MODUL 5
POINTER

Tujuan

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

Dasar Teori

Pointer merupakan sebuah variabel yang menyimpan alamat memori dari variabel lain. Dengan menyimpan alamat pada variabel pointer, nilai pada alamat juga bisa diakses melalui pointer. Pointer memberi programmer kontrol lebih dalam mengakses suatu variabel, karena kebebasannya dalam menunjuk suatu alamat.

Kelebihan pointer memberi keunikan dari variabel lain, yaitu sifatnya yang dinamis dan fleksibel. Variabel biasa memiliki sifat statis dan pasti, sehingga ukuran variabel harus ditentukan terlebih dahulu sebelum dijalankan karena tidak dapat diubah ketika runtime. Pointer bernilaikan sebuah alamat, namun jika tidak ada alamat yang ditunjuk, maka pointer bernilai NULL.

Pengoperasian pointer dalam bahasa C menggunakan 2 buah operator, yaitu "*" dan "&". Contoh deklarasi **variabel pointer**:

```
double * p1;
double* p2; // think p2 is of type "double*"
double *p3; // think *p3 is a double
```

Jarak / penempatan simbol "*" tidak mempengaruhi berjalannya program

Penggunaan simbol "" dan "&"*

1. Deference Operator "*" pada **variabel pointer** :
   - Ketika digunakan pada variabel pointer → mengambil nilai dari alamat suatu variabel
   - Ketika tidak digunakan pada variabel pointer → mengambil alamat yang ditunjuk oleh pointer
2. Unary Operator "&" pada **variabel** :
   - Ketika digunakan pada variabel → mengambil alamat dari variabel
   - Ketika tidak digunakan pada variabel → mengambil nilai dari variabel

Inisialisasi Pointer

Pointer diinisialisasikan dengan simbol “*” dan diisi dengan alamat memori.

```
int var = 20;
int *var_ptr = &var;

atau

int var = 20;
int *var_ptr;
*var_ptr = &var;
```

Oleh karena variabel pointer diisi dengan alamat, maka variabel var perlu diberi tanda “&” di depan untuk memanggil alamat dari var.

Pointer Array

Array secara internal diterjemahkan dalam bentuk pointer, sehingga hubungan antara pointer dan array sangat dekat. Dalam penerapannya, array menggunakan index sebagai penunjuk alamat. Sehingga secara sederhana, index sama halnya dengan pointer yang merepresentasikan sebuah alamat. Berikut contoh sederhana untuk dipahami:

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

int main(int argc, char *argv[]) {
  char hari[3] = {'D', 'A', 'Y'};
  char *hariPtr;

  hariPtr = hari;
  // atau hariPtr = &hari[0];

  int i;
  for(i = 0; i< 3; i++){
    printf("%c", *hariPtr++);
    // atau printf("%c", hari[i]);
    // atau printf("%c", *(hariPtr+i)); // mengapa bagian ini diberi kurung setelah "*"?
  }

    return 0;
}
```

Hal yang perlu diperhatikan yaitu bahwa dalam assigning pointer ke array, dapat dilakukan tanpa menuliskan index, dan pointer akan langsung menunjuk index pertama dalam array. Array dalam contoh di atas juga digambarkan sebagai variabel yang alamatnya bersebelahan, sehingga dengan menambah nilai pointer seperti hariPtr+1 maka akan menunjuk ke index selanjutnya.

Dynamic Array

Array dinamis adalah array yang ukurannya ditentukan ketika runtime. Dengan array dinamis, program tidak terpaku pada ukuran array yang telah ditentukan sebelum program dijalankan. Pengaturan array dinamis, bisa dilakukan menggunakan pointer.

Cara alokasi array dinamis sebagai berikut:

```
<tipedata> *array;
int size = array_size;
*array = (<tipedata>*) malloc(array_size * sizeof(<tipedata>));
```

Pada contoh di atas menggunakan fungsi bawaan yaitu `malloc` yang mengalokasikan suatu blok memori. Setelah suatu memori dialokasikan, tipe data dari setiap alamat dikonversikan (menggunakkan casting) sesuai dengan (`<tipedata>`). Jika ingin membuat array data yang dibuat merupakan kelipatan dari tipe data yang ingin digunakan.

GUIDED

1. Guided 1 – Assignment Pointer

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

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

    printf("\nMasukkan nilai untuk variabel x : ");scanf("%d", &x);
    // penggunaan tanda "&" ketika melakukan scanf, mengarahkan di mana variabel yang di scan ditempatkan
    xPointer = &x; // pointer menunjuk ke alamat dari variabel x

    printf("\nx:\n");
    printf("Alamat  : %d\n", &x); // seharusnya angka yang tampil sama dengan alamat yang ditunjuk oleh xPointer
    printf("Value   : %d\n", x);

    printf("\nxPointer:\n");
    printf("Alamat pointer  : %d\n", &xPointer); // sama dengan variabel biasa, variabel pointer juga memiliki alamat tetap
    printf("Alamat tunjuk   : %d\n", xPointer); // alamat yang ditunjuk oleh pointer
    printf("Value           : %d\n", *xPointer); // nilai dari alamat yang ditunjuk pointer

    return 0;
}
```

2. Guided 2 – Pointer Array

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

int main(int argc, char *argv[]) {
    int arr[5], i;
    int *ptrArr;

    for(i=0;i<5;i++){
        arr[i] = i+2; // setiap index array bernilai index + 2
    }

    ptrArr = arr;
    // pointer di assign ke index pertama
    // penggunaan di atas sama dengan ptrArr = &arr[0]
    // alamat ptrArr akan menunjuk ke alamat arr index ke-0

    for(i=0;i<5;i++){
        printf("\nAlamat pada elemen %d : %d, value : %d", i, ptrArr, *ptrArr);
        ptrArr++; // pointer menunjuk alamat index arr selanjutnya
    }

    // mengubah nilai arr dengan pointer
    printf("\n\nMasukkan index yang ingin diubah (0-4) : ");scanf("%d", &i);
    ptrArr = &arr[i]; // pointer menunjuk alamat arr index ke-i sesuai inputan

    printf("\n\nMasukkan value : ");scanf("%d", ptrArr);
    // &arr[i] == ptrArr
    // karena sebelumnya telah di assign ptrArr sama dengan alamat arr index ke-i

    printf("\n Alamat pada elemen %d : %d, value : %d", i, ptrArr, *ptrArr);

    return 0;
}
```

3. Guided 3 – Dynamic Array

header.h

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

typedef char string[128];

typedef struct{
    string nama;
    int id;
}User;

typedef struct{
    User *user; // variabel pointer akan menunjuk alamat dengan tipe data
                // User baik alamat variabel biasa maupun array
    int size;
}UserList;

void initUserArray(UserList *user, int size);
void initUserList(UserList *user, int size);
```

source.c

```c
#include "header.h"

void initUserArray(UserList *user, int size){
    if(size < 1){
        printf("ukuran harus lebih dari 0");
    }
    (*user).user = (User*) malloc(size * sizeof(User));
    // malloc adalah fungsi untuk mengalokasi data
    // tipe data (User*) memberiinformasi mengenai tipe data dari alamat yang dialokasikan
    // parameter dalam malloc merupakan ukuran yang akan digunakan oleh variabel
    (*user).size = size;
}

void initUserList(UserList *user, int size){
    initUserArray(&(*user), size);
    int i;

    for(i=0;i<size;i++){
        strcpy((*user).user[i].nama, "- kosong -");
        (*user).user[i].id = i+1;
    }
}
```

main.c

```c
#include "header.h"

int main(int argc, char *argv[]) {
    UserList ul;
    int jml_arr;
    int i;

    printf("\nMasukkan jumlah array : ");scanf("%d", &jml_arr);

    // inisialisasi array sesuai dengan jumlah
    initUserList(&ul, jml_arr);

    // memastikan jumlah array yang terisi
    for(i=0;i<ul.size;i++){
        printf("\ndata %d\t", i+1);
        if(i<9)
            printf("\t");
        printf(": %s", ul.user[i].nama);
    }
    return 0;
}
```

4. Guided 4 – Studi Kasus

Berikut adalah visualisasi data dalam array:

1	2	3	4	5

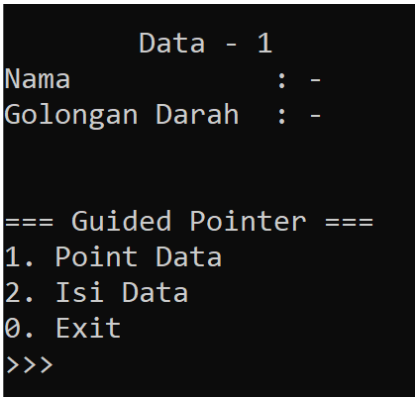
(nomor di atas adalah id)

Sebagai programmer, anda diminta untuk mengakses data pada array index tertentu menggunakan pointer. Ketika array berjalan, program akan memperlihatkan isi index pertama yaitu index ke-0. Data pada array meliputi:

- Id
  - Tipe data Integer
  - Diinisialisasikan secara urut angka 1-5
- Nama
  - Tipe data string dengan max length 128
  - Inputan tidak boleh kosong atau “-“

- Golongan Darah
  - Tipe data char
  - Inputan tidak boleh “-“

Tampilan awal ketika program dijalankan sebagai berikut.



Berikut menu utama yang harus anda penuhi menggunakan konsep pointer:

1. Point Data

   Menu point data atau tunjuk data digunakan untuk memilih index. Anda diminta untuk membuat sub menu pada menu point data untuk memilih index dengan beberapa cara:

   1) Pilih Data by Id – user menginputkan id, jika ditemukan pointer akan menunjuk index
   2) Geser Kanan – pointer akan menunjuk index selanjutnya, jika ada
   3) Geser Kiri – pointer akan menunjuk index sebelumnya, jika ada

2. Isi Data

   Menu ini berfungsi untuk mengisi data pada index yang ditunjuk pointer, hanya jika data yang ditunjuk masih kosong. Data yang di inputkan adalah nama dan golongan darah.

header.h

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

#define MAX_SIZE 5

typedef char string[128];

typedef struct{
    int id;
    string nama;
    char golongan_darah;
}Data;

void initData(Data data[]);
Data editData(int id, string nama, char golongan_darah);
void printData(Data *DPtr);
```

source.c

```c
#include "header.h"

void initData(Data data[]){
    Data *ptrData;
    int i=1;

    /*
       Perulangan di bawah sama dengan perulangan berikut
       for(i = 0; i < MAX_SIZE; i++)
       hanya saja menggunakan pointer untuk menunjuk ke alamat selanjutnya
    */
    for(ptrData = data; ptrData != &data[MAX_SIZE]; ptrData++){
        (*ptrData).id = i++;
        strcpy((*ptrData).nama, "-");
        (*ptrData).golongan_darah = '-';
    }
}

Data editData(int id, string nama, char golongan_darah){
    Data d;
    d.id = id;
    strcpy(d.nama, nama);
    d.golongan_darah = golongan_darah;

    return d;
}

void printData(Data *DPtr){
    // pemanggilan variabel struct pointer dapat menggunakan panah seperti dibawah
    printf("\n\tData - %d", (*DPtr).id);
    printf("\nNama      : %s", (*DPtr).nama);
    printf("\nGolongan Darah    : %c", (*DPtr).golongan_darah);
}
```

main.c

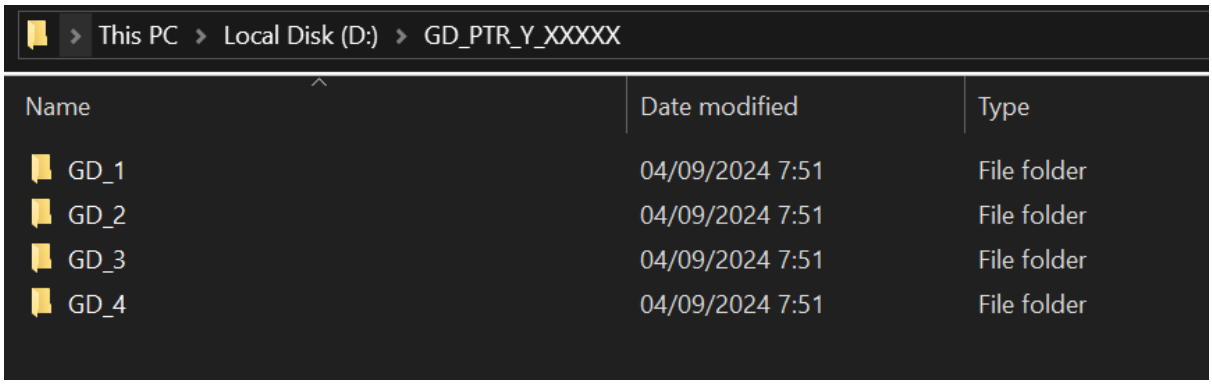
```c
#include "header.h"

int main(int argc, char *argv[]) {
    int menu, subMenu;
    int search;
    string nama;
    char golongan_darah;
    Data D[5];
    Data *DPtr = D;

    initData(D);

