MODUL 12

ARRAY OF RECORD

MODUL 12

ARRAY OF RECORD

I. Tujuan

1. Praktikan dapat memahami konsep Array of Record dalam bahasa pemrograman C.
2. Praktikan dapat mengimplemantasikan Array of Record dalam kasus ril.

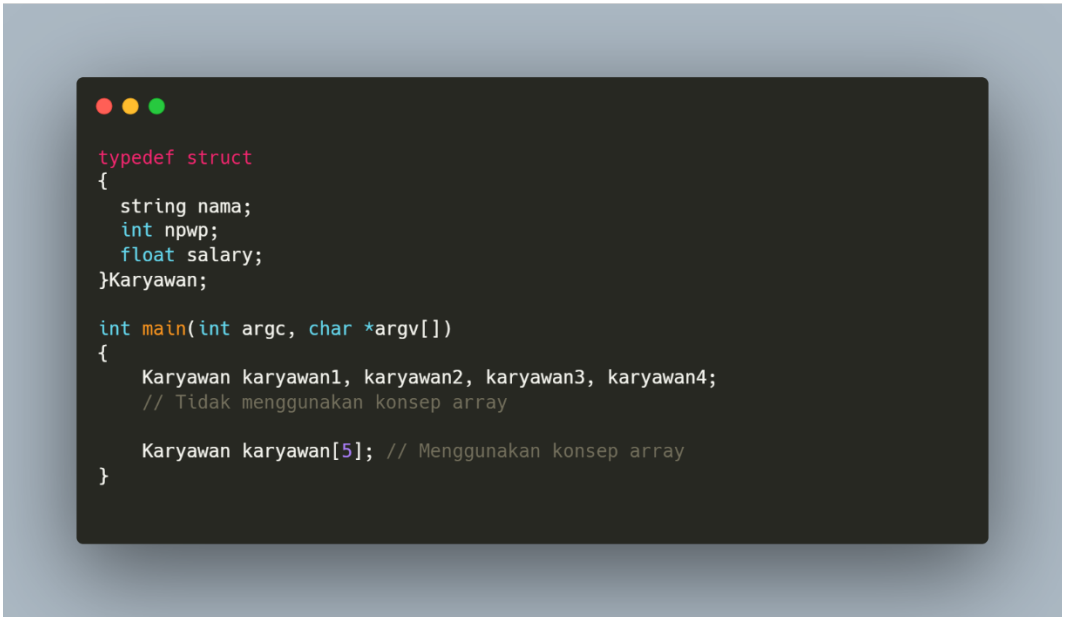
II. Pengantar

Array adalah struktur data yang dapat menyimpan kumpulan data dengan jenis yang sama secara berurutan dalam memori komputer sehingga pengaksesan elemen dalam array dapat menggunakan index.

Record adalah tipe data bentukan yang dapat menyimpan beragam jenis tipe data yang memiliki hubungan menjadi satu kesatuan.

III. Array of Record

Penggabungan prinsip array dengan prinsip record adalah metode pemrograman yang efisien karena dengan array kita dapat memiliki n kumpulan data yang sama dan juga dapat menyimpan beragam jenis data.



```
typedef struct
{
  string nama;
  int npwp;
  float salary;
}Karyawan;

int main(int argc, char *argv[])
{
    Karyawan karyawan1, karyawan2, karyawan3, karyawan4;
    // Tidak menggunakan konsep array

    Karyawan karyawan[5]; // Menggunakan konsep array
}
```

Gambar 1. Contoh penerapan Array of Record

IV. Penerapan

A. Deklarasi

Deklarasi Aor dapat dilakukan dengan syntax seperti **tipeData namaVariabel[ukuran];**

```c
#define MAX 5
typedef char string[100];

typedef struct
{
    string nama;
    int npwp;
    float salary;
}Karyawan;


int main(int argc, char *argv[])
{
    Karyawan karyawan1, karyawan2, karyawan3, karyawan4;
    // Tidak menggunakan konsep array


    Karyawan karyawan[MAX];
    // Menggunakan konsep array

}
```

Gambar 2. Mendeklarasikan array of record bertipe karyawan sesuai dengan MAX

B. Inisialisasi

Inisialisasi nilai pada array of record bisa dilakukan dengan dua cara.

