

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
PEMBAHASAN	3
A. PENGERTIAN POINTER	3
1. Mengakses data dengan Pointer	8
2. Pointer Menuju Objek Lain.....	11
3. Mendeklarasikan Variable Pointer	11
4. Mengatur Pointer Agar Menunjuk ke Variable Lain.	13
5. Mengakses Isi Suatu Variable Melalui Pointer	14
6. Tipe Variable Pointer Dan Tipe Objek yang ditunjuk.....	16
7. Mengubah Isi Suatu variable Melalui Pointer.	18
8. Pointer Dan Array.....	20
9. Pointer dan String	22
10 Inisialisasi Array Pointer	23
B. Pointer Menunjuk Pointer	24
C. Pointer Dan Fungsi	26
D. Pointer Sebagai Keluaran Fungsi.....	27
BAB II	30
PENUTUP.....	30
A. KESIMPULAN	30
DAFTAR PUSTAKA	32

BAB I

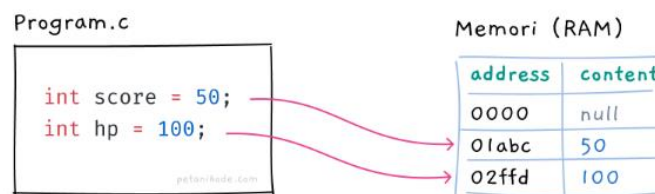
PEMBAHASAN

A. 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:



Setiap variabel yang kita buat di dalam program akan memiliki alamat memori untuk menyimpan nilai

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:

```
#include <stdio.h>

void main()
{
    int a;
    char b[10];

    printf("Alamat memori variabel a: %x\n", &a);
    printf("Alamat memori variabel b: %x\n", &b);
}
```

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:

```
PS D:\C\PENGKONDISIAN\output> & .\'tes.exe'
Alamat memori variabel a: 61ff1c
Alamat memori variabel b: 61ff12
```

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

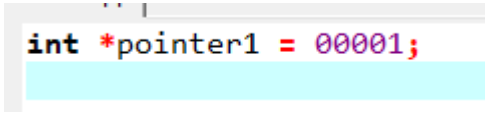
Kata kunci yang perlu kamu ingat:

“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:



```
int *pointer1 = 00001;
```

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:

```
int *pointer_ku;  
  
// atau bisa juga  
  
int *pointer_ku = NULL;
```

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

Sekarang masalahnya:

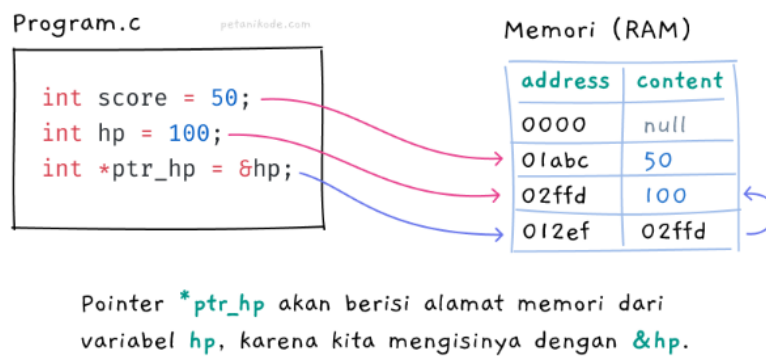
Karena kita tidak bisa lihat daftar alamat memori secara langsung, kita akan kesulitan memberikan referensi alamat memori untuk pointer.

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:



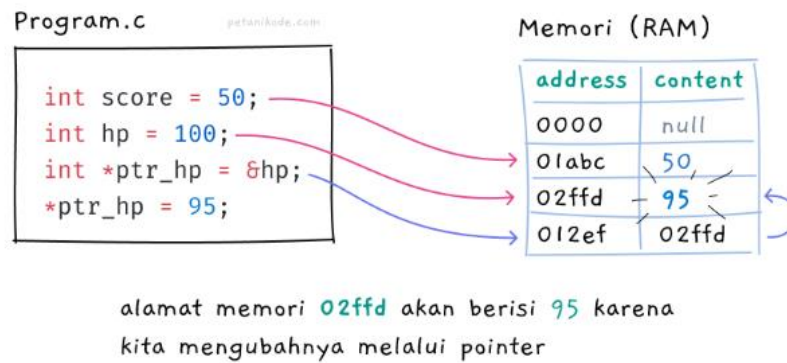
Pada gambar ini, kita membuat pointer dengan nama `*ptr_hp` dengan isi alamat memori dari variabel `hp`.

