

# **PERTEMUAN 9**

## **Record Dan Pointer**

**Hermanto, S.Kom. M.Kom**

# REVIEW RECORD (REKAMAN)

---

**Record** disusun oleh satu atau lebih field. Tiap field menyimpan data dari tipe dasar tertentu atau dari tipe bentukan lain yang sudah didefinisikan sebelumnya. Nama Record ditentukan oleh pemrogram.

Record disebut juga **tipe terstruktur**.

**Record** adalah jenis tipe data terstruktur yang berisi beberapa data, yang masing-masing dapat berlainan tipe.



# Mendeklarasikan Record

---

Suatu tipe record dideklarasikan dengan bentuk sebagai berikut :

```
RECORD  
  Daftar_field_1 : tipe_1;  
  Daftar_field_2 : tipe_2;  
  ...  
  daftar_field_n : tipe_n;  
END
```

Masing-masing *daftar\_field* dapat berupa satu atau beberapa nama pengenalan dan masing-masing dinamakan *field*. Bila *daftar\_field* berisi lebih dari satu field, antar field perlu dipisahkan dengan koma. Masing-masing tipe dapat berupa tipe data apa saja termasuk *array*.

# Mendeklarasikan Record

---

Berikut contoh pendeklarasian record :

```
Type
    RecBarang = Record
        Nama : String;
        Kualitas : Char;
        Harga : LongInt
    End;

Var
    Barang : RecBarang;
```

Dengan mendeklarasikan seperti di atas, Barang akan mengandung tiga buah *field*, yaitu :

- Nama,
- Kualitas
- Harga.

# Cara Mengakses Field

---

Field dari suatu *record* diakses dengan bentuk :

*Variabel.field*

Sebagai contoh :

**Barang>Nama**

Berarti "*field* Nama dari variabel record bernama Barang".

Contoh penugasan nilai ke field tersebut :

**Barang>Nama := 'Ubin TISKA 20x20';**

Dengan cara seperti di atas, field Nama dari record Barang berisi string 'Ubin TISKA 20x20'.

Isi dari suatu field ditampilkan dengan menggunakan **cout**.

Contoh :

**cout (Barang>Nama);**

Merupakan perintah untuk menampilkan isi field Nama dari record Barang.

# Penugasan Antar Record

---

Jika record R1 dan R2 bertipe sama dan masing-masing memiliki F1, F2, dan F3, maka penugasan :

R1 := R2;

diperkenalkan. Pernyataan di atas merupakan penyederhanaan dari sederetan pernyataan berikut:

R1.F1 := R2.F1;

R1.F2 := R2.F2;

R1.F3 := R2.F3;

Untuk lebih jelasnya, berikut penulisan dalam program nya.



# Penugasan Antar Record

```
Program Rec2;
Uses crt;
Type
    RecBarang = Record
        Nama    : string[25];
        Kualitas : char;
        Harga    : Int
    End;
Var
    Barang1, Barang2 : RecBarang;    {variabel bertipe record}
Begin
    Clrscr;
    {penugasan nilai terhadap field-field}
    Barang1.Nama    := 'Ubin TISKA 20x20';
    Barang1.Kualitas := 'A';
    Barang1.Harga    := 14000;

    {menyalin record}
    Barang2 := Barang1;

    Menampilkan isi field}
    cout ('Nama Barang      : ', Barang2.Nama);
    cout ('Kualitas        : ', Barang2.Kualitas);
    cout ('Harga            : ', Barang2.Harga);
End.
```

# Penugasan Antar Record

---

Dengan adanya penugasan

Barang2 := Barang1;

maka semua field pada record Barang2 akan berisi record Barang1.

Hasil dari program di atas :

**Nama Barang : Ubin TISKA 20x20**

**Kualitas : A**

**Harga : 14000**





# Record Di Dalam Record

Mungkin saja sebuah record berisi record. Sebagai gambaran hal ini, perhatikan deklarasi berikut:

```
RecTanggal = Record
    Tanggal,
    Bulan,
    Tahun :Integer
End;
RecPegawai = Record
    Nomor : Int;
    Nama  : String [35];
    TglLahir      : RecTanggal;
    Gaji   : LongInt
End;
```

Tampak bahwa tipe record bernama RecPegawai berisi record yang lain (RecTanggal). Hal yang menarik yang perlu diperhatikan adalah cara mengakses field seperti Tanggal, Bulan dan Tahun. Notasi yang diperlukan adalah sebagai berikut.

```
Nama_variabel.TglLahir.Tanggal
Nama_variabel.TglLahir.Bulan
Nama_variabel.TglLahir.Tahun
```