Pertama menggunakan inisialisasi secara langsung seperti gambar berikut

```c
#define MAX 5

typedef struct
{
    string nama;
    int npwp;
    float salary;
}Karyawan;


int main(int argc, char *argv[])
{
    Karyawan karyawan[MAX] = {
      {"Vergie", 111, 5400},
      {"Ferrincia", 222, 5200},
      {"Farel", 333, 5000},
      {"Axel", 444, 5100},
      {"Zelika", 555, 5150}
    };

}
```

Gambar 3. Contoh inisalisasi secara langsung.

Cara kedua bisa memanfaatkaan loop seperti berikut :

```c
#define MAX 5
typedef char string[100];

typedef struct
{
    string nama;
    int npwp;
    float salary;
}Karyawan;


int main(int argc, char *argv[])
{
    int i;
    Karyawan temp[MAX];

    for(i = 0; i < MAX; i++)
    {
        printf("\nMasukan Nama  : "); fflush(stdin); gets(temp[i].nama);
        printf("\nMasukan NPWP  : "); scanf("%d", &temp[i].npwp);
        printf("\nMasukan Gaji  : "); scanf("%d", &temp[i].salary);
    }

}
```

Gambar 4. Contoh pemberian nilai menggunakan loop.

C. Indexing

Pengaksesan nilai pada array of record sama seperti array pada umumnya menggunakan index yang dimulai dari 0. Kelebihan struktur data array adalah random akses, dimana kita bisa akses sembarang data dari rentang 0 sampai (jumlah data – 1). Misal kita akses index-1, seperti ini karyawan[1], maka kita merujuk pada struct Karyawan yang disimpan di index 1, untuk mengakses fieldnya diperlukan dot sama seperti record biasa.

```
#define MAX 5

typedef struct
{
    string nama;
    int npwp;
    float salary;
}Karyawan;


int main(int argc, char *argv[])
{
    Karyawan karyawan[MAX] = {
      {"Vergie", 111, 5400},
      {"Ferrincia", 222, 5200},
      {"Farel", 333, 5000},
      {"Axel", 444, 5100},
      {"Zelika", 555, 5150}
    };

    printf("%s - %d - %.2f",
    karyawan[1].nama, karyawan[1].npwp, karyawan[1].salary);
    // Output (Ferrincia - 222 - 5200)

    // Bisa juga disimpan seperti karyawan k = karyawan[1]
    // Ingat karyawan adalah array of Karyawan sehingga indexing akan merujuk
      ke struct karyawan!
}
```

```
int main(int argc, char *argv[]) {

    Karyawan karyawan[MAX] = {
      {"Vergie", 111, 5400},
      {"Ferrincia", 222, 5200},
      {"Farel", 333, 5000},
      {"Axel", 444, 5100},
      {"Zelika", 555, 5150}
    };

    Karyawan k = karyawan[1];

    printf("%s - %d - %.2f",
    k.nama, k.npwp, k.salary);

    return 0;
}
```

```
C:\Users\LENOVO\One
Ferrincia - 222 - 5200.00
--------------------------------
Process exited after 0.0577 seconds with ret
urn value 0
Press any key to continue . . .
```

Syntax : namaArray[index].field

V. Array of Record dalam Array of Record

Bagaimana jika sekarang ditambah data manager dimana manager ingin menyimpan data-data karyawan yang di bawah asuhannya.

