**MAKALAH**

**TENTANG POINTER**

Disusun oleh :

|  |  |
| --- | --- |
| Nama | **:** L Hafid Alkhair |
| NIM | **:** 2023903430060 |
| Kelas | **:** TRKJ 1.C |
| Jurusan | **:** Teknologi Informasi dan Komputer |
| Program Studi | **:** Teknologi Rekayasa Komputer Jaringan |
| Dosen Pengajar | **:** Indrawati, SST. MT |



**JURUSAN TEKNOLOGI INFORMASI KOMPUTER**

**PRODI TEKNOLOGI REKAYASA KOMPUTER JARINGAN**

**POLITEKNIK NEGERI LHOKSEUMAWE**

**TAHUN AJARAN 2023/2024**

**DAFTAR ISI**

[**BAB I** 3](#_Toc147572181)

[**PEMBAHASAN** 3](#_Toc147572182)

[**A.** **PENGERTIAN POINTER** 3](#_Toc147572183)

[**1.** **Mengakses data dengan Pointer** 8](#_Toc147572184)

[**2.** **Pointer Menuju Objek Lain** 11](#_Toc147572185)

[**3.** **Mendeklarasikan Variable Pointer** 11](#_Toc147572186)

[**4.** **Mengatur Pointer Agar Menunjuk ke Variable Lain.** 13](#_Toc147572187)

[**5.** **Mengakses Isi Suatu Variable Melalui Pointer** 14](#_Toc147572188)

[**6.** **Tipe Variable Pointer Dan Tipe Objek yang ditunjuk** 16](#_Toc147572189)

[**7.** **Mengubah Isi Suatu variable Melalui Pointer.** 18](#_Toc147572190)

[**8.** **Pointer Dan Array.** 20](#_Toc147572191)

[**9.** **Pointer dan String** 22](#_Toc147572192)

[**10 Inisialiasi Array Pointer** 23](#_Toc147572193)

[**B.** **Pointer Menunjuk Pointer** 24](#_Toc147572194)

[**C.** **Pointer Dan Fungsi** 26](#_Toc147572195)

[**D.** **Pointer Sebagai Keluaran Fungsi** 27](#_Toc147572196)

[**BAB II** 30](#_Toc147572197)

[**PENUTUP** 30](#_Toc147572198)

[**A.** **KESIMPULAN** 30](#_Toc147572199)

[**DAFTAR PUSTAKA** 32](#_Toc147572200)

# **BAB I**

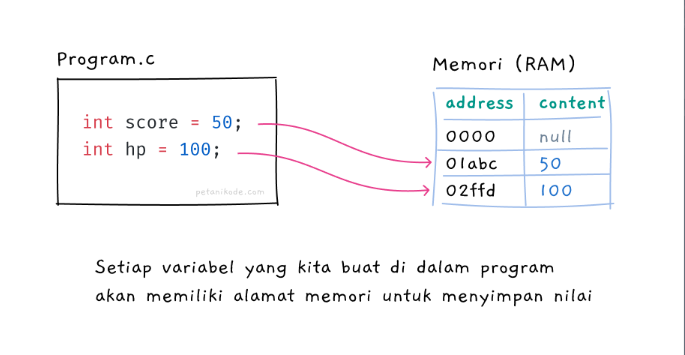
# **PEMBAHASAN**

# **PENGERTIAN POINTER**

Apa itu Pointer?

Setiap variabel yang kita buat pada program akan memiliki alamat memori. Alamat memori berfungsi untuk menentukan lokasi penyimpanan data pada memori (RAM). Kadang alamat memori ini disebut reference atau referensi.

Coba perhatikan gambar ini:

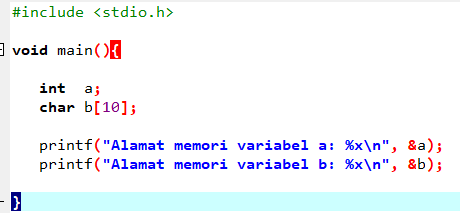


Pada gambar ini, kita membuat dua variabel.. yakni score dan hp.

Kedua variabel ini punya alamat memori masing-masing.

