BAB 1. Pointer & Array 1 Dimensi

1.1 Definisi Pointer

Sebuah pointer berfungsi sebagai variabel penunjuk yang menyimpan alamat memori suatu lokasi tertentu. Pointer tidak menampung nilai data, melainkan alamat memori yang dapat mewakili sebuah variabel atau alamat memori secara langsung.

Sebagai contoh, jika kita memiliki variabel x yang berada di alamat memori 0x000001 dan ingin menetapkan nilai 100 ke dalam variabel tersebut, maka processor harus mengakses alamat memori 0x000001 untuk memperbarui nilainya. Perlu diingat bahwa ukuran setiap variabel berbeda-beda dalam memori dan dapat diwakili oleh beberapa lokasi memori. Misalnya, variabel bertipe int biasanya memakan 4 byte dalam memori, dan dapat menempati beberapa lokasi memori seperti 0x000001, 0x000002, 0x000003, dan 0x000004. Jika dua variabel bertipe int bersebelahan dalam memori, maka alamat variabel pertama akan menjadi 0x000001 dan variabel kedua akan berada pada alamat 0x000005. Biasanya, bilangan heksadesimal digunakan untuk mengidentifikasi alamat memori, yang ditandai dengan awalan '0x'. Sebagai contoh, jika variabel menempati blok kesepuluh dalam memori, alamatnya akan menjadi 0x00000a.

<table>
<tr><td>0x000001</td><td rowspan="4">int x</td></tr>
<tr><td>0x000002</td></tr>
<tr><td>0x000003</td></tr>
<tr><td>0x000004</td></tr>
<tr><td>0x000005</td><td rowspan="4">int y</td></tr>
<tr><td>0x000006</td></tr>
<tr><td>0x000007</td></tr>
<tr><td>0x000008</td></tr>
</table>

1.2 Operator Pointer

Terdapat dua operator pointer yang ada pada tipe data pointer, yaitu:

a. Operator Reference (&)

Reference adalah suatu operator yang berfungsi untuk mendapatkan alamat dari suatu variabel. Apabila kita memberikan simbol & pada awal variabel dan mencetak hasilnya pada CLI, maka yang akan tercetak adalah alamat dari variabel tersebut dan bukan nilai yang ditampung oleh variabel tersebut.

Contoh program 1 :

```cpp
#include <iostream>

using namespace std;

int main(){
    int x = 9;
    cout<<"Alamat Variabel x :"<<&x<<endl;
    cout<<"Nilai Variabel x :"<<x<<endl;
    return 0;
}
```

Output Program :

Alamat Variabel x :0x61fe1c
Nilai Variabel x :9

b. Operator Dereference (*)

Dereference adalah suatu operator yang berfungsi menyatakan suatu variabel merupakan variabel pointer. Sama seperti operator ference, peletakan simbol operator dereference diletakan diawal variabel. Operator dereference ini akan membuat suatu variabel pointer untuk menampung alamat.

Contoh program 2 :

```cpp
#include <iostream>

using namespace std;

int main(){

int x=5;
int *y;
y = &x;

cout<<"Nilai variabel x :"<<x<<endl;
cout<<"Alamat variabel x :" <<&x<<endl;
cout<<"Isi dari variabel y :"<<y<<endl;
cout<<"Nilai yang tersimpan dalam variabel y :"<<*y<<endl;

return 0;

}
```

Output program :

Nilai variabel x :5
Alamat variabel x :0x61fe14
Isi dari variabel y :0x61fe14
Nilai yang tersimpan dalam variabel y :5

1.3 Mendeklarasikan Variabel Pointer

Suatu variabel dengan tipe data pointer dapat didefinisikan dengan 3 bentuk sebagai berikut :

tipe_data *nama_variabel
tipe_data* nama_variabel
tipe_data * nama_variabel

• **tipe_data** dapat berupa berbagai tipe seperti halnya pada pendefinisian variabel bukan pointer.
• **nama_variabel** adalah nama variabel pointer.

Contoh program 3 :

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5  int x, y;
6  int *px;
7
8  x = 77;
9  y = x;
10 px = &x;
11
12 cout << "Nilai x  : " << x << endl;
13 cout << "Nilai y  : " << y << endl;
14 cout << "Alamat x : " << &x << endl;
15 cout << "Alamat x yang disimpan pada px : " << px << endl;
16 cout << "Nilai px : " << *px << endl;
17 }
```

Output program :

Nilai x : 77
Nilai y : 77
Alamat x : 0x61fe0c
Alamat x yang disimpan pada px : 0x61fe0c

Nilai px : 77

1.4 Pointer pada Pointer

Tidak hanya menunjuk alamat dari suatu variabel, pointer juga dapat menunjuk ke pointer lainnya. Dalam pendeklarasiannya, kita tambahkan pointer dereference (*) pada variabel yang akan ditunjuk.

Contoh program 4 :

```
#include <iostream>

using namespace std;

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

x = 170;
px = &x;
ppx = &px;
cout << "Nilai x : " << x << endl;
cout << "Nilai px : " <<*px<<endl;
cout << "Nilai ppx : "<<**ppx<<endl;

return 0;

