DAFTAR ISI

DAFTAR PROGRAM

Bab 2. POINTER

Pointer merupakan tipe data berukuran 32 bit yang berisi satu nilai yang berpadanan dengan alamat memori tertentu. Sebagai contoh, sebuah variabel P bertipe pointer bernilai 0x0041FF2A, berarti P menunjuk pada alamat memori 0041FF2A. Pointer dideklarasikan seperti variabel biasa dengan menambahkan tanda * (asterik) yang mengawali nama variabel.

Bentuk Umum:

<tipe data> namaVariabel;

Contoh :

float * px;

Statement di atas mendeklarasikan variabel px yang merupakan pointer. Penyebutan tipe data float berarti bahwa alamat memori yang ditunjuk oleh px dimaksudkan untuk berisi data bertipe float.

Contoh Program 1:

```
#include <iostream.h>

void main()
{
   int x;
   int *px;

   x = 2;
   px = &x;   //membaca alamat dari x

   cout<<"Nilai x             : "<<x<<endl;
   cout<<"Nilai *px           : "<<x<<endl;
   cout<<"Nilai px (alamat x) : "<<px<<endl;
}
```

Output Program 2 :

```
Nilai x             : 2
Nilai *px           : 2
Nilai px (alamat x) : 0x2467242e
```

Contoh Program 2 :

```
#include <iostream.h>

void main()
{
   int x[10]={0,1,2,3,4,5,6,7,8,9};
   int *px;
   int i;

   for(i=0;i<10;i++)
   {
      px = &x[i];   //membaca alamat dari x


      cout<<x[i]<<" "<<*px<<" "<<px<<endl;
   }
}
```

Output Program 2 :

```
0 0 0x250f23da
1 1 0x250f23dc
2 2 0x250f23de
3 3 0x250f23e0
4 4 0x250f23e2
5 5 0x250f23e4
6 6 0x250f23e6
7 7 0x250f23e8
8 8 0x250f23ea
9 9 0x250f23ec
```

Contoh Program 3 :

```
#include <iostream.h>

void main()
{
   char *nama;

   nama = "Muhammad Fachrurrozi";
   cout<<"Selamat datang "<<nama<<endl;
}
```

Output Program 3 :

```
Selamat datang Muhammad Fachrurrozi
```

Bab 3. ARRAY

Array adalah suatu struktur yang terdiri dari sejumlah elemen yang memiliki tipe data yang sama. Elemen-elemen array tersusun secara sekuensial dalam memori komputer. Array dapat berupa satu dimensi, dua dimensi, tiga dimensi ataupun banyak dimensi (multi dimensi).

3.1. Array Satu Dimensi

Array Satu dimensi tidak lain adalah kumpulan elemen-elemen identik yang tersusun dalam satu baris. Elemen-elemen tersebut memiliki tipe data yang sama, tetapi isi dari elemen tersebut boleh berbeda.

Elemen ke-	0	1	2	3	4	5	6	7	8	9
Nilai	50	35	10	15	23	11	25	34	23	32

Bentuk umum:

<tipe data> NamaArray[n] = {elemen0, elemen1, elemen2,.....,n};

n = jumlah elemen

Contoh Program 4 :

```
/* Program Mencari bilangan terkecil
                Dan terbesar di dalam array */

#include<iostream.h>

void main()
{   int x[10]={50, 35, 10, 15, 23, 11, 25, 34, 23, 32};
    int i;
    int maks = -1000; //asumsi paling minimum
    int min = 1000;  //asumsi paling maksimum
    for (i=0; i<10; i++)
    {   if(x[i]>maks)
        {
            maks= x[i];
        }
        if(x[i]<min)
        {
            min= x[i];
        }
    }
    cout<<"Nilai maksimum : "<<maks<<endl;
    cout<<"Nilai minimum : "<<min<<endl;
}
```