Variabel score alamat memorinya adalah 01abc, sedangkan hp alamat memorinya 02ffd. Selama sebuah alamat masih kosong.. maka alamat itu yang akan dipilih. Ya, pemilihan alamat memori ini, dilakukan secara acak. Inilah mengapa memori ini di sebut RAM (Random Access Memory). Intinya, setiap kita membuat variabel pasti akan punya alamat memori. Kalau tidak percaya, kamu bisa buktikan sendiri dengan menggunakan simbol & (ampersand).

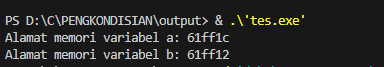
Contoh:



Pada program ini, kita menggunakan simbol & untuk mengambil alamat memori dari variabel a dan b.

Lalu menggunakan format specifier %x untuk menampilkannya dalam bilangan heksadesimal.

Hasilnya:



Lalu apa hubungannya alamat memori dengan pointer?

Pointer adalah sebuah variabel khusus yang berisi alamat memori. Pointer nantinya akan bisa mengakses data yang ada di suatu alamat memori.1

Kata kunci yang perlu kamu ingat:

**“Pointer berisi alamat memori”**

Cara Membuat Pointer

Pointer dibuat dengan menambahkan simbol \* (asterik) di depan namanya, kemudian diisi dengan alamat memori yang akan digunakan sebagai referensi.

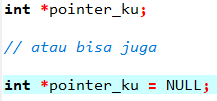
Contoh:



Maka \*pointer1 akan bisa mengakses data yang ada pada alamat memori 00001. Dengan kata lain, si \*pointer1 akan menggunakan alamat 00001 sebagai referensinya.

Kita juga bisa membuat pointer tanpa harus mengisinya langsung dengan alamat memori.

Contoh:



Maka \*pointer\_ku akan menggunakan alamat memori 00000, alamat memori ini khusus untuk menyimpan data null atau data kosong.

**Sekarang masalahnya:**

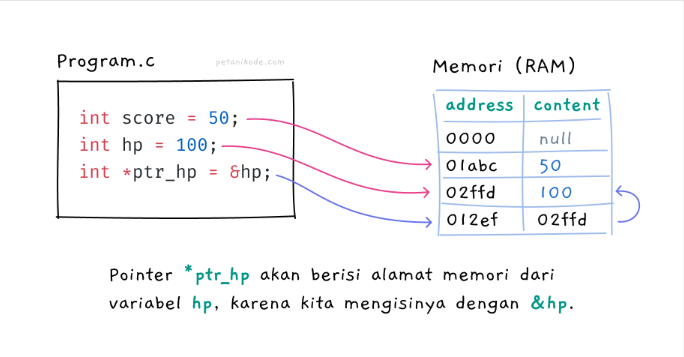
Karena kita tidak bisa lihat daftar alamat memori secara langsung, kita akan kesulitan memberikan referensi alamat memori untuk pointer.

Belum lagi.. beda komputer beda juga alamat memorinya. Ada yang 8 bit, ada juga yang 16, 32, dan sebagainya.

**Solusinya:**

Kita harus mengambil alamat memori dari variabel yang lain. Dengan menggunakan simbol &.

Coba perhatikan gambar ini:



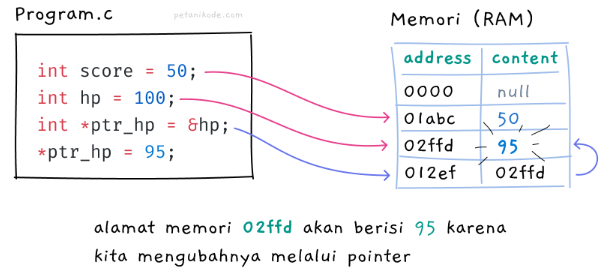
Pada gambar ini, kita membuat pointer dengan nama \*ptr\_hp\* dengan isi alamat memori dari variabel hp.

Pointer \*ptr\_hp akan bisa mengakses nilai pada alamat memori 02ffd yang mana ini adalah alamat memori dari variabel hp.