}
```

Output program :

Nilai x : 170
Nilai px : 170
Nilai ppx : 170

1.5 Pointer pada Array

Pada Array, pointer hanya perlu menunjukan alamat elemen pertama saja karena alamat array dalam memori sudah disusun secara berurutan.

Contoh program 5 :

```cpp
#include <iostream>

using namespace std;

int main() {
    int x[5];
    int *px;
    px = x; //px = &x[0]

    for (int i = 0; i < 5; i++) {
        cout << "Masukkan Nilai "<<i+1<<" : ";
        cin >> x[i];
    }
    cout << endl;
    for (int i = 0; i < 5; i++) {
        cout << "Nilai  x["<<i<<"] : "<<*px<<endl;
        cout << "Alamat x["<<i<<"] : "<<px<<endl;
        px++;
    }
}

```

Output program :

Masukkan Nilai 1 : 1
Masukkan Nilai 2 : 2
Masukkan Nilai 3 : 3
Masukkan Nilai 4 : 4
Masukkan Nilai 5 : 5

Nilai x[0] : 1
Alamat x[0] : 0x61fdf0
Nilai x[1] : 2
Alamat x[1] : 0x61fdf4
Nilai x[2] : 3
Alamat x[2] : 0x61fdf8
Nilai x[3] : 4
Alamat x[3] : 0x61fdfc
Nilai x[4] : 5
Alamat x[4] : 0x61fe00

2. Array

Tipe data array adalah tipe data terstruktur yang memungkinkan untuk menyimpan kumpulan data dengan nama yang sama dan tipe data yang sama, di mana setiap elemen data dalam array memiliki indeks yang unik untuk membedakannya dari elemen lain dalam array. Dengan menggunakan array, kita dapat menyimpan dan mengakses banyak data dengan lebih efisien daripada menyimpan setiap data secara terpisah dalam variabel-variabel yang berbeda. Array juga memudahkan kita dalam melakukan operasi matematika atau manipulasi data yang sama secara bersamaan.

2.1 Array Satu Dimensi

Array Satu dimensi 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
Nilai	11	51	23	53	68	94	35	71	47

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

Contoh program 6 :

```cpp
#include <iostream>

using namespace std;

int main() {
    int x[10] = {12,35,20,23,54,49,81,92,119,29};
    int i;
    int min = 1000; //asumsi paling minimum
    int maks = -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 maksismum : "<<maks<<endl;
    cout<<"Nilai minimum : "<<min<<endl;

}
```

Output program :

Nilai maksismum : 119
Nilai minimum : 12

Contoh program 7 :

```cpp
#include <iostream>

using namespace std;

int main()
{
    int x[10] = {12,35,20,23,54,49,81,92,119,29};
    int i,dicari;
    bool ketemu = false;

    cout<<"Bilangan yang ingin dicari : ";
    cin>>dicari;

    for(i=0;i<10; i++)
    {
        if(x[i] == dicari)
        {
            ketemu = true;
            cout<<"Bilangan ditemukan di elemen : "<<i<<endl;
        }
    }
    if(ketemu)
    {
        cout<<"Bilangan ditemukan!"<<endl;
    } else {
        cout<<"Bilangan tersebut tidak ditemukan!"<<endl;
    }
}
```

Output program :

(jika ketemu)

Bilangan yang ingin dicari : 12
Bilangan ditemukan di elemen : 0
Bilangan ditemukan!

(Jika tidak ketemu)

Bilangan yang ingin dicari : 1
Bilangan tersebut tidak ditemukan!

Latihan :

Diberikan sebuah array karakter chars. Buat sebuah fungsi compress, yang mengubah isi dari chars menjadi bentuk yang sudah terkompresi.

Kompresi dilakukan dengan menggabungkan karakter yang sama berurutan dan menghitung berapa kali karakter tersebut muncul. Jika sebuah karakter hanya muncul satu kali, maka karakter tersebut tetap harus dipertahankan tanpa diikuti oleh angka. Namun, jika sebuah karakter muncul lebih dari satu kali, maka karakter tersebut harus diikuti oleh angka yang menunjukkan jumlah kemunculannya.

Setelah array terkompresi, fungsi compress harus mengembalikan panjang array yang sudah terkompresi.

Anda harus mengubah array chars "in-place" dan mengembalikan nilai integer yang merepresentasikan panjang array terkompresi.

Anda tidak perlu mempertimbangkan setiap karakter yang muncul dalam array, karena length dari array tidak akan melebihi 10^4.

Contoh 1:

Input: chars = ["a","a","b","b","c","c","c"]

Output: 6, ["a","2","b","2","c","3"]

Penjelasan:

"aa" dapat diubah menjadi "a2". "bb" dapat diubah menjadi "b2". "ccc" dapat diubah menjadi "c3".

Array `chars` yang sudah terkompresi adalah ["a","2","b","2","c","3"].

`length` array yang sudah terkompresi adalah 6.

Contoh 2:

Input: chars = ["a"]

Output: 1, ["a"]

Penjelasan: Karena `length` array adalah 1, tidak ada yang dapat dikompresi.

Array `chars` yang sudah terkompresi adalah ["a"]. `length` array yang sudah terkompresi adalah 1.

Catatan:

Karakter yang muncul dalam array hanya berupa huruf (a-z atau A-Z).