Pointer di C++

Definisi Pointer

Pointer adalah built-in type di C dan C++, dimana C++ mengambil konsep pointer dari C. Pointer sebenarnya sangat terkait dengan "Abstract C Machine", yaitu model mesin abstrak dimana program C bekerja. Abstract C Machine adalah mesin abstrak dimana mesin tersebut memiliki prosesor untuk menginterpretasikan stream of instruction, dan addressable memory yang terbagi kedalam 3 bagian : automatic memory, static memory dan free memory. Addressable memory adalah memory yang konten-nya dapat diambil jika diketahui alamatnya. Lebih jauh lagi, terdapat asumsi bahwa konten memori dapat di ambil dengan waktu konstan, tidak peduli berapa nilai alamat.Hal ini disebut dengan Random Access Memory.

Menurut Frieyadie. 2007. Edisi Revisi Buku Pemrograman C++ dengan Borland C++ 5.02.

Pointer adalah sebuah variabel yang berisi alamat lain. Suatu pointer dimaksudkan untuk menunjukan ke suatu alamat memori sehingga alamat dari suatu variabel dapat diketahui dengan mudah.

Arti pointer dalam bahasa sehari-hari adalah putunjuk atau bisa di bilang penentu atau pointer secara sederhana bisa diartikan sebagai tipe data yang nilainya mengarah pada nilai yang terdapat pada sebuah area memori (alamat memori).namun dalam Dalam bahasa C, Pointer bisa berfungsi sebagai variabel arrayberarti pointer sebagai penunjuk elemen array ke-0 dalam variabel C.

Kegunaan Pointer Di C++ :

Kegunaan pointer yang utama adalah untuk menyimpan alamat memori dari sebuah variabel (data type atau object dari class). Selain menyimpan alamat dari variabel, pointer juga dapat digunakan untuk menyimpan alamat dari sebuah fungsi (function pointer).

Function pointer telah digunakan sejak dikenalkannya bahasa C,dan banyak digunakan untuk sebuah fungsi callback atau untuk meningkatkan readability dari sebuah code

Anda dapat memperlakukan function pointer seperti pointer biasa (pointer ke datatype/object), anda dapat menyimpan,mengirimkan, merubah address, atau meng-evaluasi address dari pointer ke fungsi ini dengan sifat tambahan anda dapat memanggil fungsi yang ditunjuk oleh function pointer.

Pointer bias juga berguna untuk :

1. Mengirimkan “Parameter yang berupa variabel” ke dalam fungsi, artinya nilai variabel bisa diubah di dalam fungsi.

2. Untuk membuat variabel DINAMIS (Bukan variabel Statis)

penggunaan function pointer pada C++ dibatasi, yaitu function pointer tidak boleh menunjuk pada function yang berada dalam sebuah class (class member function) kecuali function tersebut berjenis static.

Mengapa sih harus menggunakan Pointer dalam bahasa C++ ?

Karena pointer dapat meningkatkan kinerja untuk operasi yang dilakukan secara berulang. Dengan syarat Kalau mendeklarasikan pointer kedalam array, tidak boleh menggunakan tanda bintang.

Pointer juga di gunakan untuk mengalokasikan tempat pada memori secara dinamis yakni dapat diubah-ubah alokasi tempatnnya pada memori data yang dimasukkan sebagai nilai pointer akan selalu tersimpan sehingga diperlukan penghapusan yang tujuannya untuk mengosongkan memori, perintah yang digunakan untuk menghapus memori adalah delete [] nama variable.

Penggunaan Awal Pointer

Jika variabel merupakan isi memori, dan untuk mengakses isi memori tersebut diperlukan address, lalu bagaimana cara kita mengetahui alamat dari suatu variabel ? Jawabannya adalah : untuk kebanyakan kasus kita sama sekali tidak perlu tahu alamat dari sebuah variabel. Untuk mengakses sebuah variabel kita hanya perlu nama dari variabel tersebut. Tugas kompiler lah yang mentranslasikan nama ke alamat mesin yang diperlukan oleh komputer.