Jika kita ingin mengubah nilai pada alamat memori tersebut, maka kita bisa gunakan pointer \*ptr\_hp seperti ini:

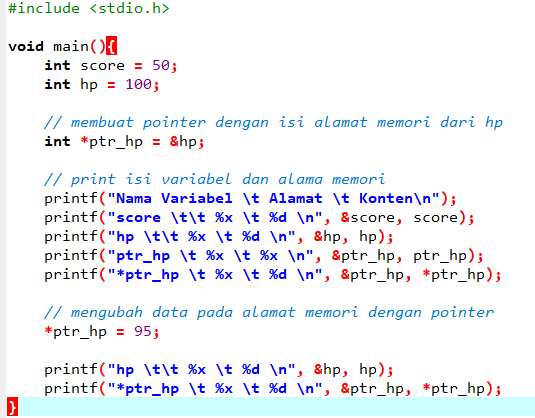


Maka, sekarang alamat memori 02ffd akan berisi 95 begitu juga dengan variabel hp.



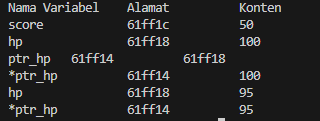
## **Mengakses data dengan Pointer**

Buatlah program baru dengan nama contoh\_pointer.c, kemudian isi dengan kode berikut:



Setelah itu, coba compile dan jalankan.

Maka hasil:



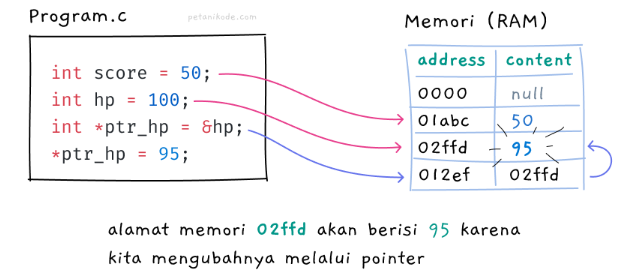
Pointer \*ptr\_hp berhasil mengubah nilai pada alamat d57ba6c menjadi 95.

Saat menggunakan pointer, kita menggunakan tanda \* di depan nama pointer untuk mengakses nilai pada alamat memori. Jika tidak menggunakan tanda ini, maka kita akan mendapatkan alamat memori yang di-pointing.



pointer juga punya alamat memorinya sendiri.

Pada contoh di atas, alamat memori dari pointer \*ptr\_hp adalah d57ba70. Mungkin di komputermu akan berbeda, silahkan di cek sendiri.

Jika kamu perhatikan gambar ini:

Alamat memori yang dipakai \*ptr\_hp adalah 012ef dengan isi alamat memori 02ffd.

Sekarang pertanyaanya:

Kalau kita menggunakan pointer, bukankah ini akan boros memori? Krena kita harus mengalokasikan alamat memori untuk si pointernya juga.

Jika kita bisa menggunakan variabel biasa, ngapain pakai pointer?

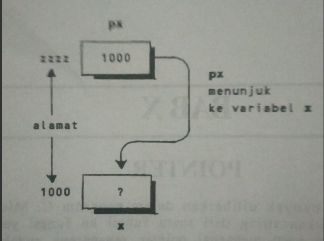
Penggunaan pointer sebenarnya opsional, kamu boleh pakai.. boleh juga tidak.

Namun..

Pada kondisi tertentu, penggunaan pointer lebih optimal.

## **Pointer Menuju Objek Lain**

Variable pointer sering dikatakan sebagai variable yang menunjukan ke objek lain. Pada kenyataan yang sebenarnya, variable pointer ( Atau di singakat menjadi pointer) berisi Alamat dari suatu objek lain ( Yaitu objek yang dikatakan ditunjuk oleh pointer). Sebagai contoh, px adalah pointer dan x adalah variable yang di tunjukan oleh px. Kalau x berada pada Alamat memori ( Alamat awal ) 1000, maka px akan berisi 1000.



