Modul V

POINTER

TUJUAN

1. Memahami konsep pointer
2. Memahami implementasi pointer dalam pemrograman Bahasa C

DASAR TEORI

Pointer merupakan sebuah variable yang menyimpan alamat memori dari variable lain. Dengan Pointer kita bisa mengakses nilai/value yang disimpan pada alamat memori tertentu tanpa perlu memanggil variabelnya. Pointer juga sering digunakan untuk mengakses array selain dengan menggunakan indeks.

Perbedaan antara penggunaan variabel biasa dengan pointer terletak pada sifatnya. Variabel biasa bersifat statis dan sudah pasti, sedangkan pada pointer sifatnya dinamis dan lebih fleksibel. Variabel pointer yang tidak menunjuk ke nilai apapun berarti memiliki nilai NULL (disebut juga sebagai *dangling pointer* karena nilainya tidak diinisialisasi dan tidak dapat diprediksi).

Pada operasi pointer dalam bahasa C, kita menggunakan 2 buah operator yaitu *"*"* dan *"&"*. Dalam pembuatan program menggunakan pointer, tanda *asterisk* ("*") digunakan untuk menentukan variable yang menjadi suatu pointer.

Pointer dalam Bahasa C dideklarasikan dengan format:

```
Tipe_Data *nama_variable_pointer;
```

Atau

```
Tipe_Data* nama_variable_pointer;
```

Perlu diperhatikan!

- *Deference Operator* (*) pada **pointer**
  - Jika menggunakan → merujuk ke **nilai** yg ada pada variabel yg alamatnya dipegang
  - Tidak menggunakan → merujuk ke **alamatnya**
- *Unary operator* (&) pada **variabel**
  - Jika menggunakan → merujuk ke **alamatnya**,
  - Tidak menggunakan → merujuk ke **nilainya**

Inisialisasi Pointer

Pointer akan menampung alamat memori, maka dari itu dalam melakukan inisialisasi suatu pointer juga harus memberikan alamat memori. Jika tidak, tentu akan terjadi suatu kesalahan (*error*).

Contoh:

Benar	Salah
`int a = 5;` `int *p = &a;`	`int a = 5;` `int *p = a;`

Assignment Pointer

Assignment adalah memberikan nilai pada suatu variabel, cara untuk melakukannya pada pointer memiliki sedikit perbedaan dengan biasanya.

Contoh:

```
int a = 5;
int *p;
p = &a;
```

Pointer Array

Hubungan antara pointer dan array pada C sangatlah erat. Sebab sesungguhnya array secara internal akan diterjemahkan dalam bentuk pointer. Misalnya dideklarasikan di dalam suatu fungsi, sbb:

```
int tgl_lahir[3] = {1,9,64};
int *ptr;
```

Kemudian diberikan instruksi

```
ptr = &tgl_lahir[0];
```

maka `ptr` akan berisi alamat dari elemen array `tgl_lahir` yang berindeks nol. **Namun**, Instruksi di atas bisa juga ditulis menjadi

```
ptr = tgl_lahir;
```

sebab nama array tanpa tanda kurung (tanpa menyertakan indeks) berarti menyatakan alamat awal dari array. Inilah yang perlu dicermati, bahwa jika kita menggunakan pointer pada array, maka program akan otomatis merujuk pada array elemen pertama (indeks 0). Berbeda hal jika kita menyertakan indeksnya (`&tgl_lahir[0]`)

Guided

1. Guided 1 – *Assignment Pointer*

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

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

    printf("\nMasukkan value untuk variabel X : "); scanf("%d", &x);
    pointerX = &x; //pointerX diassignment alamat variabel x

    printf("\nAlamat dari variabel X adalah: %d dan valuenya %d", pointerX, *pointerX);

    return 0;
}
```

2. Guided 2 – *Pointer Array*

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

int main(int argc, char *argv[]) {
    int myArr[5], i;
    int *pointerArr;

    for(i=0;i<5;i++){
        myArr[i]=i+3; //myArr index ke-i diassignment nilai i+3
    }

    pointerArr = myArr; //sama artinya dengan pointerArr = &myArr[0]
        /*  pointerArr diassignment dengan variabel myArr
            (karena tidak menyertakan indeksnya, maka pointerArr akan otomatis merujuk
            pada array variabel myArr elemen pertama (indeks 0) ) */

    for(i=0;i<5;i++){
        //Amati output alamatnya
        printf("\nAlamat dari elemen ke-%d: %d, value: %d", i+1, pointerArr,*pointerArr);
        pointerArr++; //pointerArr berpindah menunjuk ke alamat dari indeks variabel myArr selanjutnya
    }

    //Mengubah nilai dari indeks tertentu
    printf("\nMasukkan indeks yang ingin diubah (0-4): "); scanf("%d", &i);
    pointerArr = &myArr[i];
        /*  pointerArr diassignment dengan variabel myArr indeks ke- i
            (karena menyertakan indeksnya, maka pointerArr akan merujuk
            pada array variabel myArr indeks i ) */

    printf("\nMasukkan value baru : "); scanf("%d", pointerArr);
    printf("\nAlamat dari elemen yang diubah: %d, value: %d", pointerArr,*pointerArr);

    return 0;
}
```