# Array Record

Elemen suatu array juga bisa berupa record. Sebagai contoh dapat dilihat di bawah ini.

```
Const
    Jum_Maks = 20;
Type
    RecBarang = Record
        Nama   : String [25];
        Kualitas : Char;
        Harga  : Int
    End;
    TabelBarang = Array [ 1 .. Jum_Maks] of RecBarang;
Var
    DafBarang : TabelBarang;    {array record}
```

Pada contoh di atas, DafBarang adalah array yang maksimum berisi 20 buah elemen bertipe record. Untuk mengakses suatu field, kita perlu menggunakan notasi :

```
DafBarang [indeks].NamaField
```

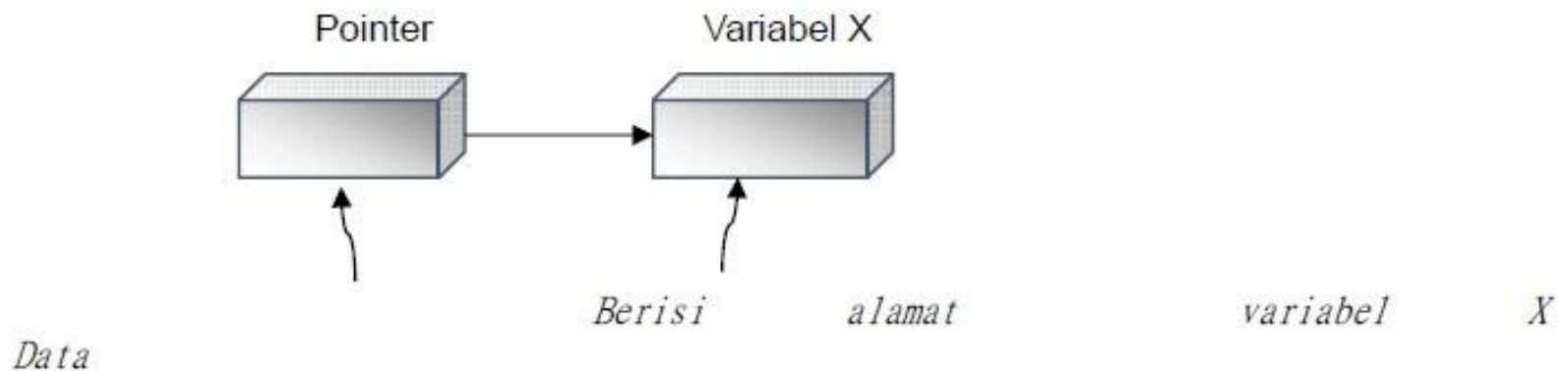
# Kesimpulan

---

- *Record* adalah salah satu tipe data terstruktur bentukan yang digunakan untuk mempresentasikan sebuah objek yang tidak dapat dipresentasikan menggunakan tipe data dasar, seperti integer, real, boolean, character. Setiap *record* terdiri dari beberapa elemen yang disebut *field*. Setiap *field* menggambarkan informasi tertentu, dan tipe setiap *field* sudah dikenal, baik itu tipe dasar atau tipe bentukan lainnya.

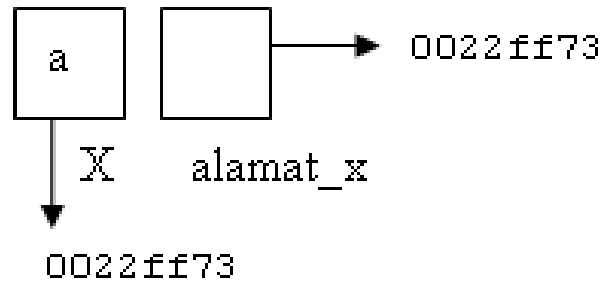
# Pointer

Pointer adalah tipe data yang digunakan untuk menunjuk ke suatu data. Suatu variable yang bertipe pointer (variabel pointer) tidak berisi data, melainkan berisi alamat suatu data. Di dalam komputer setiap lokasi data mempunyai alamat yang khas. Gambar berikut contoh suatu pointer yang menunjuk ke suatu data.



Berdasarkan kondisi di atas, dimungkinkan untuk mengakses data pada variabel X melalui Pointer. Pointer biasa digunakan sehubungan dengan pembentukan variabel dinamis.

