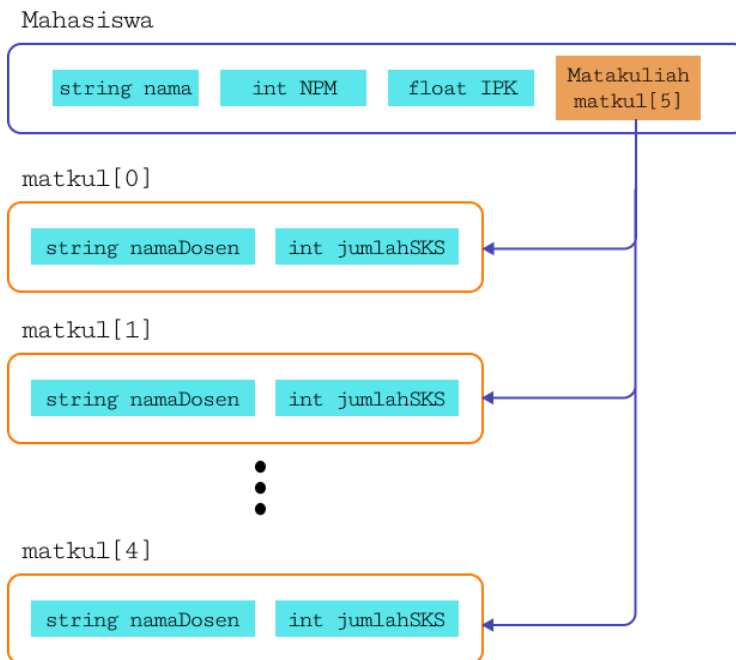
Gambar 10. Output dari penampilan nilai Array of Record dengan perulangan

## V. Array of Record dalam Array of Record

Mungkin kalian bingung (atau bahkan takut dikit) setelah lihat judulnya. Tenang aja, hal ini tidak semengerikan yang kalian bayangkan. Array of Record di dalam Record adalah sebuah kondisi dimana suatu record memiliki atribut atau field yang berupa array of record.

### a. Deklarasi

Untuk memudahkan pemahaman kalian, coba lihat ilustrasi di bawah ini.



```
1  typedef struct{
2      string namaMataKuliah;
3      int jumlahSKS;
4  } MataKuliah;
5  // !! Struct yang akan dijadikan AoR pada record di bawahnya
6  // harus dideklarasikan terlebih dahulu agar dapat
7  // dikenali program
8
9  typedef struct{
10     string nama;
11     int npm;
12     float ipk;
13
14     MataKuliah matkul[5];
15 } Mahasiswa;
16 // Terdapat 5 mata kuliah
17 // yang ditampung dalam struct mahasiswa
```