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:

```
*ptr_hp = 95;
```

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



1. Mengakses data dengan Pointer

Buatlah program baru dengan nama contoh_pointer.c, kemudian isi dengan kode berikut:

```
#include <stdio.h>

void main() {
    int score = 50;
    int hp = 100;

    // membuat pointer dengan isi alamat memori dari hp
    int *ptr_hp = &hp;

    // print isi variabel dan alamat memori
    printf("Nama Variabel \t Alamat \t Konten\n");
    printf("score \t\t %x \t %d \n", &score, score);
    printf("hp \t\t %x \t %d \n", &hp, hp);
    printf("ptr_hp \t %x \t %x \n", &ptr_hp, ptr_hp);
    printf("*ptr_hp \t %x \t %d \n", &ptr_hp, *ptr_hp);

    // mengubah data pada alamat memori dengan pointer
    *ptr_hp = 95;

    printf("hp \t\t %x \t %d \n", &hp, hp);
    printf("*ptr_hp \t %x \t %d \n", &ptr_hp, *ptr_hp);
}
```


Setelah itu, coba compile dan jalankan.

Maka hasil:

Nama Variabel	Alamat	Konten
score	61ff1c	50
hp	61ff18	100
ptr_hp	61ff14	61ff18
*ptr_hp	61ff14	100
hp	61ff18	95
*ptr_hp	61ff14	95

Pointer *ptr_hp berhasil mengubah nilai pada alamat d57ba6c menjadi 95.

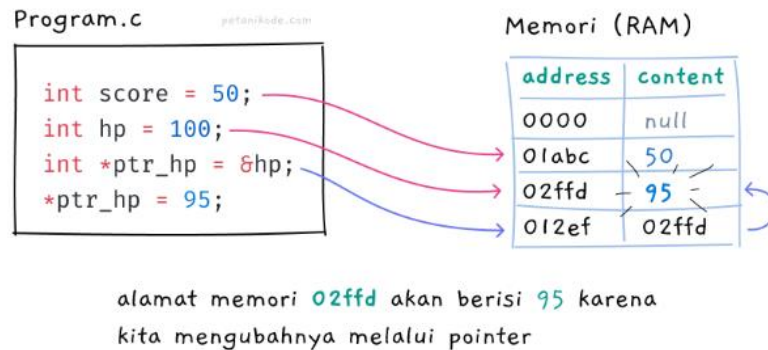
Saat menggunakan pointer, kita menggunakan tanda * di depan nama pointer untuk mengakses nilai pada alamat memori. Jika tidak menggunakan tanda ini, maka kita akan mendapatkan alamat memori yang di-pointing.

```
*ptr // ini akan berisi 95 (nilai dari alamat)  
ptr // ini akan berisi d57ba6c (alamat memori dari variabel hp)
```

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?

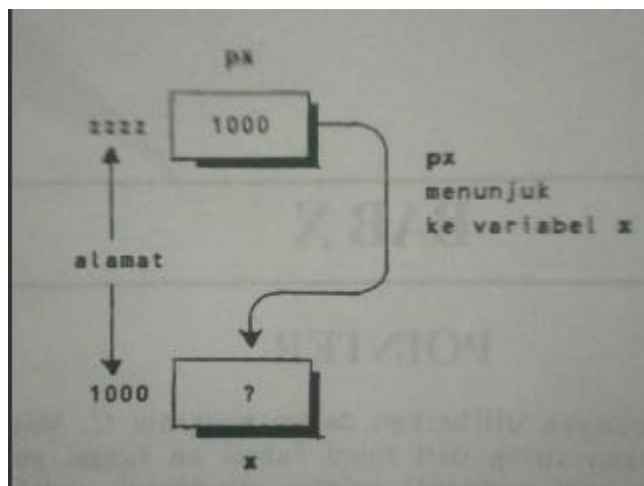
Penggunaan pointer sebenarnya opsional, kamu boleh pakai.. boleh juga tidak.

Namun..