Akan tetapi terdapat beberapa kasus dimana kita tidak mungkin memberi nama pada sebuah entitas di program kita. Hal ini terjadi terutama saat kita menggunakan data struktur dinamis seperti linked list, resizeable array, tree dan lain sebagainya. Hal ini karena kita tidak mungkin memberi nama terhadap entitas yang mungkin ada atau tidak ada. Struktur seperti linked list hampir mustahil dibuat tanpa pointer tanpa harus mendefinisikan LISP-like list.

1. Penggunaan Pointer Sebagai Moniker.

Istilah moniker di sini berarti sesuatu yang menunjuk atau mengacu kepada entitas lain. Istilah moniker ini bukanlah istilah standard dan lazim , tetapi sesuatu yang saya pilih impromptu untuk membedakan dengan pointer atau reference yang sudah memiliki arti tersendiri.

Penggunaan lain pointer sebagai moniker adalah untuk mengatasi kelemahan bahasa C awal : Dahulu fungsi - fungsi di C hanya mengerti pass by value. Pointer digunakan untuk mengemulasi pass by reference karena pointer berisi alamat ke objek lain, sehingga fungsi tersebut dapat mengubah objek tersebut dengan memanipulasi pointer.

2. Operasi pointer arithmatic lain juga didefinisikan untuk pointer.

Yang menarik adalah increment dan decrement. programmer dapat memeriksa semua elemen dalam array dengan cara menginkremen pointer dari pointer penunjuk elemen pertama. Tentu saja hal yang sama dapat dilakukan dengan indexing biasa, ar[idx], akan tetapi dengan operasi pointer bisa lebih efisien. Alasannya terletak pada bagaimana cara komputer membaca data di ar[idx]. Untuk mesin yang memiliki indexed addressing hal ini cukup sederhana dan efisien (ar jadi base, idx jadi index, fetching cukup 1 instruksi mov). Tetapi untuk mesin yang tidak memiliki indexed addressing, akan ada operasi ADD antara ar dan idx, lalu simpan hasilnya ke suatu tempat (register), lalu baru mov. Kadang – kadang register tersebut digunakan untuk operasi ADD sehingga terdapat beberapa mov untuk menyimpan state. Akan tetapi jika menggunakan pointer arithmatic, cukup meng-increase nilai yang

sudah ada di register, lalu mov. Tentu saja instruksi di dalam loop juga mempengaruhi efisiensi ini, tetapi untuk mesin yang mendukung operasi increment langsung, iterasi lewat pointer biasanya lebih efisien.

3. Penggunaan Pointer Sebagai Iterator.

Nama iterator diambil dari STL, dan iterator di STL adalah abstraksi dari pointer. Yang menakjubkan adalah konsep iterator, yang digeneralisasi dari pointer, adalah konsep yang cukup powerful untuk merepresentasikan semua algoritma yang bekerja untuk linear container (linear container adalah semua container yang memiliki iterator yang menunjuk pada elemen pertama, memiliki iterator yang menunjuk pada elemen one-past-end, dan semua elemen dapat dicapai dengan melakukan operasi incremen dari iterator penunjuk elemen pertama sebanyak yang diperlukan. Contoh linear container adalah array, vector, linked – list, dan deque. contoh yang bukan linear container adalah graph dan forest.).

Ketiga fungsi pointer di atas memerlukan operasi yang berbeda-beda. Contohnya jika pointer berfungsi sebagai moniker, operasi yang sangat diperlukan adalah fungsi malloc, calloc, free, new, delete, operator ->, operator * dan operator &. sebagai moniker pointer tidak memerlukan konvertability ke integer dan operasi pointer arihmatic (walaupun ada trik mengakses field struct dari pointer dengan mengcast pointer to struct menjadi char*, tambahkan offsetnya, lalu baca dengan operator * dan di cast ke tipe field tersebut. trik ini sangat berbahaya dan sebaiknya tidak dipakai). Jika pointer berfungsi sebagai iterator, operasi pointer arithmatic adalah esensial. Tetapi operasi new dan delete sama

sekali tidak di perlukan (kecuali untuk array of pointer). bottom line is: you do not do memory management via iterator.