Gambar 11 & 12. Visualisasi AoR dalam AoR dan implementasi dalam bahasa C

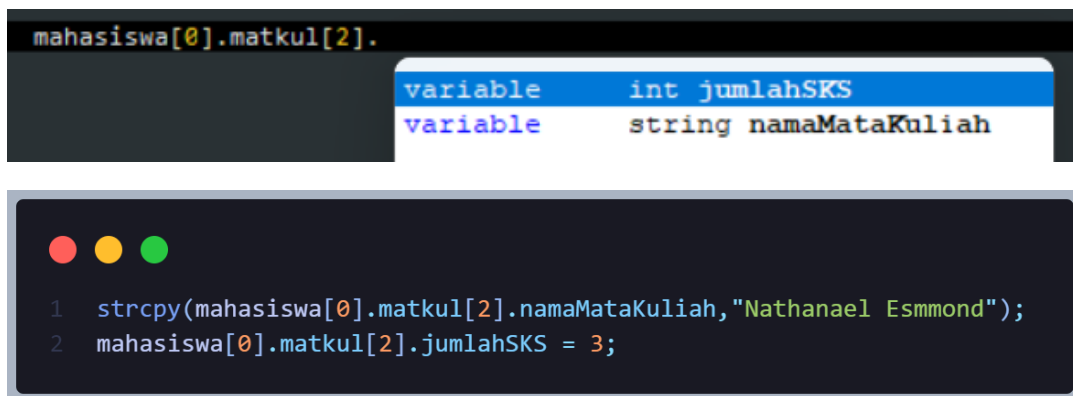


Perhatikan potongan code pada gambar 12. Pada code tersebut, terdapat Array of Record `MataKuliah matkul[5]` di dalam `Mahasiswa`. Sempelnya, dapat kita simpulkan bahwa **“Mahasiswa memiliki atribut berupa nama, NPM, IPK, dan 5 MataKuliah, di mana MataKuliah memiliki 2 atribut yaitu namaMataKuliah dan jumlahSKS.”**

Penting untuk diperhatikan juga bahwa **struct yang akan menjadi Array of Record di dalam record lainnya harus dideklarasikan terlebih dahulu agar dapat dikenali oleh program** karena program berjalan secara sekuensial (eksekusi dari line paling atas ke line paling bawah). Kemudian, struct yang akan menjadi AoR dalam record lain tinggal dideklarasikan seperti AoR pada umumnya sesuai dengan penjelasan yang ada sebelumnya.

#### b. Pengaksesan index

Pengaksesan index AoR dalam AoR tidak berbeda jauh dari pemanggilan AoR biasa. Akan tetapi, perlu diperhatikan karena **kalian akan mengakses melalui 2 buah array, maka harus menggunakan pengaksesan array 2 kali pula**. Gampangnya kalian harus ada 2 buah kurung siku `[ ]` jika kalian mengakses field AoR dalam AoR.



```
matkul[0].matkul[2].  
variable    int jumlahSKS  
variable    string namaMataKuliah
```

```
1 strcpy(mahasiswa[0].matkul[2].namaMataKuliah, "Nathanael Esmond");  
2 mahasiswa[0].matkul[2].jumlahSKS = 3;
```

Gambar 13 & 14. Mengakses index & field pada Array of Record dalam Array of Record

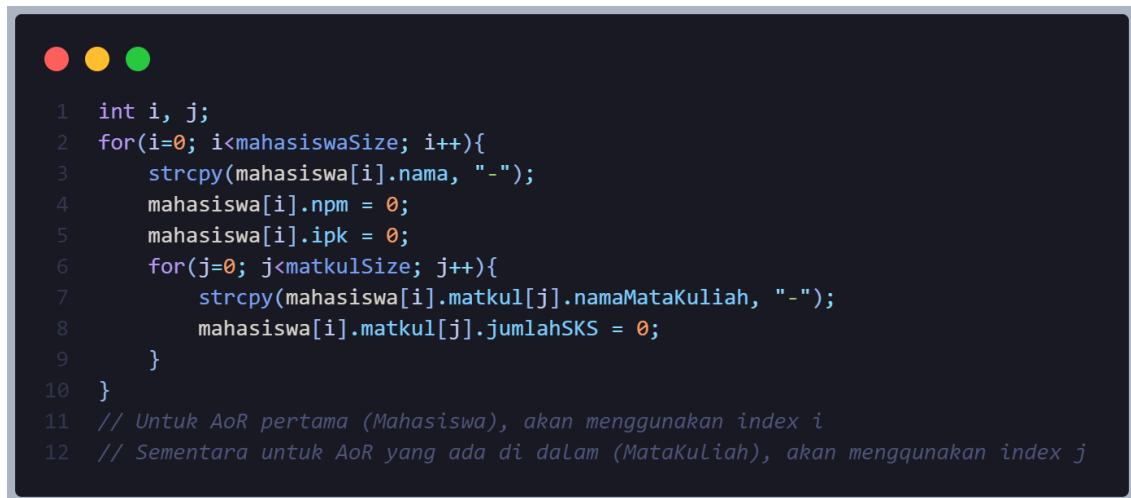
Pada gambar di atas, kita dapat melihat bentuk pengaksesan Array of Record dalam Array of Record. Penting untuk diperhatikan, pengaksesan atribut di dalam record selalu diikuti dengan tanda titik (`.`). Setelah mengakses Array of Record pertama dan memberikan kurung siku `[ ]` untuk mengakses index, harus dilanjutkan dengan tanda titik lagi untuk bisa mengakses Array of Record kedua.