Output Program 4 :

```
Nilai maksimum : 50
Nilaia minimum : 10
```

3.2. Array Dua Dimensi

Array dua dimensi sering digambarkan sebagai sebuah matriks, merupakan perluasan dari array satu dimensi. Jika ***array satu dimensi hanya terdiri dari sebuah baris*** dan beberapa kolom elemen, maka ***array dua dimensi terdiri dari beberapa baris*** dan beberapa kolom elemen yang bertipe sama sehingga dapat digambarkan sebagai berikut:

	0	1	2	3	4	5	6
0	10	21	23	43	45	78	65
1	45	43	65	12	21	12	21
2	32	34	23	56	54	34	45
3	11	12	32	23	56	76	45

Bentuk umum:

```
<tipe data> NamaArray [m][n];
```

Atau

```
<tipe data> NamaArray [m][n] = { {a,b,..z},{1,2,...,n-1} };
```

Contoh :

```
double matrix[4][4];
bool papan[2][2] = { {true,false},{true,false} };
```

Pendeklarasian array dua dimensi hampir sama dengan pendeklarasian array satu dimensi, kecuali bahwa array dua dimensi terdapat dua jumlah elemen yang terdapat di dalam kurung siku dan keduanya boleh tidak sama.

Elemen array dua dimensi diakses dengan menuliskan kedua indeks elemennya dalam kurung siku seperti pada contoh berikut:

```
//papan nama memiliki 2 baris dan 5 kolom
    bool papan[2][5];
```

papan[0][0] = true;

papan[0][4] = false;

papan[1][2] = true;

papan[1][4] = false;

Contoh Program 5 :

```
/* Program Penjumlahan 2 Buah Matriks 2 x 2 */

#include<iostream.h>
#include<conio.h>

#define Nmaks 10

typedef int matrik[Nmaks][Nmaks];

void main()
{  int n,i,j;
   Matrik A,B,C;

   cout<<"Program Penjumlahan Matrik A 2x2 dan B 2x2"<<endl;
   cout<<endl;

   n=2;
   cout<<"Masukkan Nilai Matrik A"<<endl;
   for (i=1; i<=n; i++)
   {   for (j=1; j<=n; j++)
       {
            cout<<"A["<<i<<","<<j<<"]";
            cin>>A[i][j[;
       }
    }

    clrscr();
    cout<<"Masukkan Nilai-nilai Matrik B"<<endl;
    for (i=1; i<=n; i++)
    {   for (j=1; j<=n; j++)
        {
            cout<<"B["<<i<<","<<j<<"]";
            cin>>B[i][j[;
       }
    }

    clrscr();
    cout<<endl;
    //Proses Penjumlahan Matrik C = A + B
    for (i=1; i<=n; i++)
    {   for (j=1; j<=n; j++)
        {
            C[i][j]=A[i][j] +B[i][j];
       }
    }
```

```
    clrscr();
    cout<<"Nilai-nilai Matrik A, B dan C"<<endl;
    cout<<endl;

    //Proses Output Matrik A
    gotoxy(1,5);
    cout<<"A = ";
    for (i=1; i<=n; i++)
    {   for (j=1; j<=n; j++)
        {
            Gotoxy(2+4*j,2+2*i);
            Cout<<A[i][j];
        }
    }

    //Proses Output Matrik B
    gotoxy(1,10);
    cout<<"B = ";
    for (i=1; i<=n; i++)
    {   for (j=1; j<=n; j++)
        {
            gotoxy(2+4*j,7+2*i);
            cout<<B[i][j];
        }
    }

    //Proses Output Matrik C
    gotoxy(1,15);
    cout<<"C = ";
    for (i=1; i<=n; i++)
    {   for (j=1; j<=n; j++)
        {
            Gotoxy(2+4*j,12+2*i);
            Cout<<C[i][j];
        }
    }

    gotoxy(12,15);
    cout"+";
    for (i=1; i<=n; i++)
    {   for (j=1; j<=n; j++)
        {
            gotoxy(13+4*j,12+2*i);
            Cout<<B[i][j];
        }
    }

    gotoxy(23,15);
    cout"+";
    for (i=1; i<=n; i++)
    {   for (j=1; j<=n; j++)
        {
            gotoxy(24+4*j,12+2*i);
            Cout<<B[i][j];
        }
    }
}
```