Sifat konvertibilitas antara integer dan pointer hanya diperlukan jika pointer tersebut dipakai sebagai abstraksi fixed address. Dua fungsi lain tidak memerlukan sifat ini.

Atau bisa juga pointer berguna untuk :

1. Mengirimkan “Parameter yang berupa variabel” ke dalam fungsi, artinya nilaivariabel bisa diubah di dalam fungsi
2. penggunaan function pointer pada C++ dibatasi, yaitu function pointer tidak boleh menunjuk pada function yang berada dalam sebuah class (class member function) kecuali function tersebut berjenis static.

Mengapa sih harus menggunakan POINTER dalam bahasa C++ ...??? Karena dapat meningkatkan kinerja untuk operasi yang dilakukan secara berulang.

dengan syarat Kalau mendeklarasikan pointer kedalam array, tidak boleh menggunakan tanda bintang

Operator yang digunakan untuk pointer adalah tanda “*” dan tanda “&”.

tanda “*” ini berfungsi sebagai penunjuk nilai dari suatu pointer sedangkan tanda“&” digunakan sebagai penunjuk alamat dari suatu variable.

Pointer di gunakan untuk mengalokasikan tempat pada memori secara dinamis yakni dapat diubah-ubah alokasi tempatnnya pada memori.

data yang dimasukkan sebagai nilai pointer akan selalu tersimpan sehingga diperlukan penghapusan yang tujuannya untuk mengosongkan memori,perintah yang digunakan untuk menghapus memori adalah delete [] nama variable.

*Operasi pointer dapat diterapkan pada tipe data Array,tipe data Struck,

*Pointer dapat digunakan untuk mengakses elemen array

*Pointer dapat menunjuk alamat pointer lain.

Tipe Data Pointer :

Dimana Tipe_data merupakan tipe dari data yang ditunjuk, bukan tipe dari pointer-nya. Contoh :

1. Mensubstitusikan address sebuah variabel ke pointer dengan memakai address operator &

   int x;

   int *ptr;

   ptr = &x;

2. Mensubstitusikan address awal sebuah array ke pointer

   char t[5];

   char *ptr;

   ptr = t;

3. Mensubstitusikan address salah satu elemen array dengan address operator

   char t[5] ;

   char *ptr;

   ptr = &t[3];

4. Mensubstitusikan address awal character string ke pointer char

   char *ptr;

   ptr = "jakarta"

5. Mensubstitusikan NULL pada pointer. NULL ada pointer kosong, menunjukkan suatu status dimana pointer itu belum diinisialisasikan dengan sebuah address tertentu.

6. Memakai fungsi MALLOC.

Operator Pointer

Ada 2 operator pointer yang dikenal secara luas, yaitu operator & dan operator *.

Operator &

Operator & merupakan operator alamat. Pada saat pendeklarasian variable, user tidak diharuskan menentukan lokasi sesungguhnya pada memory, hal ini akan dilakukan secara otomatis oleh kompiler dan operating sysem pada saat run-time.

Jika ingin mengetahui dimana suatu variable akan disimpan, dapat dilakukan dengan memberikan tanda ampersand (&) didepan variable , yang berarti "address of". Contoh :

ted = &andy;

Penulisan tersebut berarti akan memberikan variable ted alamat dari variable andy. Karena variabel andy diberi awalan karakter ampersand (&), maka yang menjadi pokok disini adalah alamat dalam memory, bukan isi variable.