Pada kondisi tertentu, penggunaan pointer lebih optimal.

2. Pointer Menuju Objek Lain

Variable pointer sering dikatakan sebagai variable yang menunjukan ke objek lain. Pada kenyataan yang sebenarnya, variable pointer (Atau di singkat 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.



3. Mendeklarasikan Variable Pointer

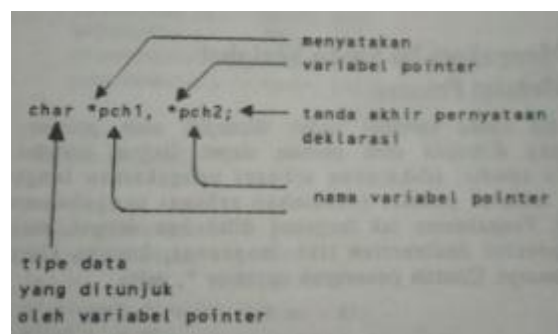
Suatu variable pointer dideklarasikan dengan bentuk sebagai berikut :

```
// tipe data *Nama_variable;
```

Adapun *Nama_variable* adalah nama dari variable pointer

```
int *px; // contoh 1
char *pch1, *pch2; // contoh 2
```

Contoh pertama menyatakan 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

```
#include <stdio.h>

int main() {
    int num = 10; // variabel integer
    int *ptr; // deklarasi pointer

    ptr = &num; // ptr menunjuk ke alamat memori dari variabel num

    printf("Nilai dari variabel num: %d\n", num);
    printf("Alamat memori dari variabel num: %p\n", &num);
    printf("Nilai yang ditunjuk oleh pointer ptr: %d\n", *ptr); // dereferensi pointer untuk mendapatkan nilai
    printf("Alamat memori yang ditunjuk oleh pointer ptr: %p\n", ptr);

    return 0;
}
```

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

```
Nilai dari variabel num: 10
Alamat memori dari variabel num: 000000000062FE14
Nilai yang ditunjuk oleh pointer ptr: 10
Alamat memori yang ditunjuk oleh pointer ptr: 000000000062FE14
-----
```

4. 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 ditunjukkan. Untuk 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 dieksekusi berulah dapat dikatakan bahwa **px** menunjuk ke variable x.

5. Mengakses Isi Suatu Variable Melalui Pointer

Kalau suatu variable sudah ditunjukan oleh pointer, variable yang ditunjukan oleh pointer dapat diakses 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 * (bersifat 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 .

```
px = &x;  
y = *px;
```

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

```

Web > C tess.c > main()
1  #include <stdio.h>
2
3  main()
4  {
5      int x, y;
6      int *px;
7
8      x = 87;
9
10     px = &x;
11     y = *px;
12
13     printf("Alamat x = %p\n", &x);
14     printf("Isi px = %p\n", px);
15     printf("Isi x = %d\n", x);
16     printf("Nilai yang di tunjukan oleh px = %d\n", *px);
17     printf("Nilai y = %d\n", y);
18 }

```

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 :

```
Alamat x = 0061FF14
Isi px = 0061FF14
Isi x = 87
Nilai yang di tunjukan oleh px = 87
Nilai y = 87
```

6. 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 :


```

Web > C tess.c > main()
1  #include <stdio.h>
2
3  main()
4  {
5      float *pu;
6      float nu;
7
8      int u = 1234;
9
10     pu = &u; // Pernyataan ini salah
11             // Sebab pu adalah pointer float
12             // padahal u bertipe int
13
14     nu = *pu;
15
16     printf("*u = %d\n", u);
17     printf("*nu = %f\n", nu);
18 }