## **Mendeklarasikan Variable Pointer**

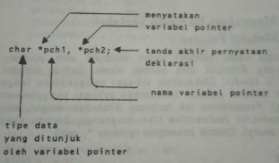
Suatu variable pointer dideklarasikan dengan bentuk sebagai berikut :



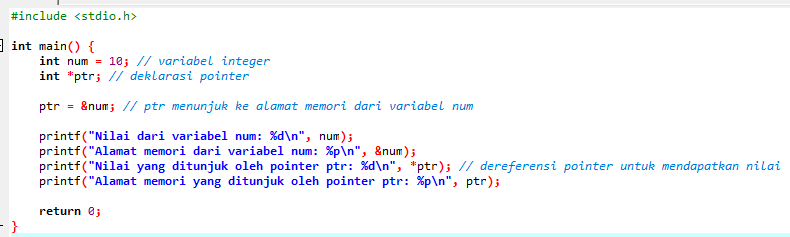
Adapun ***Nama\_variable*** adalah nama dari variable pointer



Contoh pertama meyatakan bahwa px adalah variable pointer yang menunjukan ke suatu dat bertipe int. Pada contoh kedua, masing- masing pch1 dan pch2 adalah variable pointer yang menunjukan ke data bertipe **char.**



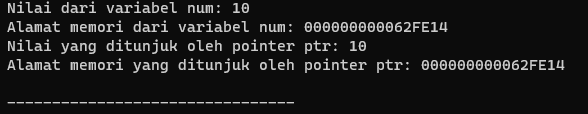
Contoh program nya



Dalam program ini, kita mendeklarasikan sebuah variabel integer num dengan nilai 10. Kemudian, kita mendeklarasikan sebuah pointer ptr yang akan menunjuk ke alamat memori dari variabel num. Operator & digunakan untuk mendapatkan alamat memori dari variabel num, dan nilai alamat memori tersebut disimpan dalam pointer ptr.

Ketika kita ingin mendapatkan nilai yang disimpan di alamat memori yang ditunjuk oleh pointer, kita menggunakan operator dereferensi (\*ptr). Pada contoh di atas, output dari program akan menunjukkan nilai dari variabel num, alamat memori variabel num, nilai yang ditunjuk oleh pointer ptr, dan alamat memori yang ditunjuk oleh pointer ptr.

Harap dicatat bahwa %p digunakan untuk mencetak nilai pointer dalam bentuk hexadecimal pada fungsi printf().

Hasil programnya

## **Mengatur Pointer Agar Menunjuk ke Variable Lain.**

Agar Satu Pointer menunjuk ke variable lain, mula-mula pointer harus diisi dengan Alamat dari variable yang akan ditunjukan. Utnuk menyatakan Alamat dari suatu variable, operator & ( operator Alamat, yang bersifat **unary**) bisa di pergunakan, dengan cara menepatkan operator di depan nama variable. Sebagai contoh, apabila x dideklarasikan sebagai variable bertipe int, maka “**&x**

Berarti alamat dari variable “x”. Adapun contoh pemberian alamat ke suatu variable pointer **px** ( yang dideklarasikan sebagai pointer yang menunjukan ke data bertipe **int** ) yaitu **px = &x;**

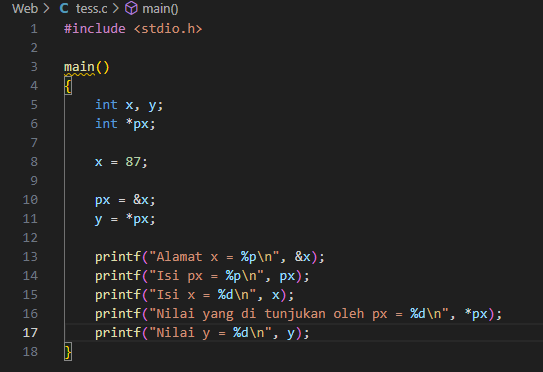
Pernyataan diatas berarti bahwa **px** diberi nilai berupa Alamat dari variable x. Setelah pernyataan tersebut diesekusi berulah dapat dikatakan bahwa **px** menunjuk ke variable x.

## **Mengakses Isi Suatu Variable Melalui Pointer**

Kalau suatu variable sudah ditunjukan oleh pointer, variable yang ditunjukan oleh pointer dapat diakese melalui variable itu sendiri( dikatakan sebagai pengakses langsung) ataupun melalui pointer ( dikatakan sebagai pengakses tak langsung ). Pengakses tak langsung dilakukan dengan menggunakan operator ***indirection*** ( tak-langsung ), berupa symbol \* ( besifat Unary ). Contoh penerapan operator \*, yaitu : ***\*px*** yang menyatakan “isi atau nilai variable/data yang ditunjukan oleh pointer ***px”.*** Sebagai contoh jika y bertipe int, maka sesudah dua pernyataan berikut .