Output Program 5 :

```
Program Penjumlahan Matrik A 2x2 dan B 2x2

Masukkan Nilai-Nilai Matrik A
A[1,1] = 1
A[1,2] = 2
A[2,1] = 5
A[2,2] = 8_

Masukkan Nilai-Nilai Matrik B
B[1,1] = 5
B[1,2] = 1
B[2,1] = 6
B[2,2] = 4

Nilai-Nilai Matriks A, B, dan C


     1   2
A =
     5   8


     5   1
B =
     6   4


     1   2       5   1       6   3
C =          +           =
     5   8       6   4       11  12
```

LATIHAN:

1. Buat penjumlahan matrik ordo 3x3
2. Buat perkalian matrik ordo 2x2
3. buat perkalian matrik ordo 3x3

BAB 4 STRUCTURE

Structure adalah kumpulan elemen-elemen data yang digunakan menjadi satu kesatuan. Masing-masing elemen data tersebut dikenal dengan sebuah ield. Field data tersebut dapat memiliki tipe data yang sama ataupun berbeda. Walaupun field-field tersebut berada dalam satu kesatuan, masing-masing field tersebut tetap dapat diakses secara individual.

Bentuk umum

```
struct namastruct
{
        <tipe data> field1;
        <tipe data> field2;
        <tipe data> field3;
};
```

Contoh :

```
struct mahasiswa
{
        char nim[11];
        char nama[30];
        char alamat[50];
        float ipk;
};
```

Untuk menggunakan structure, tulis nama structure beserta dnegan fieldnya yang dipisahkan dengan tanda titik (“.”). Misalnya Anda ingin menulis nim seorang mahasiswa ke layar maka penulisannya yang benar adalah sebagai berikut :

```
cout<<mahasiswa.nim;
```

Jika Pmhs adalah pointer bertipe mahasiswa* maka field dari Pmhs dapat diakses dengan mengganti tanda titik dengan tanda panah (”→”).

```
cout<<mahasiswa->nim;
```

Contoh Program 6

```
/*Mengisi Biodata dan Nilai IPK mahasiswa*/
#include<iostream.h>

struct mahasiswa
{
      char nim[15];
      char nama[30];
      char alamat[50];
      float ipk;
};

void main()
{
      mahasiswa mhs;

      cout<<"NIM  :  "; cin.getline(mhs.nim,15);
      cout<<"Nama : ";  cin.getline(mhs.nama,30);
      cout<<"Alamat     : ";
cin.getline(mhs.alamat,50);
      cout<<"Nilai IPK  : ";  cin>>mhs.ipk;

      cout<<endl;
      cout<<endl;

      cout<<"NIM Anda   :  "<<mhs.nim
      cout<<"Nama Anda  : "<<mhs.nama;
      cout<<"Alamat Anda      : " <<mhs.alamat;
      cout<<"Nilai IPK Anda   : "<<mhs.ipk<<endl;
}
```

Output Program 5 :

```
NIM        : 09983110077
Nama       : M. Fachrurrozi
Alamat     : Jl. Joko Atas No 23 Palembang
Nilai IPK  : 3.00

NIM Anda        : 09983110077
Nama Anda       : M. Fachrurrozi
Alamat Anda     : Jl. Joko Atas No 23 Palembang
Nilai IPK Anda  : 3
```

Bab 5 LINKED LIST

Pada bab sebelumnya dibahas mengenai variabel array yang bersifat statis dimana ukuran dan urutannya sudah pasti. Serta ruang memori yang dipakai tidak dapat dihapus bila array tersebut sudah tidak digunakan lagi pada saat program dijalankan. Penyelesain permasalahn tersebut dapat dilakukan dengan menggunakan variabel **pointer**. Tipe data pointer bersifat dinamis, variabel akan dialokasikan hanya pada saat dibutuhkan dan sesudah tidak dibutuhkan dapat direlokasikan kembali.