```

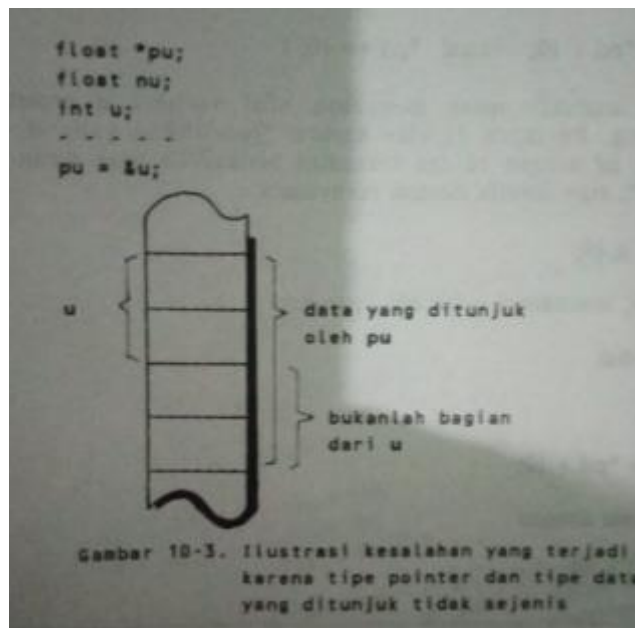
Pada contoh program diatas, saat penugasan

Pu = &u;

Maka **pu** akan menunjuk data yang berukuran 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



7. 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 mengisi nilai 54,5 secara **langsung** ke variable **d**. Adapun

Pd = &d;

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

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

Merupakan intruksi untuk 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;

```
1  #include <stdio.h>
2
3  main()
4  {
5      float d, *pd;
6
7      d = 54.5;
8      printf("Isi d semula = %g\n", d);
9
10     pd = &d;
11     *pd = *pd + 10;
12
13     printf("Isi d kini = %g\n", d);
14 }
```

Hasil program :

```
Isi d semula = 54.5  
Isi d kini = 64.5
```

8. 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 gambaran 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

```
#include <stdio.h>

main()
{
    static int tgl_lahir[] = {24, 6, 1965};

    int *ptgl;
    ptgl = tgl_lahir; // ptgl berisi alamat array
    printf(" Nilai yang ditunjuk oleh ptgl = %d\n", *ptgl);
    printf("Nilai dari tgl_lahir[0] = %d\n", tgl_lahir[0]);
}
```

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

```
PS D:\Web> cd 'd:\Web\output'
PS D:\Web\output> & .\'tess.exe'
  Nilai yang ditunjuk oleh ptgl = 24
  Nilai dari tgl_lahir[0] = 24
PS D:\Web\output> |
```

9. Pointer dan String

```
1  #include <stdio.h>
2
3  main()
4  {
5      char *pkota = "Aceh"; // pkota menunjuk
6                          // konstanta string "Aceh"
7
8      puts(pkota);
9  }
```

Hasil program

```
PS D:\Web\output> & .\'tess.exe'
Aceh
```

Pada program diatas ,

Char *pkota = “ ACEH”;

Akan menyebabkan 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 Inisialisasi Array Pointer

Array pointer bisa diinisialisasikan sewaktu pendeklarasian. Sebagai contoh

```
1 static char *namahari []= {  
2     "senin"  
3     "selasa"  
4     "rabu"  
5     "kamis"  
6     "jumat"  
7     "sabtu"  
8     "minggu"};
```

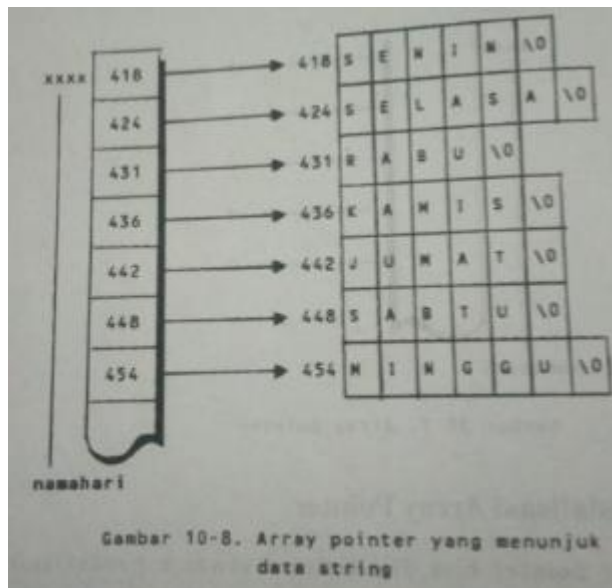
Pada contoh ini,

Namahari[0] menunjuk ke string “senin”

Namahari[1] menunjuk ke string “selasa”