    //NPM | NAMA LENGKAP

    do{
        system("cls");
        printData(DPtr);

        printf("\n\n\n=== Guided Pointer ===");
        printf("\n1. Point Data");
        printf("\n2. Isi Data");
        printf("\n0. Exit");
        printf("\n>>> "); scanf("%d", &menu);

        switch(menu){
            case 1:
                do{
                    system("cls");
                    printf("\n=== Point Data ===");
                    printf("\n1. Select By Id");
                    printf("\n2. Move Left");
                    printf("\n3. Move Right");
                    printf("\n0. Back");
                    printf("\n>>> "); scanf("%d", &subMenu);

                    switch(subMenu){
                        case 1:
                            printf("Masukan Id (1-10) : "); scanf("%d", &search);
                            if(search < 1 || search > 10){
                                printf("\n\t [*] Id tidak ditemukan");
                                break;
                            }
                            DPtr = &D[search-1];
                            printf("\n\t[*] Berhasil Geser");
                            break;
```

```c
                    case 2:
                        if(DPtr == &D[0]){
                            printf("\n\tSudah Mentok [!]");
                            break;
                        }
                        DPtr--;
                        printf("\n\t[*] bergeser ke Id - %d", (*DPtr).id);
                        break;

                    case 3:
                        if(DPtr == &D[MAX_SIZE-1]){
                            printf("\n\tSudah Mentok [!]");
                            break;
                        }
                        DPtr++;
                        printf("\n\t[*] bergeser ke Id - %d", (*DPtr).id);
                        break;

                    case 0:
                        printf("\n\tpress any key to go back...");
                }
                getch();
            }while(subMenu!=0);
            break;

        case 2:
            if(strcmpi((*DPtr).nama, "-")!=0){
                printf("\n\tData Sudah Di Isi [!]");
                break;
            }
            system("cls");
            printf("\nMasukkan Data: ");
            do{
                printf("\n\tNama        : "); fflush(stdin); gets(nama);
                if(strcmp(nama, "-") == 0 || strlen(nama) == 0){
                    printf("\n\t\tNama Tidak Boleh Kosong [!]\n");
                }
            }while(strcmp(nama, "-") == 0 || strlen(nama) == 0);

            do{
                printf("\n\tGolongan Darah  : "); golongan_darah = getch();
                if(golongan_darah == '-'){
                    printf("\n\t\tGolongan Darah Tidak Boleh '-' [!]\n");
                }
            }while(golongan_darah == '-');

            *DPtr = editData((*DPtr).id, nama, golongan_darah);
            printf("\n\t[*] Data Berhasil Diisi");
            break;

        case 0:
        printf("\nIt's done already");
        printf("\nWho did it ?");
        printf("\nGOD DID !!");
        break;
    }
    getch();
}while(menu!=0);
return 0;
}
```

Pengumpulan

Setiap guided di simpan dalam folder **GD_<Nomor GD>**. Kemudian disatukan dalam suatu folder dan di kompres dalam bentuk zip dengan format penamaan **GD_PTR_Y_XXXXX.zip**.

(dalam zip akan berisi folder **GD_1, GD_2,GD_3, dan GD_4** tanpa NPM, NPM hanya dituliskan di zip utama)



Ket:

Y = Kelas

X = 5 digit terakhir NPM

Note:

- Pelajari kembali yang berhubungan dengan array, karena sangat berhubungan dengan pointer dan modul 2 akan sangat berhubungan dengan UGD kedepannya.
- Perdalam pemahaman tentang penggunaan malloc, dan istilah-istilah seperti casting, dll.

Source:

Belajar Pemrograman C #15: Apa itu Pointer? (petanikode.com)

Pointers (drexel.edu)

C - Pointer to Pointer (Double Pointer) - GeeksforGeeks