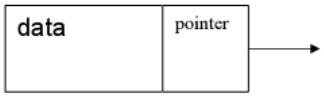
Setiap ingin menambah data selalu menggunakan variabel pointer baru, yang akan mengakibatkan membutuhkan banyak pointer. Permasalahan ini dapat dipecahkan dengan menggunakan satu variabel pointer saja untuk menyimpan banyak data dengan metode Linked List. Linked List adalah salah satu bentuk struktur data, berisi kumpulan data (node) yang **tersusun** secara sekuensial, **saling sambung-menyambung, dinamis** dan **terbatas**.

Bentuk umum :

```
typedef struct telmtlist
    {
        infotype info;
        address next;
    }elmt list;
```

infotype → sebuah tipe terdefinisi yang menyimpan informasi sebuah elemen list
next → adddress dari elemen berikutnya (sulsesor)

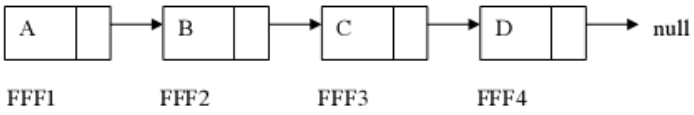
5.1 Single Linked List



Menempati alamat memori tertentu

Pengertian :

- Single : artinya field pointer-nya hanya satu buah saja dan satu arah serta pada akhir node, pointernya menunjuk NULL
- Linked List : artinya node-node tersebut saling terhubung satu sama lain.



Ilustrasi Linked List

- Setiap node pada linked list mempunyai field yang berisi pointer ke node berikutnya, dan juga memiliki field yang berisi data.
- Node terakhir akan menunjuk ke NULL yang akan digunakan sebagai kondisi berhenti pada saat pembacaan isi linked list.

Pembuatan Single Linked List dapat menggunakan 2 metode :

- LIFO (Last In First Out), aplikasinya : Stack
  Suatu metode pembuatan Linked List dimana data yang masuk paling akhir adalah data yang keluar paling awal. Penambahan (insert) simpul dilakukan melalui belakang, yang dikenal dengan istilah INSERT.
- FIFO (First In Last Out), aplikasinya : Queue
  Suatu metode pembuatan Linked List dimana data yang masuk paling awal adalah data yang keluar paling awal.Penambahan (insert) simpul dilakukan di depan.

Contoh Program 7 :

```
/* Membuat Single Linked List */

#include<iostream.h>
#include<stdlib.h>
#include<malloc.h>
```

```
#include<conio.h>

#define Nil NULL
#define info(P) P->info
#define next(P) P->next
#define First(L) (L)

typedef int InfoType;
typedef struct telmtlist *address;
typedef struct telmtlist
{
      InfoType info;
      Address next;
}elmtlist;

typedef address list;

void BuatLinked(list *L)
{
   First(*L) = Nil;
}

list NodDaru(int m)
{
   list n;
   n = (list) malloc(sizeof (elmlist));
   if (n != NULL)
   {
      info(n) m;
      next(n) = Nil;
   }

   return n;
}

void SisipSenarai (list *L, list t, list p)
{
   if (p ==Nil)
   {
     t->next = *L;
     *L = t;
   }
   else
   {
     t->next = p->next;
     p->next = t;
   }
}

void CetakSenarai(list L)
{
   list ps;
   for (ps=L; ps!=Nil; ps=ps->next)
   {
     cout<<" " <<info(ps)<<"-→";
   }
   cout<<"NULL "<<endl;
}
```

```
int mail()
{
  list pel;
  list n;
  int I,k,nilai;

  CiptaSenarai(&pel);
  cout<<"Masukkan Banyak Data = ";
  cin>>k;
  for(i=1; i<=k; i++)
  {
     cout<<"Masukkan Data Senarai ke- "<<i<<" = ";
     cin>>nilai;
     n = NodBaru(nilai);
     SisipSenarai(&pel, n, NULL);
  }
  CetakSenarai(pel);
  return 0;
}
```