A. Implementasi

```
typedef struct
{
    string nama;
    int npwp;
    float salary;
}Karyawan;

typedef struct
{
    string nama;
    Karyawan dataAnakBuah[3];
}Manager;
```

Gambar 5. Contoh pembuatan struct Manager yang berisi array of record Karyawan

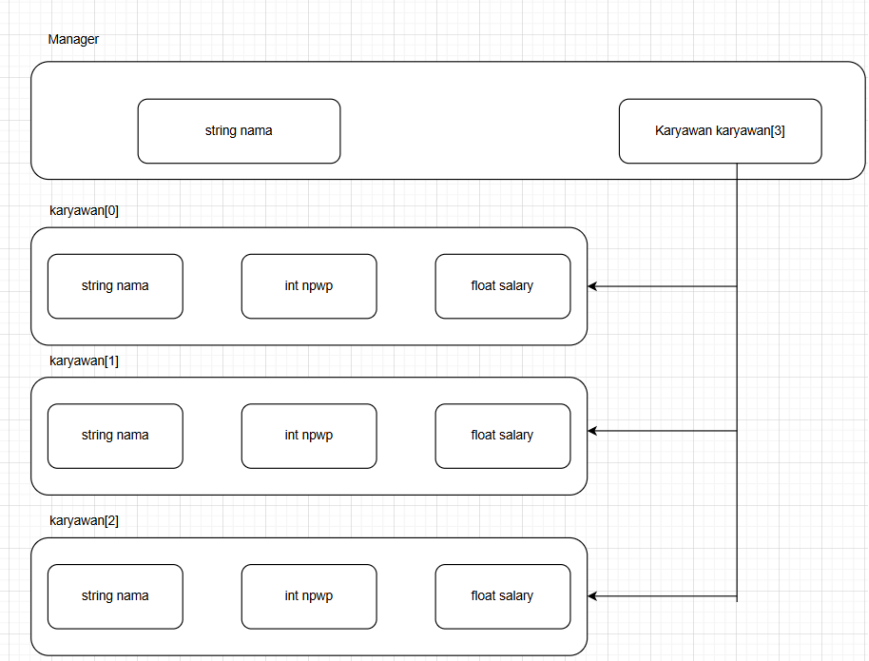
B. Indexing

```
int main(int argc, char *argv[]) {

    Manager manager[MAX];
    int i, j;

    for(i = 0; i < MAX; i++)
    {
        printf("Masukan Nama Manager : ");
        fflush(stdin); gets(manager[i].nama);

        for(j = 0; j < 3; j++)
        {
            printf("Masukan Nama Karyawan - %d : ", i+1);
            fflush(stdin); gets(manager[i].dataAnakBuah[j].nama);


            printf("Masukan NPWP Karyawan - %d : ", i+1);
            scanf("%d", &manager[i].dataAnakBuah[j].npwp);


            printf("Masukan Gaji Karyawan - %d : ", i+1);
            scanf("%f", &manager[i].dataAnakBuah[j].salary);
        }
    }


    return 0;
}
```

Gambar 6. Penggunaan loop untuk pemberian nilai

Untuk array of record dalam array of record kita menggunakan nested loop, dimana outer loop untuk array pertama dan inner loop untuk array of record di dalamnya.

Syntax : arrayLuar[indexLuar].arrayDalam[indexDalam].field



Gambar 7. Visualisasi array of record dalam array of record

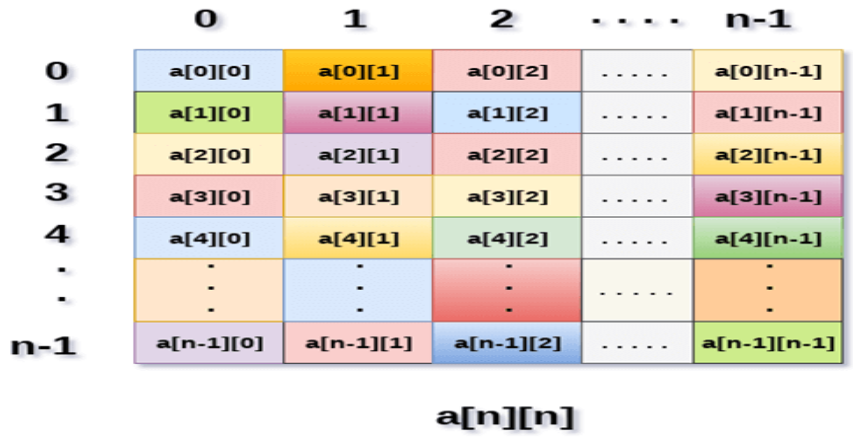
VI. Array of Record 2 Dimensi

Sama seperti array yang bisa dibuat menjadi dua dimensi, array of record yang merupakan tipe data buatan berupa array juga bisa dibuat menjadi dua dimensi.