### c. Pengisian dan Pengubahan Data

Pengisian dan Pengubahan data di sini sama saja seperti pengisian dan pengubahan data pada Array of Record biasa. Kalian dapat membaca dan memahami bagian pengaksesan index di atas untuk detail lebih jelas. Gambar 14 merupakan contoh bagaimana kalian bisa mengisi nilai dari field Array of Record dalam Array of Record.

### d. Perulangan untuk AoR dalam AoR

Perlu dipahami bahwa perulangan digunakan untuk mengakses index dari array. Karena di konsep AoR dalam AoR akan ada dua array yang kalian akses, maka kita akan menggunakan **nested loop** untuk mengakses field dari AoR dari sebuah AoR. Konsep ini seringkali digunakan untuk inisialisasi dan pencarian di konsep AoR dalam AoR.



```
1  int i, j;
2  for(i=0; i<mahasiswaSize; i++){
3      strcpy(mahasiswa[i].nama, "-");
4      mahasiswa[i].npm = 0;
5      mahasiswa[i].ipk = 0;
6      for(j=0; j<matkulSize; j++){
7          strcpy(mahasiswa[i].matkul[j].namaMataKuliah, "-");
8          mahasiswa[i].matkul[j].jumlahSKS = 0;
9      }
10 }
11 // Untuk AoR pertama (Mahasiswa), akan menggunakan index i
12 // Sementara untuk AoR yang ada di dalam (MataKuliah), akan menggunakan index j
```

Gambar 15. Penerapan nested loop dalam konsep AoR dalam AoR

## VI. Penerapan Fungsi dan Prosedur pada Array of Record

Pada modul-modul sebelumnya (array dan record) tentu kalian sudah mempelajari bagaimana cara menerapkan fungsi serta prosedur dalam array maupun record. Penerapan fungsi dan Prosedur dalam modul kali ini tidak akan jauh berbeda. Kalian tinggal mengkombinasikan kedua hal tersebut dan menggunakannya pada program kalian. Misalnya, berikut ini adalah contoh bagaimana kalian akan menginisialisasi (set semua nilai pada AoR menjadi kosong) dalam Array of Record.



```

1 void inisialisasi(Mahasiswa mahasiswa[]);
2
3 int main(int argc, char *argv[]) {
4     Mahasiswa mahasiswa[5];
5
6     inisialisasi(mahasiswa);
7
8     return 0;
9 }
10
11 void inisialisasi(Mahasiswa mahasiswa[]){
12     int i, j;
13     for(i=0; i<mahasiswaSize; i++){
14         strcpy(mahasiswa[i].nama, "-");
15         mahasiswa[i].npm = 0;
16         mahasiswa[i].ipk = 0;
17         for(j=0; j<matkulSize; j++){
18             strcpy(mahasiswa[i].matkul[j].namaMataKuliah, "-");
19             mahasiswa[i].matkul[j].jumlahSKS = 0;
20         }
21     }
22 }

```

Gambar 16. Contoh penerapan prosedur dalam AoR

Kalian juga akan mempelajari contoh penerapan prosedur dan fungsi dalam AoR lainnya di modul ini.

## VII. Kesimpulan

Gimana? Array of Record mudah dan menyenangkan bukan ??

Balik lagi seperti yang sudah dibilang di awal, Array of Record itu cuma *array dari tipe data bentukan (record)*, gabungan dari kedua modul sebelumnya. Pahami materi array dan record secara matang, niscaya kalian akan dimudahkan dalam mengerjakan soal UGD nantinya. Semangat !!

## GUIDED

Di guided ini, kita akan membuat sebuah sistem pencatatan sederhana yang dapat menampung data Mahasiswa berupa nama, NPM, IPK dan mata kuliah yang diambil. Untuk selengkapnya, kalian dapat mengikuti sesuai dengan code di bawah. Aku juga mengajak kalian untuk memodifikasi dan mengeksplor code di bawah (selama tidak mempengaruhi fungsionalitas program. Selamat mengerjakan!