Misalkan andy diletakkan pada alamat 1776 kemudian dituliskan instruksi sbb :

```
andy = 25;
fred = andy;
ted = &andy;
```

Operator *

perator * merupakan operator reference. Dengan menggunakan pointer,kita dapat mengakses nilai yang tersimpan secara langsung dengan memberikan awalan operator asterisk (*) pada identifier pointer, yang berarti "value pointed by". Contoh :

beth = *ted;

(dapat dikatakan:"beth sama dengan nilai yang ditunjuk oleh ted") beth = 25, karena ted dialamat 1776, dan nilai yang berada pada alamat 1776 adalah 25.

Ekspresi dibawah ini semuanya benar, perhatikan :

```
andy = 25;
&andy = 1776;
```

```
ted = 1776;
*ted = 25;
```

Ekspresi pertama merupakan assignation bahwa andy = 25;. Kedua, menggunakan operator alamat (address/derefence operator (&)), sehingga akan mengembalikan alamat dari variabel andy. Ketiga bernilai benar karena assignation untuk ted adalah ted = &andy;. Keempat menggunakan reference operator (*) yang berarti nilai yang ada pada alamat yang ditunjuk oleh ted, yaitu 25. Maka ekspresi dibawah ini pun akan bernilai benar :

```
*ted = andy;
```

Deklarasi Pointer

Seperti halnya variabel lain, variabel pointer juga harus dideklarasikan terlebih dahulu sebelum digunakan. Bentuk umum deklarasi pointer adalah :

Dimana Tipe_data

merupakan tipe dari data yang ditunjuk, bukan tipe dari pointer-nya. Contoh :

1. Mensubstitusikan address sebuah variabel ke pointer dengan memakai address operator &

```
int     x;
int     *ptr;
ptr = &x;
```

2. Mensubstitusikan address awal sebuah array ke pointer

```
char     t[5];
char    *ptr;
ptr =     t;
```

3. Mensubstitusikan address salah satu elemen array dengan address operator

```
char     t[5] ;
char    *ptr;
ptr =     &t[3];
```

4.Mensubstitusikan address awal character string ke pointer char

```
char    *ptr;
ptr =     "jakarta"
```

5.Mensubstitusikan NULL pada pointer. NULL ada pointer kosong, menunjukkan suatu status dimana pointer itu belum diinisialisasikan dengan sebuah address tertentu.
6.Memakai fungsi MALLOC.

Macam – macam Pointer

1.Pointer Bertipe Void

Pada C++ terdapat pointer yang dapat menunjuk ke tipe data apapun, po¬inter semacam ini dideklarasikan dengan tipe void sehingga sering dikenal de¬ngan is¬ti¬lah Void Pointer. Berikut ini contoh listing program yang meng¬gu¬na¬kan void po¬inter.

2.Pointer Aritmetika

Elemen-elemen array biasanya diakses melalui indeksnya, sebenarnya ada cara lain yang lebih efisien, yaitu dengan menggunakan pointer. Pointer sema¬cam ini disebut dengan istilah pointer aritmetika.

Konsep dasar dari pointer arit¬me¬tika ini adalah melakukan operasi aritmetika terhadap variabel yang bertipe po¬int¬er.

3.Pointer NULL

Pada saat program dijalankan, pointer akan menunjuk ke alamat acak pa¬da memori, sehingga diperlukan inisialisasi agar hal tersebut tidak terjadi. Dalam C++ terdapat sebuah cara untuk membuat pointer

tidak menunjuk ke alamat ma¬na¬pun, yaitu dengan mengisikan
ponter tersebut dengan nilai NULL. Karena hal inilah maka pointer tersebut sering dinamakan pointer NULL (NULL Pointer).Se¬bagai contoh kita mempunyai pointer p, dan kita ingin melakukan inisialisasi pada pointer tersebut dengan nilai NULL
maka sintaknya adalah sebagai berikut. P=NULL;

Operasi-operasi Pointer

1.Operasi Penugasan

Suatu variabel pointer seperti halnya variabel yang lain, juga bisa mengalami operasi penugasan.

Nilai sari suatu variabel pointer dapat disalin ke variabel pointer yang lain.