Y akan berisi nilai yang sesuai dengan nilai x. Untuk lebih jelasnya perhatikan program berikut :



Pada program diatas, dua pernyataan

**Px = &x;**

**Y = \*px;**

Sebenarnya dapat digantikan dengan sebuah pernyataan berupa

**Y = x;**

Seandainya pada program di atas tak terdapat pernyataan

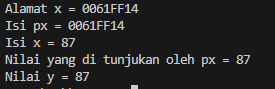
**Px = &x;**

Namun terdapat pernytaan

**Y = \*px;**

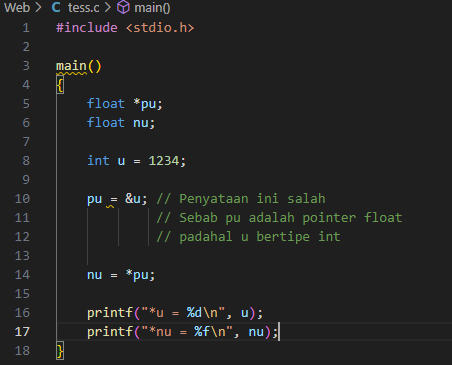
maka, y tidaklah berisi nilai x, sebab px belum di atur agar menunjuk variable x. hal seperti ini harap diperhatikan. Kalau program melibatkan pointer, dan pointer belum diinisialisasi, ada kemungkinan akan terjadi masalah yang dinamakan “Bug ”. masalah “ Bug ” terjadi bisa menyebabkan computer tak dapat dikendalikan lagi (“ hang”).

Hasil program di atas :



## **Tipe Variable Pointer Dan Tipe Objek yang ditunjuk**

Antara tipe data pointer ( sesuai dengan pendeklarasian pointer) dan tipe objek yang akan di tunjukan oleh pointer haruslah sejenis. Kalau misalnya pointer ***pu*** dimaksudkan untuk menunjukan data bertipe **int** maka data yang akan di tunjukan oleh pointer **pu** juga harus bertipe **int** . Sesuatu kesalahan akan terjadi dikalau misalnya pointer **float** digunakan untuk menunjukan data bertipe . Hal ini ditunjukan oleh contoh program berikut :



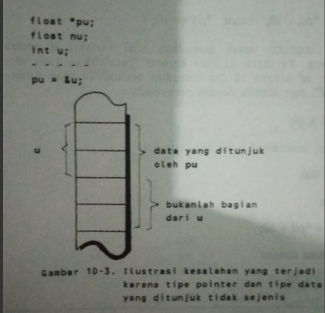
Pada contoh program diatas, saat penugasan

**Pu = &u;**

Maka **pu** akan menujuk data yang berukurran 4 byte ( **tipe float** ) sekalipun **u** bertipe **int** . Oleh karena itu , pernyataan

**Nu = \*pu;**

tidak akan membuat **nu** berisi nilai **u.** untuk lebih jelasnya lihat gambar berikut



## **Mengubah Isi Suatu variable Melalui Pointer.**

Contoh berikut memberikan gambaran tentang pengubahan isi suatu variable **secara tidak langsung** ( yaitu melalui pointer ). Mula- mula **pd** dideklarasikan sebagai pointer yang menunjukan ke suatu data bertipe **float** dan **b** sebagai variable bertipe **float.** Selanjutnya

**D = 54, 5;**

Digunakan untuk nilai mengisikan nilai 54,5 secara **langsung** ke variable **d.** Adapun

**Pd = &d;**

Digunakan untuk memberikan alamt dari **d** ke **pd.** Dengan demikian **pd** menunjukan ke variable **d.** Sedangkan pernyataan berikutnya.