A. Implementasi

```
typedef struct
{
    string nama;
    int npwp;
    float salary;
}Karyawan;


int main(int argc, char *argv[]) {

    Karyawan karyawan[2][2];


    return 0;
}
```

Gambar 8. Deklarasi Array of Record 2 Dimensi



(Pasti tidak asing dengan gambar ini). **BENAR** array 2d bisa direpresentasikan sebagai matriks.

```
typedef struct
{
    string nama;
    int npwp;
    float salary;
}Karyawan;


int main(int argc, char *argv[]) {

    Karyawan karyawan[2][2];
    int i, j;

    for(i = 0; i < 2; i++)
    {
        for(j = 0; j < 2; j++)
        {
            printf("Masukan Nama Karyawan : ");
            fflush(stdin); gets(karyawan[i][j].nama);

            printf("Masukan NPWP Karyawan : ");
            scanf("%d", &karyawan[i][j].npwp);

            printf("Masukan Gaji Karyawan : ");
            scanf("%f", &karyawan[i][j].salary);
        }
    }


    return 0;
}
```

Gambar 9. Indexing dan Pemberian nilai pada AoR 2D

Syntax : **namaArray[indexBaris][indexKolom].field**

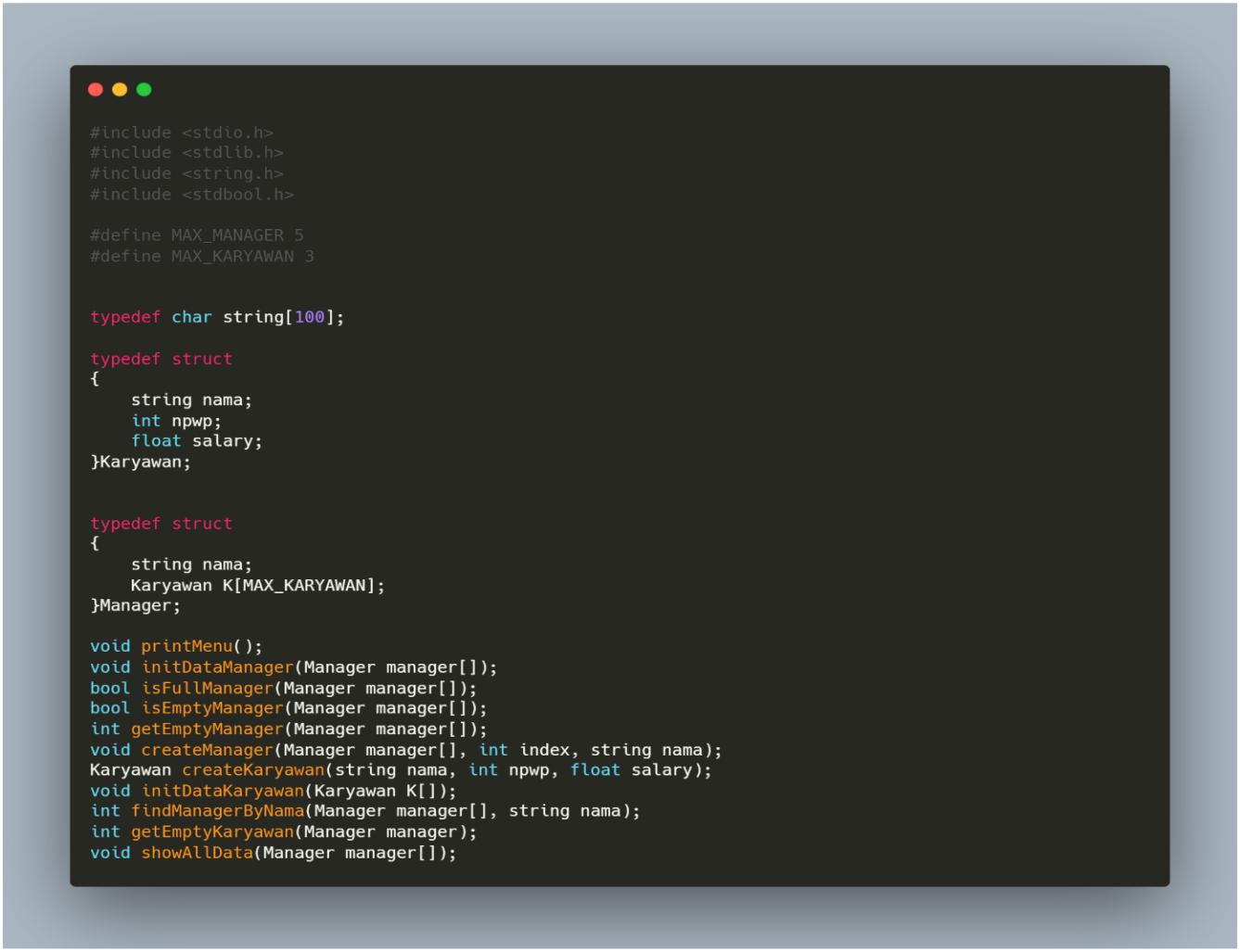
VII. Kesimpulan

Pada akhrinya, array of record adalah bagaimana sebuah tipe data buatan dibuat menjadi sebuah array. Untuk memahami materi ini diperlukan pemahaman terhadap struktur data array dan penggunaan record.

VIII. Referensi

1. https://www.geeksforgeeks.org/c-array-of-structure/
2. https://www.geeksforgeeks.org/how-to-iterate-through-array-of-structs-in-c/?ref=ml_lbp
3. https://www.geeksforgeeks.org/searching-in-array-of-struct-in-c/?ref=ml_lbp
4. https://www.tutorialspoint.com/cprogramming/c_arrays_of_structures.htm
5. https://www.w3schools.com/c/c_arrays_multi.php

IX. GUIDED



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

#define MAX_MANAGER 5
#define MAX_KARYAWAN 3


typedef char string[100];

typedef struct
{
    string nama;
    int npwp;
    float salary;
}Karyawan;


typedef struct
{
    string nama;
    Karyawan K[MAX_KARYAWAN];
}Manager;

void printMenu();
void initDataManager(Manager manager[]);
bool isFullManager(Manager manager[]);
bool isEmptyManager(Manager manager[]);
int getEmptyManager(Manager manager[]);
void createManager(Manager manager[], int index, string nama);
Karyawan createKaryawan(string nama, int npwp, float salary);
void initDataKaryawan(Karyawan K[]);