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <stdbool.h>
5
6  #define mahasiswaSize 10
7  #define matkulSize 7
8
9  typedef char string[100];
10
11 typedef struct{
12     string namaMataKuliah;
13     int jumlahSKS;
14 } MataKuliah;
15
16 typedef struct{
17     string nama;
18     int npm;
19     float ipk;
20
21     MataKuliah matkul[matkulSize];
22 } Mahasiswa;
23
24 void inisialisasi(Mahasiswa mahasiswa[]);
25 void showMenu();
26 int getEmptyIndexMahasiswa(Mahasiswa mahasiswa[]);
27 bool isEmptyArrayMahasiswa(Mahasiswa mahasiswa[]);
28 Mahasiswa createMahasiswa(string namaMahasiswa, int npm, float ipk, MataKuliah matkul[]);
29 void showDataMahasiswa(Mahasiswa mahasiswa[]);
30 void deleteOneMahasiswa(Mahasiswa mahasiswa[], int index);
31 int findMahasiswaByNPM(Mahasiswa mahasiswa[], int npm);
32 int findMataKuliahByNamaMatkul(Mahasiswa mahasiswa[], int index, string namaMatkul);
```

```

34 int main(int argc, char *argv[]) {
35     Mahasiswa mahasiswa[mahasiswaSize];
36     MataKuliah matkul[matkulSize];
37
38     int i, index, indexMatkul, menu;
39     int npm, jumlahSKS;
40     string namaMahasiswa, namaMataKuliah;
41     float ipk;
42
43     inisialisasi(mahasiswa);
44
45     do{
46         system("cls");
47         showMenu();
48         printf("\n\t>>> "); scanf("%d", &menu);
49
50         switch (menu){
51             case 1:
52                 printf("\n\t\t=== [ Create Data Mahasiswa ]===\n");
53
54                 index = getEmptyIndexMahasiswa(mahasiswa);
55                 if(index != -1){
56                     printf("\n\tNama Mahasiswa : "); fflush(stdin); gets(namaMahasiswa);
57                     printf("\tNPM          : "); scanf("%d", &npm);
58                     printf("\tIPK           : "); scanf("%f", &ipk);
59
60                     printf("\n\t\t== Input Mata Kuliah ==");
61                     for(i=0; i<matkulSize; i++){
62                         printf("\n\tNama Mata Kuliah Ke-%d : ", i+1); fflush(stdin); gets(namaMataKuliah);
63                         printf("\tJumlah SKS : "); scanf("%d", &jumlahSKS);
64
65                         strcpy(matkul[i].namaMataKuliah, namaMataKuliah);
66                         matkul[i].jumlahSKS = jumlahSKS;
67                     }
68
69                     mahasiswa[index] = createMahasiswa(namaMahasiswa, npm, ipk, matkul);
70                     printf("\n\t[+] Berhasil Create Mahasiswa [+]");
71                 } else {
72                     printf("\n\t[!] Data Mahasiswa Sudah Penuh [!]);
73                 }
74                 break;
75
76             case 2:
77                 printf("\n\t\t=== [ Read Data Mahasiswa ]===\n");
78                 if(isEmptyArrayMahasiswa(mahasiswa) == false){
79                     showDataMahasiswa(mahasiswa);
80                 } else {
81                     printf("\n\t[!] Data Mahasiswa Kosong [!]);
82                 }
83                 break;

```