**\*pd = \*pd + 10; ( atau \*pd +=10; )**

Merupakan intruksi utnuk mengubah nilai variable **d** secara tak langsung. Perintah diatas berarti “Jumlahkan yang ditunjukan oleh **pd** dengan 10 dan kemudian berikan ke yang ditunjukan oleh **pd”** , atau identik dengan pernyataan

**D = d + 10;**

Akan tetapi, seandainya tidak ada intruksi

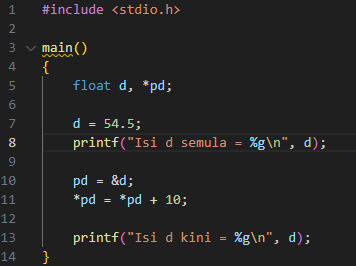
Pd = &d;

Pernyataan

**\*pd = \*pd +10;**

Tidaklah sama dengan

**D = d + 10;**

****

Hasil program :

****

## **Pointer Dan Array.**

Hubungan antara pointer dan array pada C sangatlah erat. Sebab sesungguhnya array secara internal akan diterjemahkan dalam bentuk pointer. Pembahasan berikut akan memberikan gmabaran hubungan antara pointer dan array.

Misalnya dideklarasikan di dalam suatu fungsi

**Static int tgl\_lahir[3] = { 01, 09, 64};**

Kemudian diberikan intruksi

**Ptgl = &tgl\_lahir[0];**

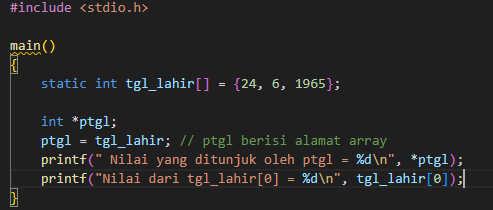
Maka ***ptgl*** akan berisi Alamat dari elemen array ***tgl\_lahir*** yang berindeks nol. Intruksi di atas bisa juga ditulis menjadi

***Ptgl = tgl\_lahir;***

Sebab nama array tanpa tanda kurang menyatakan Alamat awal dari array. Sesudah penugasan seperti di atas,

**\*ptgl**

Dengan sendirinya menyatakan elemen pertama ( berindeks sama dengan nol) dari array tgl\_lahir. Hal ini bisa dilihat melalui pembuktian program berikut

****

Secara umum operasi pointer bisa diterangkan sebagai berikut. Katakanlah **a** adalah suatu **array**, dan **pa** adalah pointer yang menunjukan array **a**, maka

\*(pa + i)

Akan menyatakan elemen array dengan indeks sama dengan i. jadi

**\*(pa + 0) identik a[0]**

**\*(pa + 1) identik a[1]**

**\*(pa + 2) identik a[2]**

Ungkapan seperti

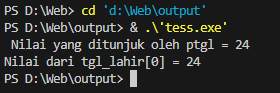
**pa + i**

memiliki arti “tambahan nilai **pa** (berisi Alamat) dengan I kali ukuran dari objek yang ditunjukan oleh **pa**” kalau **pa** dideklarasikan sebagai

**int \*pa;**

maka objek dari **pa** adalah data int (berukuran 2 byte ).

Hasil program di atas



## **Pointer dan String**



Hasil program



Pada program diatas ,

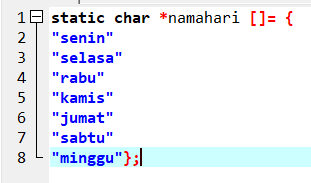
Char \*pkota = “ ACEH”;

Akan menyebabakan Kompiler

* Mengalokasikan variable pkota sebagai variable pointer yang menunjukan ke objek bertipe char dan menepatkan konstanta “ ACEH” dalam suatu memori.
* Kemudian pointer pkota akan menunjukan ke lokasi string “ACEH”

## **10 Inisialiasi Array Pointer**

Array pointer bisa diinisilisasikan sewaktu pendeklarasian. Sebagai contoh



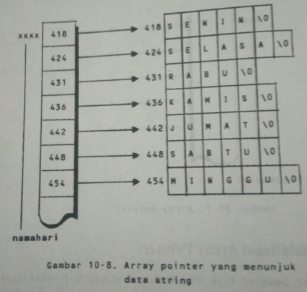
Pada contoh ini,

**Namahari[0] menunjuk ke string “senin”**

**Namahari[1] menunjuk ke string “selasa”**

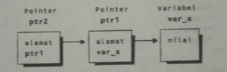
**Namahari[2] menunjuk ke string “rabu”**