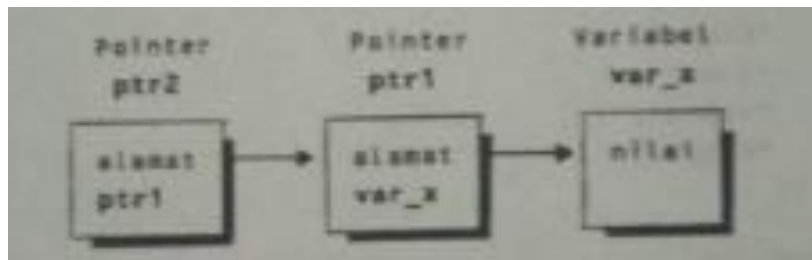
Namahari[2] menunjuk ke string “rabu”

Dan sebagainya



B. Pointer Menunjuk Pointer

Suatu pointer bisa saja menunjuk ke pointer lain. Berikut gambarnya



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

```

1 int var_x;
2 int *ptr1;
3 int **ptr2;
  
```


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**.

```
Web > C tess.c > main()
1  #include <stdio.h>
2
3  main()
4  {
5      int var_x = 273; // variable int
6      int *ptr1;      // pointer int
7      int **ptr2;     // pointer menunjuk ke pointer int
8
9      ptr1 = &var_x; // ptr1 berisi alamat dari var_x
10     ptr2 = &ptr1;  // ptr2 berisi alamat dari ptr1
11
12     // mengakses nilai var_x melalui ptr1
13     printf("Nilai var_x = %d\n", *ptr1);
14
15     // meyakini nilai var_x melalui ptr2
16     printf("Nilai var_x = %d\n", **ptr2);
17 }
```

Hasil programnya

```
PS D:\Web\output> & .\tess.exe
Nilai var_x = 273
Nilai var_x = 273
PS D:\Web\output>
```

C. 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

```
1 void naikan_nilai (int *x, int *y){
2     *x = *x + 2;
3     *x = *x + 2;
4 }
```

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

```
Web > C tess.c > main()
2
3  main()
4  {
5      int a = 3;
6      int b = 7;
7
8      printf("Semula : a = %d b = %d\n", a, b);
9
10     naikkan_nilai(&a, &b);
11     printf("Kini : a = %d b = %d\n ", a, b);
12 }
13 void naikkan_nilai(int *x, int *y)
14 {
15     *x = *x + 2;
16     *y = *y + 2;
17 }
```

operator Alamat (&) yang berarti menyatakan Alamat variable, sebab parameter fungsi dalam pendefenisian berupa p

Hasil program nya

```
PS D:\Web\output> cd 'd:\Web\output'
PS D:\Web\output> & .\'tess.exe'
Semula : a = 3 b = 7
Kini : a = 5 b = 9
```

D. Pointer Sebagai Keluaran Fungsi

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

```

1 char *nama_bulan(int n){
2     static char *bulan[]=
3     {
4         "kode bulan salah ",
5         "Januari",
6         "Februari",
7         "Maret",
8         "April",
9         "Mei",
10        "Juni",
11        "Juli",
12        "Agustus",
13        "September",
14        "Oktober",
15        "November",
16        "Desember"
17    };
18    return { ( n<1 || n>12) ? bulan[0] : bulan[n]};
19 }

```

Pada definisi fungsi di atas,

Char *nama_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 , pernyataan

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

```
Web > C tess.c > ...
1  #include <stdio.h>
2
3  char *nama_bulan(int n);
4
5  /// @brief
6  main()
7  {
8      int bl;
9
10     printf("Bulan (1..12):");
11     scanf("%d", &bl);
12     printf("%s\n", nama_bulan(bl));
13 }
14 char *nama_bulan(int n)
15 {
16     static char *bulan[] = {
17         "kode bulan salah",
18         "Januari",
19         "Februari",
20         "Maret",
21         "April",
22         "Mei",
23         "Juni",
24         "Juli",
25         "Agustus",
26         "September",
27         "Oktober",
28         "November",
29         "Desember"};
30     return ((n < 1 || n > 12) ? bulan[0] : bulan[n]);
31 }
```

Hasil program nya

```
Bulan (1..12):13
kode bulan salah
PS D:\Web\output> & .\'tess.exe'
Bulan (1..12):1
Januari
```

BAB II

PENUTUP

A. 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.

- **Penggunaan 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/>