```

85  case 3:
86      printf("\n\t\t=== [ Update Data Mahasiswa ]===\n");
87
88      if(isEmptyArrayMahasiswa(mahasiswa) == false){
89          printf("\n\tMasukkan NPM Mahasiswa : "); scanf("%d", &npm);
90          index = findMahasiswaByNPM(mahasiswa, npm);
91          if(index != -1){
92              printf("\n\tNama Mahasiswa : "); fflush(stdin); gets(namaMahasiswa);
93              printf("\tIPK : "); scanf("%f", &ipk);
94
95              printf("\n\t\t== Input Mata Kuliah ==");
96              for(i=0; i<matkulSize; i++){
97                  printf("\n\tNama Mata Kuliah Ke-%d : ", i+1); fflush(stdin); gets(namaMataKuliah);
98                  printf("\tJumlah SKS : "); scanf("%d", &jumlahSKS);
99
100                 strcpy(matkul[i].namaMataKuliah, namaMataKuliah);
101                 matkul[i].jumlahSKS = jumlahSKS;
102             }
103
104             mahasiswa[index] = createMahasiswa(namaMahasiswa, mahasiswa[index].npm, ipk, matkul);
105             printf("\n\t[+] Berhasil Update Mahasiswa [+]");
106         } else {
107             printf("\n\t[!] Data Mahasiswa Tidak Ditemukan [!]\n");
108         }
109     } else {
110         printf("\n\t[!] Data Mahasiswa Kosong [!]\n");
111     }
112     break;
113
114     case 4:
115         printf("\n\t\t=== [ Delete Data Mahasiswa ]===\n");
116         if(isEmptyArrayMahasiswa(mahasiswa) == false){
117             printf("\n\tMasukkan NPM Mahasiswa : "); scanf("%d", &npm);
118             index = findMahasiswaByNPM(mahasiswa, npm);
119             if(index != -1){
120                 deleteOneMahasiswa(mahasiswa, index);
121                 printf("\n\t[+] Berhasil Delete Mahasiswa [+]");
122             } else {
123                 printf("\n\t[!] Data Mahasiswa Tidak Ditemukan [!]\n");
124             }
125         } else {
126             printf("\n\t[!] Data Mahasiswa Kosong [!]\n");
127         }
128         break;
129
130     case 0:
131         printf("\n\t\t=== [ NAMA PRAKTIKAN ]===");
132         printf("\n\t\t=== [ NPM - KELAS (A / B / C / D / E)]");
133         // Silahkan diganti sesuai nama, NPM, dan kelas kalian
134         break;
135
136     default:
137         printf("\n\t[!] Menu Tidak Tersedia [!]\n");
138         break;
139     }
140     getch();
141 } while(menu != 0);
142
143 return 0;
144 }

```

```

146 void inisialisasi(Mahasiswa mahasiswa[]){
147     int i, j;
148     for(i=0; i<mahasiswaSize; i++){
149         strcpy(mahasiswa[i].nama, "-");
150         mahasiswa[i].npm = 0;
151         mahasiswa[i].ipk = 0;
152         for(j=0; j<matkulSize; j++){
153             strcpy(mahasiswa[i].matkul[j].namaMataKuliah, "-");
154             mahasiswa[i].matkul[j].jumlahSKS = 0;
155         }
156     }
157     // Untuk AoR pertama (Mahasiswa), akan menggunakan index i
158     // Sementara untuk AoR yang ada di dalam (MataKuliah), akan menggunakan index j
159 }
160
161 void showMenu(){
162     printf("\n\t\t===[ GUIDED ARRAY OF RECORD ]===\n");
163     printf("\n\t[1]. Create Data Mahasiswa");
164     printf("\n\t[2]. Read Data Mahasiswa");
165     printf("\n\t[3]. Update Data Mahasiswa");
166     printf("\n\t[4]. Delete Data Mahasiswa");
167     printf("\n\n\t[5]. Edit Mata Kuliah");
168     printf("\n\t[6]. Delete Mata Kuliah");
169     printf("\n\n\t[0]. Exit");
170 }
171
172 int getEmptyIndexMahasiswa(Mahasiswa mahasiswa[]){
173     int i;
174     for(i=0; i<mahasiswaSize; i++){
175         if(mahasiswa[i].npm == 0){
176             return i;
177         }
178     }
179     return -1; // Artinya array sudah penuh, tidak ada index kosong
180 }
181
182 bool isEmptyArrayMahasiswa(Mahasiswa mahasiswa[]){
183     int i;
184
185     for(i=0; i<mahasiswaSize; i++){
186         if(mahasiswa[i].npm != 0){
187             return false;
188         }
189     }
190     return true;
191 }

```