int findManagerByNama(Manager manager[], string nama);
int getEmptyKaryawan(Manager manager);
void showAllData(Manager manager[]);
```

```c
int main(int argc, char *argv[]) {
    Manager manager[MAX_MANAGER];
    Karyawan temp[MAX_KARYAWAN];
    int menu, indexM, indexK, i;
    string nama, namaK;
    int npwp;
    float gaji;


    initDataManager(manager);

    do
    {
        system("cls");
        printf("\n\t === [ UGD Array of Record ] ===\n");
        printMenu();
        printf("\n>>> ");
        scanf("%d", &menu);

        system("cls");

        switch(menu)
        {
            case 1:
                if(isFullManager(manager))
                {
                    printf("\n\t [!] Data Manager Sudah Penuh [!]");
                }
                else
                {
                    indexM = getEmptyManager(manager);
                    if(indexM != -1)
                    {
                        do
                        {
                            printf("Masukan Nama Manager    : "); fflush(stdin); gets(nama);
                            if(strcmp(nama, "-") == 0)
                            {
                                printf("\n [!] Invalid [!]");
                            }
                        }while(strcmp(nama, "-") == 0);

                        createManager(manager, indexM, nama);
                        printf("\n\t [+] Berhasil Menambahkan Data Manager [+]");
                    }
                }
            break;

            case 2:
                if(isEmptyManager(manager))
                {
                    printf("\n\t [!] Belum Ada Data Manager [!]");
                }
                else
                {
                    printf("\nMasukan Nama Manager  : "); fflush(stdin); gets(nama);
                    indexM = findManagerByNama(manager, nama);

                    if(indexM != -1)
                    {
                        do
                        {
                            printf("Masukan Nama Manager    : "); fflush(stdin); gets(nama);
                            if(strcmp(nama, "-") == 0)
                            {
                                printf("\n [!] Invalid [!]");
                            }
                        }while(strcmp(nama, "-") == 0);