5.2 Double Linked List

salah satu kelemahan single linked list adalah pointer hanya dapat bergerak satu arah saja, maju/ mundur, atau kanan/kiri sehingga pencarian data pada single linked list hanya dapat bergerak ke satu arah. Untuk mengatasi kelemahannya dapat digunakan metode double linked list. Linked list ini dikenal dengan nama Linked list berpointer Ganda atau Double Linked List.

5.3 Circular Doubel Linked List

CDLL merupakan double linked list yang simpul terakhirnya menunjuk ke simpul awal dan simpul awal menunjuk ke simpul akhir sehingga membnetu suatu lingkaran.

Operasi-operasi yang ada pada Linked List

Insert : menmabhkan sebuah simpul baru ke dalam suatu linked list

IsEmty : fungsi yang menentukan apakah linked list kosong atau tidak

Find First : fungsi untuk mencari elemen pertama dari linked list

Find Next : fungsi untuk mencari elemen sesudah elemen yang ditunjuk now

Retrieve : fungsi untuk mengambil elemen yang ditunjuk oleh now. Elemen tersebut lalu dikembalikan oleh fungsi.

Updatei : fungsi untuk mengubah elemen yang ditunjuk oleh now dengan isi dari sesuatu.

Delete Now : fungsi untuk menghapus elemen yang ditunjuk oleh now. Jika yang dihapus adalah elemen pertama dari linked list (head), head akan berpindah ke elemen berikut.

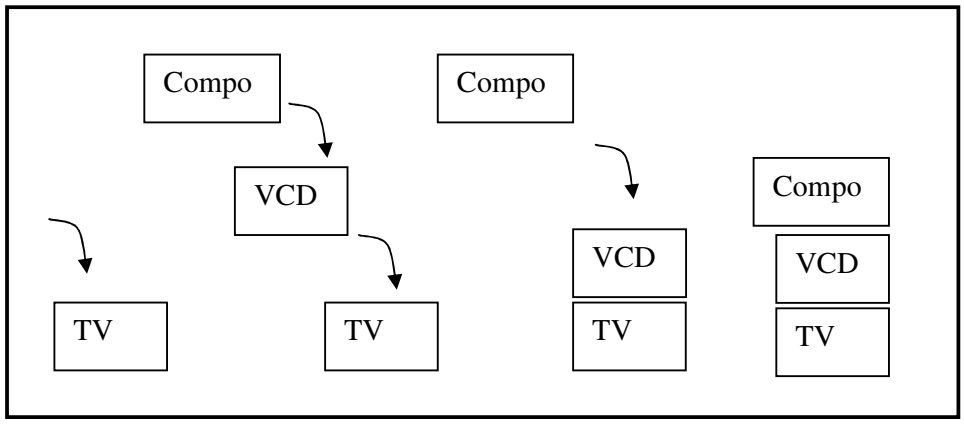
Delete Head : fungsi untuk menghapus elemen yang ditunjuk Head. Head berpindah ke elemen sesudahnya.

Clear : fungsi untuk menghapus linked list yang sudah ada. Fungsi ini wajib dilakukan bila anda ingin mengakhiri program yang menggunakan linked list. Jika anda melakukannya, data-data yang dialokasikan ke memori pada program sebelumnya akan tetap tertinggal di dalam memori.