```

193 Mahasiswa createMahasiswa(string namaMahasiswa, int npm, float ipk, MataKuliah matkul[]){
194     int i;
195     Mahasiswa m;
196
197     strcpy(m.nama, namaMahasiswa);
198     m.npm = npm;
199     m.ipk = ipk;
200
201     for(i=0; i<matkulSize; i++){
202         strcpy(m.matkul[i].namaMataKuliah, matkul[i].namaMataKuliah);
203         m.matkul[i].jumlahSKS = matkul[i].jumlahSKS;
204     }
205
206     return m;
207 }
208
209 void showDataMahasiswa(Mahasiswa mahasiswa[]){
210     int i, j;
211     int urutanMahasiswa = 1, urutanMatkul = 1;
212     for(i=0; i<mahasiswaSize; i++){
213         if(mahasiswa[i].npm != 0){ // IF ini digunakan agar perulangan hanya menampilkan data mahasiswa yang tidak kosong
214             printf("\n\tMahasiswa [%d]", urutanMahasiswa++);
215             printf("\n\tNama      : %s", mahasiswa[i].nama);
216             printf("\n\tNPM       : %d", mahasiswa[i].npm);
217             printf("\n\tIPK        : %.2f\n", mahasiswa[i].ipk);
218             printf("\n\t\t=== Mata Kuliah ===");
219             urutanMatkul = 1;
220             for(j=0; j<matkulSize; j++){
221                 if(strcmpi(mahasiswa[i].matkul[j].namaMataKuliah, "-")!=0){
222                     printf("\n\t\tMata Kuliah [%d]", urutanMatkul++);
223                     printf("\n\t\tNama Mata Kuliah : %s", mahasiswa[i].matkul[j].namaMataKuliah);
224                     printf("\n\t\tJumlah SKS      : %d\n", mahasiswa[i].matkul[j].jumlahSKS);
225                 }
226             }
227             printf("\n");
228         }
229     }
230 }
231 }
232
233 int findMahasiswaByNPM(Mahasiswa mahasiswa[], int npm){
234     int i;
235     for(i=0; i<mahasiswaSize; i++){
236         if(mahasiswa[i].npm == npm){
237             return i;
238         }
239     }
240     return -1; // Artinya tidak ditemukan mahasiswa dengan NPM tersebut
241 }
242
243 int findMataKuliahByNamaMatkul(Mahasiswa mahasiswa[], int index, string namaMatkul){
244     int i;
245     for(i=0; i<matkulSize; i++){
246         if(strcmpi(mahasiswa[index].matkul[i].namaMataKuliah, namaMatkul)==0){
247             return i;
248         }
249     }
250     return -1;
251 }
252
253 void deleteOneMahasiswa(Mahasiswa mahasiswa[], int index){
254     int i;
255
256     strcpy(mahasiswa[index].nama, "-");
257     mahasiswa[index].npm = 0;
258     mahasiswa[index].ipk = 0;
259
260     for(i=0; i<matkulSize; i++){
261         strcpy(mahasiswa[index].matkul[i].namaMataKuliah, "-");
262         mahasiswa[index].matkul[i].jumlahSKS = 0;
263     }
264 }

```



## TUGAS GUIDED

Buatlah dua menu tambahan seperti di bawah ini.

```
===[ GUIDED ARRAY OF RECORD ]===

[1]. Create Data Mahasiswa
[2]. Read Data Mahasiswa
[3]. Update Data Mahasiswa
[4]. Delete Data Mahasiswa

[5]. Edit Mata Kuliah
[6]. Delete Mata Kuliah

[0]. Exit
>>>
```

### 1. Edit Mata Kuliah

Di menu ini, user dapat mencari mata kuliah dari mahasiswa tertentu dan mengubah nama mata kuliah dan jumlah SKS. User harus menginputkan NPM mahasiswa dan nama mata kuliah yang dihapus dari daftar mata kuliah mahasiswa yang bersangkutan. Terdapat 2 error handling, yaitu jika NPM mahasiswa tidak ditemukan dan nama mata kuliah tidak ditemukan.

