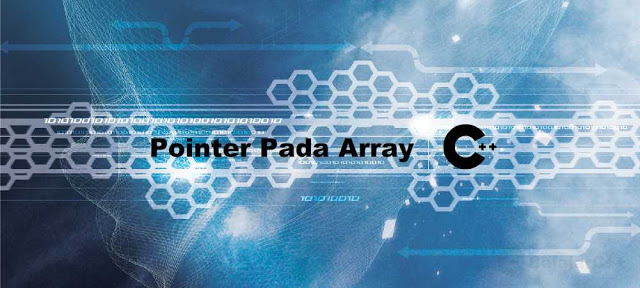
[](https://3.bp.blogspot.com/-mDREx9jtun4/VrTNFXJBdEI/AAAAAAAADEE/DqSJ2m1QucQ/s1600/pointer+pada+array.jpg)

Pointer pada Array merupakan salah satu kombinasi pemanfaatan dari dua fungsi, yaitu Pointer dan Array. Pointer adalah sebuah pointer yang berisikan alamat dari variabel lain, Suatu pointer dimaksudkan untuk menunjukan ke suatu alamat memori sehingga alamat dari suatu variabel dapat diketahui dengan mudah dan memungkinkan kita untuk mengubah nilai variabel yang ditujuk melalui variabel pointer. Dan Pengertian Array atau bisa disebut sebagai Variabel larik adalah seragkaian elemen dari jenis yang sama ditempatkan di lokasi memori yang berdekatan dengan elemen lainnya.

Pada umumnya konsep array mirip seperti konsep pointer. Konsep array adalah berdasarkan pengertian array diatas bahwa array adalah seragkaian elemen dari jenis yang sama ditempatkan di lokasi memori yang berdekatan dengan elemen lainnya , terjadi banyak loncatan dari alamat ke alamat elemen lainya atau pointer kepointer yang lain. karena secara internal array juga menyatakan alamat, yaitu pengenal array sama dengan alamat pada elemen pertama pada array.

#include <iostream>

using namespace std;

int main(){

int arrays[5]={4,5,2,6,7};

cout<<arrays<<endl;

//Menampilkan alamat dari elemen pertama

for(int i=0;i<5;i++){

cout<<(arrays+i)<<endl;

}

return 0;

}

Program diatas adalah contoh program gambaran dari konsep array. Program diatas akan menampilkan alamat “arrays” bukan menampilkan nilai isi dari alamat tersebut, jika kita memanggilnya tanpa pengarah index “[index]”. Itu cukup menjelaskan bahwa array masih berhubungan dengan pointer, semua elemen di dalam array saling berhubungan seperti halnya pointer. dan jika kita memanggil identifier array tersebut tanpa menyebut jumlah index, maka otomatis akan menampilkan alamat dari elemen pertama pada array tersebut.

Bentuk Penulisan :

int array

int \*pointer = array

//atau

pointer = array

//tidak menggunakan tanda “&” seperti biasanya. Karena pemanggilan tersebut sudah berbentuk alamat.

Tanda penentu index atau tanda kurung pada array “[]” untuk menentukan elemen dari array fungsinya sama seperti operator deferencing ‘\*’ dan dikenal sebagai Operator Offset.

Contoh Program :

#include <iostream>

using namespace std;

int main(){

int arrays[6]={4,5,2,6,7,8};

int \*pointer;

pointer=arrays;

//tidak perlu menggunakan tanda & karena

cout<<arrays<<endl<<endl;

//sudah Mengandung alamat dari elemen pertama

//Cara pertama

pointer=arrays;

/\*

pemberian tugas ini akan selalu di ulang tapi bukan

berarti harus diikuti / hanya sebagai contoh. ini hanya

untuk menetralisir pointer yang akan mencontohkan beberapa

cara yang berbeda untuk pemberian nilai pada tutorial ini.