3. Guided 3 – ***studi kasus***

Diberikan visualisasi data untuk array sebagai berikut:

1	2	3	4	5	6	7	8	9	10

(nomor tersebut adalah id)

Anda sebagai programmer diminta membuatkan program untuk mengakses data yang ada pada array indeks tertentu dengan jelas. Adapun ketika program dijalankan, akan langsung melihatkan isi data dari array indeks 0. Data yang dimaksud meliputi:

- Id
  ➔ Langsung diinisialisasikan secara urut dari 1 s.d. 10
- Nama
  ➔ Tidak boleh kosong dan tidak boleh inputan "-"
- Tanggal terbit
  ➔ Tidak boleh kosong dan tidak boleh inputan "-"

Tampilan awal ketika program dijalankan, sbb:

```
[1 | -] [2 | -] [3 | -] [4 | -] [5 | -] [6 | -] [7 | -] [8 | -] [9 | -] [10 | -]

          Posisi Sekarang di data:
   ID     : 1
   Nama   : -
   Tanggal Terbit : -
```

Adapun syarat utamanya ialah dalam mengakses data harus dengan menggunakan konsep pointer (tidak boleh mengakses dengan menggunakan indeks array). Sementara untuk menu yang ada meliputi:

1. Pilih Posisi

   Pada menu ini, terdapat sub menu. Sub menu yang ada pada menu pilih posisi ini antara lain:

   1) Pilih Data by ID

      Program akan meminta inputan id data yang ingin dituju untuk berpindah posisi. Program akan menampilkan error handling jika id tidak ditemukan atau id sama dengan id data posisi saat itu

   2) Geser Kanan

      Posisi sekarang akan berpindah satu langkah ke kanan apabila id posisi sekarang tidak sama dengan 10. Jika sama tidak dapat mengakses sub-menu ini (posisi sudah mentok)

   3) Geser Kiri

Posisi sekarang akan berpindah satu langkah ke kiri apabila id posisi sekarang tidak sama dengan 1. Jika sama tidak dapat mengakses sub-menu ini (posisi sudah mentok)

2. Isi Data

   Menu ini berfungsi untuk melakukan isi data ke posisi sekarang. Data yang diinputkan berupa nama dan tanggal terbit. Menu ini hanya bisa diakses jika data pada posisi sekarang masih kosong

---

**silakan bisa mencoba membuat sendiri dulu atau bisa langsung lihat code nya berikut:*

a. header.h

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

#define jmlData 10

typedef char string[10];

typedef struct{
    int id;
    string nama;
    string tanggalTerbit;
}Data;

Data makeData(int id, string nama, string t);
void initData(Data data[]);
void tampilData(Data data[]);
void inputan(Data *ptrDisplay);
Data* cariBerdasarID(Data *ptrDisplay, Data data[], int id);
```

b. sourch.c

```cpp
#include "header.h"

Data makeData(int id, string nama, string t){
    Data d;

    d.id=id;
    strcpy(d.nama,nama);
    strcpy(d.tanggalTerbit,t);

    return d;
}

void initData(Data data[]){
    int i, id=1;
    Data *pData;

    pData = data; //pData diassignment variabel data

    for(i=0;i<jmlData;i++){
        *pData = makeData(id, "-", "-");

        id++; //id diincrement
        pData++; //pData berpindah menunjuk ke alamat dari indeks data selanjutnya
    }
}

void tampilData(Data data[]){
    int i;
    Data *pData;

    pData = data; //pData diassignment variabel data

    printf("\n\t");
    for(i=0;i<jmlData;i++){
        printf("[%d | %s] ", pData->id, pData->nama);
        pData++; //pData berpindah menunjuk ke alamat dari indeks data selanjutnya
    }

}

void inputan(Data *ptrDisplay){

    while(true){
        printf("\n\tMasukkan Nama : "); fflush(stdin); gets(ptrDisplay->nama);
        if(strlen(ptrDisplay->nama)==0 || strcmpi(ptrDisplay->nama,"-")==0){
            printf("\n\t\t[!] Nama harus diisi!");
        }else{
            break;
        }
    }
    while(true){
        printf("\n\tMasukkan Tanggal Lahir : "); fflush(stdin); gets(ptrDisplay->tanggalTerbit);
        if(strlen(ptrDisplay->tanggalTerbit)==0 || strcmpi(ptrDisplay->tanggalTerbit,"-")==0){
            printf("\n\t\t[!] Tanggal Lahir harus diisi!");
        }else {
            break;
        }