Contoh Program :

```
#include "stdio.h"
#include "conio.h"

void main ()
{
        int nilai [3], *penunjuk;
        clrscr();
        nilai[0] = 125;
        nilai[1] = 345;
        nilai[2] = 750;
        penunjuk = &nilai[0];
        printf("Nilai %i ada di alamat memori %p\n", *penunjuk,penunjuk);
        printf("Nilai %i ada di alamat memori %p\n", *(penunjuk + 1),penunjuk + 1);
        printf("Nilai %i ada di alamat memori %p\n", *(penunjuk + 2),penunjuk + 2);
        getch ();
}
```

2.**Operasi Aritmatika**

Suatu variabel pointer hanya dapat dilakukan operasi aritmatika dengan nilai integer saja.

Operasi yang biasa

dilakukan adalah operasi penambahan dan pengurangan.

Contoh program :

```
#include “stdio.h”
#include “conio.h”

void main ()
{
        int nilai [3], *penunjuk;
        clrscr();
        nilai[0] = 125;
        nilai[1] = 345;
        nilai[2] = 750;
        penunjuk = &nilai[0];
        printf(“Nilai %i ada di alamat memori %p\n”, *penunjuk,penunjuk);
        printf(“Nilai %i ada di alamat memori %p\n”, *(penunjuk + 1),penunjuk + 1);
        printf(“Nilai %i ada di alamat memori %p\n”, *(penunjuk + 2),penunjuk + 2);
        getch ();
}
```

3. Operasi Logika

Operasi logika juga dapat dilakukan pada sebuah variabel pointer.

Contoh programnya :

```
#include “stdio.h”
#include “conio.h”

void main ()
{
        int a = 100, b = 200, *pa, *pb;
        clrscr ();
        pa = &a;
        pb = &b;
        if (pa < pb)
            printf(“pa menunjuk ke memori lebih rendah dari pb\n”);
        if (pa == pb)
            printf(“pa menunjuk ke memori yang sama dengan pb\n”);
        if (pa > pb)
            printf(“pa menunjuk ke memori yang lebih tinggi dari pb\n”);
        getch ();
```

Bahaya Pointer

1.Bahaya yang mungkin ada dengan pointer sebagai moniker: memory leak, double delete, invalid memory access.

Semuanya dapat dihindari dengan ownership analysis yang bagus (pada setiap saat, harus diketahui pihak mana yang bertanggung jawab mendelete sebuah object). Jika hal ini sulit dilakukan, misalnya karena shared ownership, anda dapat menggunakan smart pointer atau garbage collector

2.Bahaya yang mungkin ada dengan pointer sebagai iterator: array out of bound. Salah satu cara yang efektif menghindari hal ini adalah dengan menggunakan standard algorithm.

3.Bahaya yang mungkin ada dengan pointer sebagai abstraksi fixed memory : Tidak tahu, tetapi ini bukan mainan sembarang programmer.

Bahasa Pemrograman tanpa pointer ?

1.Semua Bahasa pemrograman Fungsional, terutama yang murni , tidak mengenal pointer atau memerlukan pointer.

Akan tetapi bahasa ini menggunakan model komputasi yang jauh berbeda, bukan abstract C machine.

2.Beberapa bahasa pemrograman dengan reference semantik dapat mengklaim mereka tidak memiliki pointer,
akan tetapi setiap variabel sebenarnya adalah pointer. Secara fisik mungkin reference tidak memiliki struktur
seperti pointer (biasanya merupakan data struktur yang lebih kompleks sehingga lebih friendly terhadap garbage collector) tapi reference tersebut memiliki fungsi
yang mirip dengan pointer di C atau C++.

Ada yang bilang bahwa reference dalam bahasa - bahasa ini menyebabkan optimasi lebih mudah karena tidak menyebabkan aliasing, tetapi optimasi tersebut juga mungkin dilakukan di C dan C++ (dengan restrict pointer,
sayangnya belum merupakan bagian dari standard C++).