\*/

pointer[5]=60;

//Cara Kedua

pointer=arrays;

\*pointer = 10;

//Cara Ketiga

pointer++;

\*pointer = 20;

//Cara Keempat

pointer = &arrays[2];

\*pointer = 30;

//Cara Kelima

pointer = arrays+3;

\*pointer = 40;

//Cara Keenam

pointer = arrays;

\*(pointer+4) = 50;

//Output

for(int i=0;i<6;i++){

cout<<"Memanggil Melalui Arrays "<<(arrays+i)<<" = "<<arrays[i]<<endl;

cout<<"Memanggil Melalui Pointer "<<(pointer+i)<<" = "<<pointer[i]<<endl;

}

return 0;

}

Di atas adalah contoh dari penggunaan pointer pada array, untuk pemberi penugasan biasanya kita diharuskan member tanda ‘&’ pada awal identifier, untuk ini kita cukup menggunakan nama identifier “pointer=arrays” tanpa menggunakan tanda ‘&’, karena array pada dasarnya adalah urutan elemen yang saling berekatan dan mirip seperti pointer, dan bisa kita buktikan seperti baris berikutnya, pengeluaran variabel tersebut tanpa memberi nomer index pada array tersebut maka hasilnya adalah alamat dari elemen pertama array. Jadi kita tidak perlu menggunakan tanda ‘&’.

Ada 6 cara yang bisa kita lakukan untuk mengakses variabel yang ditunjuk tersebut. Cara pertama adalah “pointer[5]=60” sama seperti pemberi tugas yang umum kita gunakan pada array. Sesungguhnya tanda “[]” itu sifatnya sama seperti operator dereference ‘\*’.

Cara kedua adalah “\*pointer = 10” ini akan memberikan angka 10 pada elemen pertama pada arrays. Dan cara ketiga “pointer++;\*pointer = 20;” masih sama seperti cara kedua, tapi di cara ketiga terdapat “pointer++” yang akan menambah nilai 1 pada alamat pointer sama seperti kita menaruh angka “[1]” yang berarti akan menunjuk elemen kedua dari array. Ingat bahwa penghitungan index selalu dimulai dari 0.

Cara ke empat “pointer = &arrays[2];\*pointer = 30;” cara ini menggunakan cara yang sedikit berbeda, petama “pointer = &arrays[2]” kita memeri tugas pada “pointer” untuk menunjuk ke “arrays[2]” array pada elemen ke tiga. Sebelumnya penulis mengatakan bahwa kita tidak perlu menggunakan tanda ‘&’ tapi pada kali ini kita diharuskan menggunakan karena “arrays[2]” mengandung nilai sama seperti pernyataan “\*(arrays+2)”, sedangkan yang dibutuhkan pointer adalah alamat maka kita diharuskan menggunakan tanda ‘&’. Dan pada baris berikutnya “\*pointer = 30” pointer yang telah menunjuk ke elemen ke tiga array akan diberi nilai 30.

Cara kelima “pointer = arrays+3;\*pointer = 40;” pointer diberikan tugas untuk menunjuk alamat array pada elemen ke empat, dan di sini tanda ‘&’ tidak berlaku karena penyebutan tersebut sudah mengarah ke alamat. Dan baris berikutnya memberi nilai 40 pada elemen ke empat array yang telah ditunjuk oleh pointer melalui pointer “\*pointer = 40”.

Cara ke enam “\*(pointer+4) = 50” akan memberikan nilai 50 pada elemen ke 5 array. “pointer+4” mengarahkan pointer ke elemen 5 dan masih dalam bentuk alamat. Di luar tanda kurung kita beri tanda ‘\*’ yang akan mengubah alamat itu menjadi isi nilai dari alamat tersebut, dan karena lokasi tempat sudah kita tentukan untuk pemberian nilai, maka kita dapat menambahkan nilai seperti nilai “50”.

Dan pada bagian pengerluaran kita juga bisa menggunakan cara-cara di atas. Tapi pada contoh program diatas penulis hanya contohkan 2 cara yaitu melalui array langsung dan melalui pointer. kita juga bisa menggunakan cara-cara di atas untuk pengeluaran.

Pada cara-cara di atas tidak hanya bekerja pada “pointer ke array” tetapi juga bisa bekerja pada penggunaan “array”.

Cukup sekian dari belajarcpp.com semoga artikel ini bisa bermanfaat untuk kita semua. Mohon maaf atas kekuranganya dan Terima Kasih atas dukungan dan kunjungan anda ke BelajarCPP. Have a nice day.

Last update : 5 Februari 2016