    }
}

Data* cariBerdasarID(Data *ptrDisplay, Data data[], int id){
    int i,j;
    Data d = makeData(0, "-", "-"); //digunakan jika id yang dicari tidak ditemukan
    Data *ptrCopy, *temp;

    ptrCopy = data; //ptrCopy diassignment variabel data
    temp = &d; //temp diassignment alamat variable d

    for(i=0;i<jmlData;i++){
        if(ptrCopy->id == id){
            return ptrCopy; //jika yg dicari ketemu, maka return ptrCopy
        }else if(ptrDisplay->id == id){ //jika id yang dicari trnyata sama dengan id yang
                                        //sedang ditunjuk pointer sekarang
            return ptrDisplay;          //maka balikin lagi ptrDisplay
        }
        ptrCopy++; //ptrCopy berpindah menunjuk ke alamat dari indeks data selanjutnya
    }
    return temp; //jika yang dicari tdk ktemu, return kan temp yang berisi d

}
```

c. main.c

```c
#include "header.h"

int main(int argc, char *argv[]) {
    int menu, menu2, id;
    char pilihData;
    Data data[jmlData];
    Data *ptrDisplay;

    ptrDisplay = data; //ptrDisplay diassignment variabel data
    initData(data);

    do{
        system("cls");
        tampilData(data);
        printf("\n\n\t\t  Posisi Sekarang di data:");
        printf("\n\t   ID      : %d", ptrDisplay->id);
        printf("\n\t   Nama    : %s", ptrDisplay->nama);
        printf("\n\t   Tanggal Terbit : %s\n\n", ptrDisplay->tanggalTerbit);

        puts("  [1] Pilih Posisi");
        puts("  [2] Isi Data");
        puts("  [0] Keluar");
        printf("\n  >>"); scanf("%d", &menu);

        switch(menu){
            case 1:
                printf("\n\t\t===Pindah Posisi===");
                printf("\n\t[1] Pilih Data by ID");
                printf("\n\t[2] Geser Kanan");
                printf("\n\t[3] Geser Kiri");
                printf("\n\t >>> "); menu2=getche();

                switch(menu2){
                    case '1':
                        printf("\n\t\tMasukkan ID data yang ingin dituju : "); scanf("%d", &id);
                        if(cariBerdasarID(ptrDisplay, data, id)->id == ptrDisplay->id){
                            printf("\n\t\t[!] Sudah berada di ID yang dimaksud!");
                        }else if(cariBerdasarID(ptrDisplay, data, id)->id==0){
                            printf("\n\t\t[!] ID tidak ditemukan!");
                        }else{
                            ptrDisplay=cariBerdasarID(ptrDisplay, data, id);
                            printf("\n\t\t  Berpindah ke ID-%d..", id);
                        }
                        getch();
                        break;

                    case '2':
                        if(ptrDisplay->id==10){
                          //mengecek jika posisi berada di id ke-10, maka tidak bisa geser kanan lagi
                            printf("\n\t\t[!] Sudah Mentok!");
                            getch();
                        }else{
                            ptrDisplay++;
                        }
                        break;

                    case '3':
                        if(ptrDisplay->id==1){
                          //mengecek jika posisi berada di id ke-1, maka tidak bisa geser kiri lagi
                            printf("\n\t\t[!] Sudah Mentok!");
                            getch();
                        }else{
                            ptrDisplay--;
                        }
                        break;

                    default:
                        break;
                }
                break;
```

```
            case 2:
                if(strcmpi(ptrDisplay->nama, "-")!=0){
                    printf("\n\t\t[!] Sudah terisi Data!");
                }else{
                    printf("\n\t\t===Isi Data===");
                    inputan(ptrDisplay);
                    printf("\n\t\tBerhasil Input Data...");
                }
                getch();
                break;

            default:
                break;

        }
    }while(menu!=0);

    return 0;
}
```

Tampilan Awal dari Guided 3

```
      [1 | -] [2 | -] [3 | -] [4 | -] [5 | -] [6 | -] [7 | -] [8 | -] [9 | -] [10 | -]

                Posisi Sekarang di data:
         ID     : 1
         Nama   : -
         Tanggal Terbit : -

[1] Pilih Posisi
[2] Isi Data
[0] Keluar

>>
```

Pengumpulan Guided:

Setiap guided dimasukan ke dalam folder dengan format **GD_NomorGD** kemudian semua guided disatukan dalam satu folder dengan format penamaan **GD5_Y_XXXXX**.zip

(Jadi, di dalam GD5_Y_XXXXX.zip terdapat 3 folder, yakni GD_1, GD_2, dan GD_3 masing-masing folder itu berisi file code sesuai guided di atas)

Ket:
Y = Kelas
XXXXX = 5 digit terakhir NPM

Tambahan:

- Penamaan File Guided diperhatikan
- Pelajari Guided (terutama guided 3) karena akan sangat membantu pada saat UGD
- Bisa coba buat tambahkan menu Ubah Data dan Hapus Data dari guided 3 dengan menggunakan konsep pointer *(tidak wajib)*