Bab 6. STACK

6.1 Definisi Stack

Stack adalah tumpukkan dari benda. Suatu tumpukkan item, dimana dapat memasukkan item baru dan menghapus item yang terakhir. Konsep utama LIFO (Last In Firt Out), benda yang terakhir masuk dalam stack akam menjadi benda yang pertama kali dikeluarkan dari stack.



Dari gambar diatas ketika akan mengambil barang, maka barang yang pertama kali diambil adalah **compo** kemudian **VCD** dan **TV.** Di C++, ada dua cara penerapan prinsip stack, yakni dengan **array** dan **linked list.** Operasi –operasi dalam stack adalah :

Push	untuk menambahkan item pada tumpukan paling atas
Pop	untuk mengambil item teratas
Clear	untuk mengosongkan stack
IsEmpty	untuk memeriksa apakah stack kosong
IsFull	untuk memeriksa apakah stack sudah penuh
Retreive	untuk mendapatkan nilai dari item tersebut.

6.2 Stack dengan Array

sesuai dengan sifat stack, pengambilan / penghapusan di elemen dalam stack harus dimulai dari elemen teratas.

Operasi-operasi pada Stack dengan Array

IsFull

Fungsi digunakan untuk memeriksa apakah stack yang ada sudah penuh. Stack penuh jika puncak stack terletak tepat di bawah jumlah maksimum yang dapat ditampung stack (**Top = MAX_STACK -1**)

Push

Fungsi yang digunakan untuk menambahkan sebuah elemen ke dalam stack dan tidak bisa dilakukan jika stack sudah penuh

IsEmpty
Fungsi yang digunakan untuk menentukan apakah stack kosong atau tidak. Tanda bahwa stack kosong adalah **Top** bernilai kurang dari nol.

Pop
Fungsi yang digunakan untuk mengambil elemen teratas dari stack dengan syarat stack tidak kosong

Clear
Fungsi yang digunakan untuk mengosongkan stack dengan cara mengeset Top dengan -1. Jika Top bernilai kurang dari nol maka stack dianggap kosong.

Retrieve
Fungsi yang digunakan untuk melihat nilai yang berada pada posisi tumpukan teratas.

Contoh Program 8 :

```
/* Program untuk Insert (Push) Nilai dan Delete(Pop) Nilai dalam Stack */

#include <iostream.h>
#include<stdio.h>
#include<stdlib.h>
#define MAX 10

void push (int stack[], int *top, int value);
void pop (int stack[], int *top, int *value);

void main()
{
    int stack[MAX];
    int top = -1;
    int n, value;

    do
    {
       do
        {
           cout<<"Masukkan Nilai yang akan di Push : ";
           cin>>value;
           push(stack, &top, value);

           cout<<"Tekan 1 untuk Melanjutkan" << endl;
           cin>>n;
        } while (n = =1);

        cout<<"Tekan 1 untuk Melanjutkan Pop" <<endl;
        cin>>n;
```

```
        while (n = = 1)
        {
          pop(stack, &top, &value);
          cout<<"Nilai yang di Pop : "<<value;
          cout<<"Tekan 1 untuk Melakukan Pop sebuah Elemen" <<endl;
          cin>>n;
        }

        cout<<endl;
        cou<<"Tekan 1 untuk melanjutkan"<<endl;
        cin>>n;
      } while (n = = 1);
}

void push (int stack[], int *top, int value)
{
    if (*top < MAX)
    {
        *top = *top + 1;
        stack[*top] = value;
    }
    else
    {
        cout<< "Stack Penuh, Push Nilai Tidak Dapat Dilakukan"<<endl;
        exit(0);
    }
}

void pop(int stack[], int *top, int *value)
{
    if (*top >= 0)
    {
        *value = stack[*top];
        *top = *top – 1;
    }
    else
    {
        cout<<"Stack Kosong, Pop Tidak Dapat Dilakukan" <<endl;
        exit(0);
    }
}
```