**Dan sebagainya**

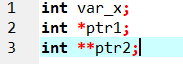


# **Pointer Menunjuk Pointer**

Suatu pointer bisa saja menunjuk ke pointer lain. Berikut gambar nya



Untuk membentuk rantai pointer seperti pada gambar di atas, pedeklarasian yang diperlukan berupa



Perhatikan pada deklarasi di depan :

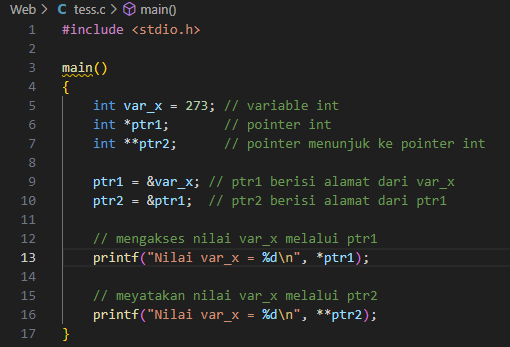
* Var\_x adalah variable bertipe int
* Ptr1 adalah variable pointer yang menunjukan ke data bertipe int
* Ptr2 adalah variable pointer yang menunjukan ke pointer int. (itulah sebabnya deklarasinya berupa : int \*\*ptr2;)

Agar **ptr1** menunjuk ke variable **var\_x**, perintah yang diperlukan berupa

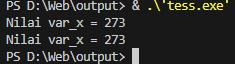
**Ptr1 = &var\_x;**

Sedangkan supaya **ptr2** menunjuk ke **ptr1**, intruksi yang di perlukan adalah

**Ptr2 = &ptr1;**

****Contoh berikut memberikan gambaran cara pengaksesan nilai pada var\_x melalui pointer **ptr2** dan **ptr1.**

Hasil programnya



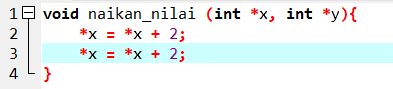
# **Pointer Dan Fungsi**

Pointer dan kaitannya dengan fungsi yang akan dibahas berikut meliputi

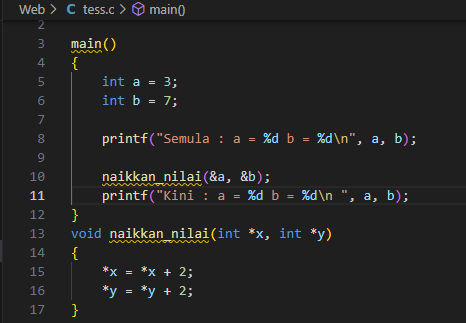
* Pointer sebagai parameter fungsi
* Pointer sebagai keluaran fungsi

POINTER SEBAGAI PARAMETER FUNGSI

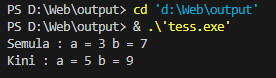
Penerapan pointer sebagai parameter yaitu jika diinginkan agar nilai suatu variable internal dapat di ubah oleh fungsi yang dipanggil. Sebagai contoh berikut



Fungsi di atas dimaksudkan agar kalau dipanggil, variable yang berkaitan dengan parameter aktual dapat diubah nilainya , masing- masing dinaikan sebesar 2. Contoh pemanggilannya :

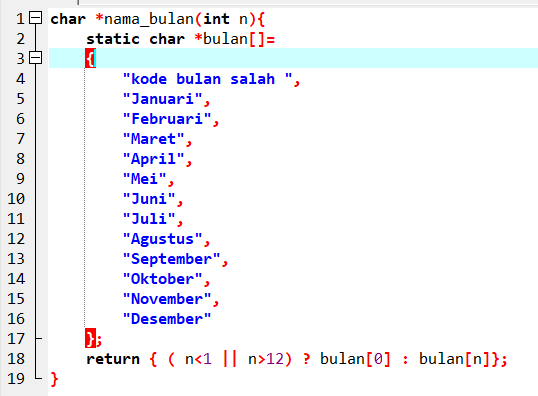
Dalam hal ini variable a dan b haruslah di tulis diawali dengan operator Alamat (&) yang berarti menyatakan Alamat variable, sebab parameter fungsi dalam pendefenisian berupa p

Hasil program nya



# **Pointer Sebagai Keluaran Fungsi**

Suatu fungsi dapat dibuat agar keluarnya berupa pointer. Misalnya, suatu fungsi menghasilkan keluaran berupa pointer yang menunjuk ke string nama bulan, seperti pada contoh berikut .