                        createManager(manager, indexM, nama);
                        printf("\n\t [+] Berhasil Mengubah Data Manager [+]");
                    }
                    else
                    {
                        printf("\n\t [!] Manager Tidak Ditemukan [!]");
                    }
                }
            break;
```

```c
            case 3:
                if(isEmptyManager(manager))
                {
                    printf("\n\t [!] Belum Ada Data Manager [!]");
                }
                else
                {
                    printf("\nMasukan Nama Manager  : "); fflush(stdin); gets(nama);
                    indexM = findManagerByNama(manager, nama);

                    if(indexM != -1)
                    {
                        indexK = getEmptyKaryawan(manager[indexM]);

                        if(indexK != -1)
                        {
                            printf("\n\t Masukan Nama Karyawan  : "); fflush(stdin); gets(namaK);
                            printf("\n\t Masukan NPWP Karyawan  : "); scanf("%d", &npwp);
                            printf("\n\t Masukan Gaji Karyawan  : "); scanf("%f", &gaji);

                            manager[indexM].K[indexK] = createKaryawan(namaK, npwp, gaji);

                            printf("\n\t [+] Manager %s memiliki karyawan baru %s [+]",
                            manager[indexM].nama, manager[indexM].K[indexK].nama);
                        }
                        else
                        {
                            printf("\n\t [!] Manager %s sudah memiliki cukup karyawan [!]\n",
                            manager[indexM].nama);
                        }
                    }
                    else
                    {
                        printf("\n\t [!] Manager Tidak Ditemukan [!]");
                    }
                }

            break;

            case 4:
                if(isEmptyManager(manager))
                {
                    printf("\n\t [!] Belum Ada Data Manager [!]");
                }
                else
                {
                    printf("\nMasukan Nama Manager  : "); fflush(stdin); gets(nama);
                    indexM = findManagerByNama(manager, nama);

                    if(indexM != -1)
                    {
                        createManager(manager, indexM, "-");
                        for(i = 0; i < MAX_KARYAWAN; i++)
                        {
                            manager[indexM].K[i] = createKaryawan("-", 0, 0);
                        }

                        printf("\n\t [+] Berhasil Menghapus Data Manager [+]");
                    }
                    else
                    {
                        printf("\n\t [!] Manager Tidak Ditemukan [!]");
                    }
                }
            break;

            case 5:
                if(isEmptyManager(manager))
                {
                    printf("\n\t [!] Belum Ada Data Manager [!]");
                }
                else
                {
                    showAllData(manager);
                }

            break;


            case 0:
                printf("\n\t NAMA PRAKTIKAN - KELAS PRAKTIKAN - NPM PRAKTIKAN");
                printf("\n\t SIAP MENAMATKAN DASPRO!");
            break;

            default:
                printf("\n [!] INVALID MENU [!]\n");
            break;
        }
    getch();
    }while(menu != 0);


    return 0;
}
```

```c
void printMenu()
{
    printf("\n[1]. Input Data Manager");
    printf("\n[2]. Update Data Manager");
    printf("\n[3]. Input Data Karyawan");
    printf("\n[4]. Delete Data Manager");
    printf("\n[5]. Show All Data");
    printf("\n[0]. Exit");
}

void initDataManager(Manager manager[])
{
    int i, j;
    for(i = 0; i < MAX_MANAGER; i++)
    {
        strcpy(manager[i].nama, "-");
        for(j = 0; j < MAX_KARYAWAN; j++)
        {
            strcpy(manager[i].K[j].nama, "-");
            manager[i].K[j].npwp = 0;
            manager[i].K[j].salary = 0.0;
        }
    }
}

void initDataKaryawan(Karyawan K[])
{
    int i;
    for(i = 0; i < MAX_KARYAWAN; i++)
    {
        strcpy(K[i].nama, "-");
        K[i].npwp = 0;
        K[i].salary = 0.0;
    }
}