```
[0]. Exit
>>> 5

===[ Update Mata Kuliah ]===

Masukkan NPM Mahasiswa : 87654321

[!] Data Mahasiswa Tidak Ditemukan [!]
```

```
[0]. Exit
>>> 5

===[ Update Mata Kuliah ]===

Masukkan NPM Mahasiswa : 12345678
Masukkan Nama Mata Kuliah : PBO

[!] Mata Kuliah Tidak Ditemukan Pada Mahasiswa Dhiyaz [!]
```

```

===[ Update Mata Kuliah ]===

Masukkan NPM Mahasiswa : 12345678

Masukkan Nama Mata Kuliah : Daspro

== Update ==

Masukkan Nama Mata Kuliah : Masyarakat Digital

Jumlah SKS : 3

[+] Berhasil Update Mata Kuliah [+]

```

## 2. Delete Mata Kuliah

Dengan error handling yang sama dengan update mata kuliah, user dapat menghapus salah satu mata kuliah pada mahasiswa tertentu. Nantinya, case 2 hanya menampilkan data mata kuliah yang memiliki nilai saja.

```

>>> 6
===[ Delete Mata Kuliah ]===

Masukkan NPM Mahasiswa : 12345678
Masukkan Nama Mata Kuliah : Masyarakat Digital
[+] Berhasil Delete Mata Kuliah [+]

Nama      : Dhiaz
NPM       : 12345678
IPK       : 4.00

=== Mata Kuliah ===
Mata Kuliah [1]
Nama Mata Kuliah : Dasar Pemrograman
Jumlah SKS       : 4

Mata Kuliah [2]
Nama Mata Kuliah : Matematika Diskrit
Jumlah SKS       : 3

Mata Kuliah [3]
Nama Mata Kuliah : Aljabar Linear
Jumlah SKS       : 3

Mata Kuliah [4]
Nama Mata Kuliah : PKN
Jumlah SKS       : 2

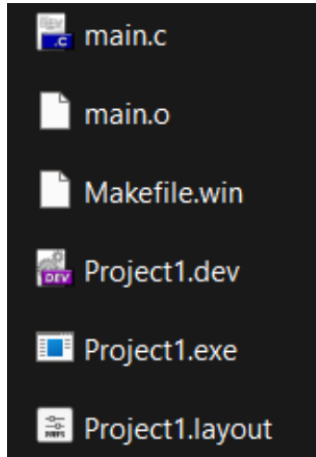
Mata Kuliah [5]
Nama Mata Kuliah : Agama
Jumlah SKS       : 2

Mata Kuliah [6]
Nama Mata Kuliah : Dasar Keamanan Sistem
Jumlah SKS       : 3

```

## KETENTUAN PENGUMPULAN GUIDED

1. **Unguided tidak akan berbeda jauh dari Guided.** Berbahagialah wahai mereka yang mempelajari Modul & Guided dengan baik.
2. Comment pada guided bersifat opsional (tidak wajib ditulis).
3. **Kumpulkan seluruh file seperti di bawah**



4. **Guided dan Tugas Guided berada dalam satu project yang sama.** Tugas guided hanya modifikasi dari guided yang tersedia.
5. File di **zip** dengan format penamaan **GD12\_X\_YYYYY.zip**  
**X = Kelas**  
**Y = 5 digit terakhir NPM**
6. **Jangan sampai salah format.** Kesalahan format penamaan file -10.
7. Ini sudah modul terakhir, **tidak ada lagi toleransi untuk plagiasi.**
8. Jangan ragu untuk bertanya ke Asisten Dosen, khususnya ke pemegang modul. Kalian bisa menghubungi saya via Teams (Pasha Rakha Paruntung, 220711976@students.uajy.ac.id).

\*Hint Bonus

Pelajari mengenai fungsi/prosedur rekursif bagi kalian yang ingin mengambil bonus.