Pada definisi fungsi di atas,

**Char \*nma\_bulan**

Menyatakan bahwa keluaran fungsi nama\_bulan () berupa pointer yang menunjuk ke objek bertipe char (atau string).

Dalam fungsi **nama\_bulan**(), mula-mula array Bernama bulan dideklarasikan dan sekaligus diinisialisasikan agar menunjuk sejumlah string yang menyatakan nama bulan. Dibagian akhir fungsi , pernyaataan

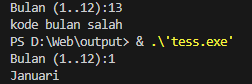
**Return ( ( n<1 || n> 12) ? bulan[0] : bulan[n]);**

Menyatakan bahwa hasil berupa pointer yang menunjuk ke

* String “ kode bulan salah “ (bulan[0]) jika masukan fungsi; n<1 atau n>12
* Bulan[n] untuk n yang terletak antara 1 sampai dengan 12.

Contoh fungsi dengan keluaran berupa pointer yang menunjuk ke string

Hasil program nya



# **BAB II**

# **PENUTUP**

# **KESIMPULAN**

* Alamat Memori: Pointer menyimpan alamat memori dari variabel lain dalam program. Dengan menggunakan pointer, programmer dapat mengakses dan memanipulasi nilai variabel tersebut secara langsung melalui alamat memori, yang memungkinkan penggunaan memori yang lebih efisien.
* Dynamic Memory Allocation: Pointer memungkinkan alokasi memori secara dinamis menggunakan fungsi seperti malloc() dan free(). Ini memungkinkan aplikasi untuk mengalokasikan memori saat dibutuhkan dan membebaskannya saat sudah tidak diperlukan lagi, menghindari pemborosan memori.
* Manipulasi Data: Dengan menggunakan pointer, programmer dapat dengan mudah memanipulasi data seperti array dan string. Pointer dapat digunakan untuk mengakses elemen-elemen array dan karakter dalam string, memungkinkan operasi pengolahan data yang efisien.
* Peningkatan Kecepatan dan Kinerja: Penggunaan pointer dapat meningkatkan kecepatan dan kinerja program karena memungkinkan akses langsung ke lokasi memori, mengurangi overhead yang terkait dengan mengakses data melalui variabel.
* Penggunakan Fungsi dan Struktur Data: Pointer memungkinkan penggunaan fungsi yang mengembalikan nilai pointer, memungkinkan pengembalian data yang kompleks seperti array dan struktur dari fungsi. Ini juga memungkinkan pembuatan struktur data yang kompleks seperti linked lists, trees, dan graphs.
* Perhatian Terhadap Kesalahan: Penggunaan pointer memerlukan kehati-hatian dalam pengelolaan memori. Kesalahan seperti dereference pointer yang belum diinisialisasi atau dereference pointer yang sudah dibebaskan memori dapat menyebabkan bug dan crash pada program.
* Penggunaan Operator: Dalam bahasa C, operator dereference (\*) dan address-of (&) digunakan untuk bekerja dengan pointer. Operator ini memungkinkan programmer untuk mengakses nilai yang disimpan di alamat memori yang ditunjuk oleh pointer, serta mendapatkan alamat memori dari variabel.

Kesimpulannya, penggunaan pointer dalam bahasa C memberikan fleksibilitas, efisiensi memori, dan kontrol yang tinggi kepada programmer. Namun, dengan kekuatan ini datang tanggung jawab besar untuk mengelola memori dengan benar agar menghindari kesalahan dan bug dalam program.

# **DAFTAR PUSTAKA**

C Self study guide

Bolon, croig, Mastering

Pemograman Dasar Bahasa C 1991,1994, 1997, 2003

https://www.petanikode.com/c-pointer/