bool isFullManager(Manager manager[])
{
    int i;
    for(i = 0; i < MAX_MANAGER; i++)
    {
        if(strcmp(manager[i].nama, "-") == 0) // Masih ada struct kosong
        {
            return false;
        }
    }

    return true;
}

bool isEmptyManager(Manager manager[])
{
    int i;
    for(i = 0; i < MAX_MANAGER; i++)
    {
        if(strcmp(manager[i].nama, "-") != 0) // Sudah ada struct terisi
        {
            return false;
        }
    }

    return true;
}
```

```
void createManager(Manager manager[], int index, string nama)
{
    strcpy(manager[index].nama, nama);
}

int getEmptyManager(Manager manager[])
{
    int i;
    for(i = 0; i < MAX_MANAGER; i++)
    {
        if(strcmp(manager[i].nama, "-") == 0) // Masih ada struct kosong
        {
            return i;
        }
    }

    return -1;
}


int findManagerByNama(Manager manager[], string nama)
{
    int i;

    for(i = 0; i < MAX_MANAGER; i++)
    {
        if(strcmp(manager[i].nama, nama) == 0)
        {
            return i;
        }
    }

    return -1;
}


int getEmptyKaryawan(Manager manager)
{
    int i;
    for(i = 0; i < MAX_KARYAWAN; i++)
    {
        if(strcmp(manager.K[i].nama, "-") == 0)
        {
            return i;
        }
    }

    return -1;
}

Karyawan createKaryawan(string nama, int npwp, float salary)
{
    Karyawan temp;
    strcpy(temp.nama, nama);
    temp.npwp = npwp;
    temp.salary = salary;
    return temp;
}


void showAllData(Manager manager[])
{
    int i, j;
    int countM = 0, countK;
    for(i = 0; i < MAX_MANAGER; i++)
    {
        if(strcmp(manager[i].nama, "-") != 0)
        {
            printf("\nDATA MANAGER -%d", countM+1);
            countM++;
            printf("\nNama  : %s\n", manager[i].nama);

            countK = 0;

            for(j = 0; j < MAX_KARYAWAN; j++)
            {
                if(strcmp(manager[i].K[j].nama ,"-") != 0)
                {
                    printf("\n\t DATA Karyawan -%d", countK+1);
                    countK++;
                    printf("\n\t Nama Karyawan  : %s", manager[i].K[j].nama);
                    printf("\n\t NPWP Karyawan  : %d", manager[i].K[j].npwp);
                    printf("\n\t Gaji Karyawan  : %.2f\n", manager[i].K[j].salary);
                }
            }
        }
    }
}
```

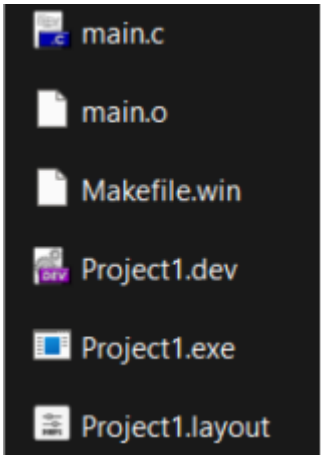
KETENTUAN PENGUMPULAN GUIDED

1. File di zip dengan format penamaan GD12_X_YYYYY.zip

   X = Kelas

   Y = 5 digit terakhir NPM

   Pastikan File Lengkap Seperti Ini.



2. Jangan sampai salah format (Extensi C jangan C++).

3. Pelajari Modul dan Perbanyak Referensi Belajar.

4. Bila ada pertanyaan, bisa langsung hubungi saya via Teams ataupun Whatsapp (Kontak sudah ada di spreadsheet).

5. Jangan copy paste atau pakai tools apapaun untuk buat guided.

6. Tidak menolerir keterlambatan pengumpulan dengan alasan apapun.

7. Hint UGD dan BONUS

(Pelajari Metode CRUDS seperti Guided dan implementasinya untuk array of record di dalam array of record).

(Pelajari penggunaan random untuk angka desimal / float).

(Pelajari implementasi array of record 2D).