# Ilustrasi Pointer



- ❑ Kita memiliki variabel `X` yang berisi nilai karakter `'a'`
- ❑ Oleh kompiler, nilai `'a'` ini akan disimpan di suatu alamat tertentu di memori.
- ❑ Alamat variabel `X` dapat diakses dengan menggunakan statemen **`&X`**.
- ❑ Jika kita ingin menyimpan alamat dari variabel `X` ini, kita dapat menggunakan suatu variabel  
misalnya **`char alamat_x = &X;`**
- ❑ `alamat_x` adalah suatu variabel yang berisi alamat dimana nilai `X`, yaitu `'a'` disimpan.
- ❑ Variabel `alamat_x` disebut variabel pointer atau sering disebut **pointer** saja.

# Mendeklarasikan Variabel Pointer

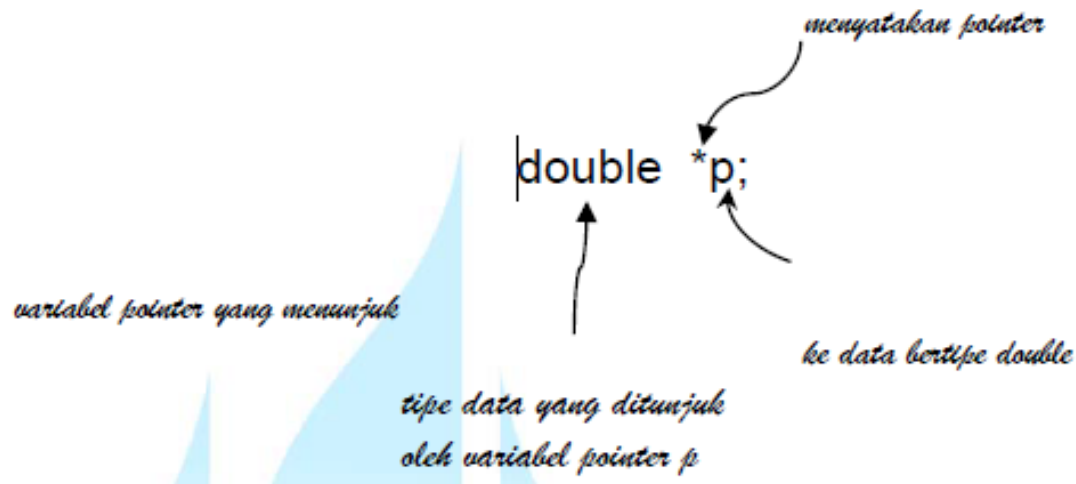
Bentuk deklarasi variabel pointer :

tipe \*variabel;

Contoh :

double \*p;

p adalah variabel pointer yang menunjuk ke data bertipe double.

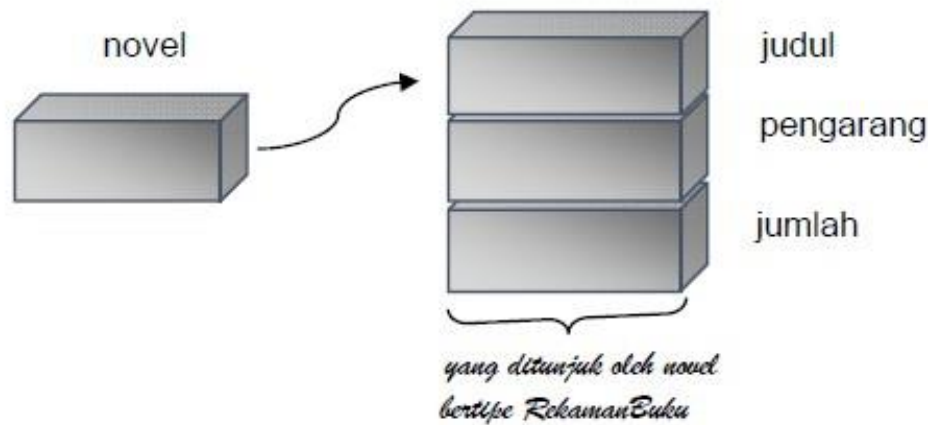


Contoh I

```
Rec RekamanBuku
{
    string judul;
    string pengarang;
    int jumlah;
}
Rec RekamanBuku *novel;
```

# Mendeklarasikan Variabel Pointer

Pada contoh ini, novel adalah variabel pointer yang menunjuk ke tipe struktur RekamanBuku. Gambarannya seperti berikut :



gambar 3. Variabel pointer yang menunjuk ke struktur

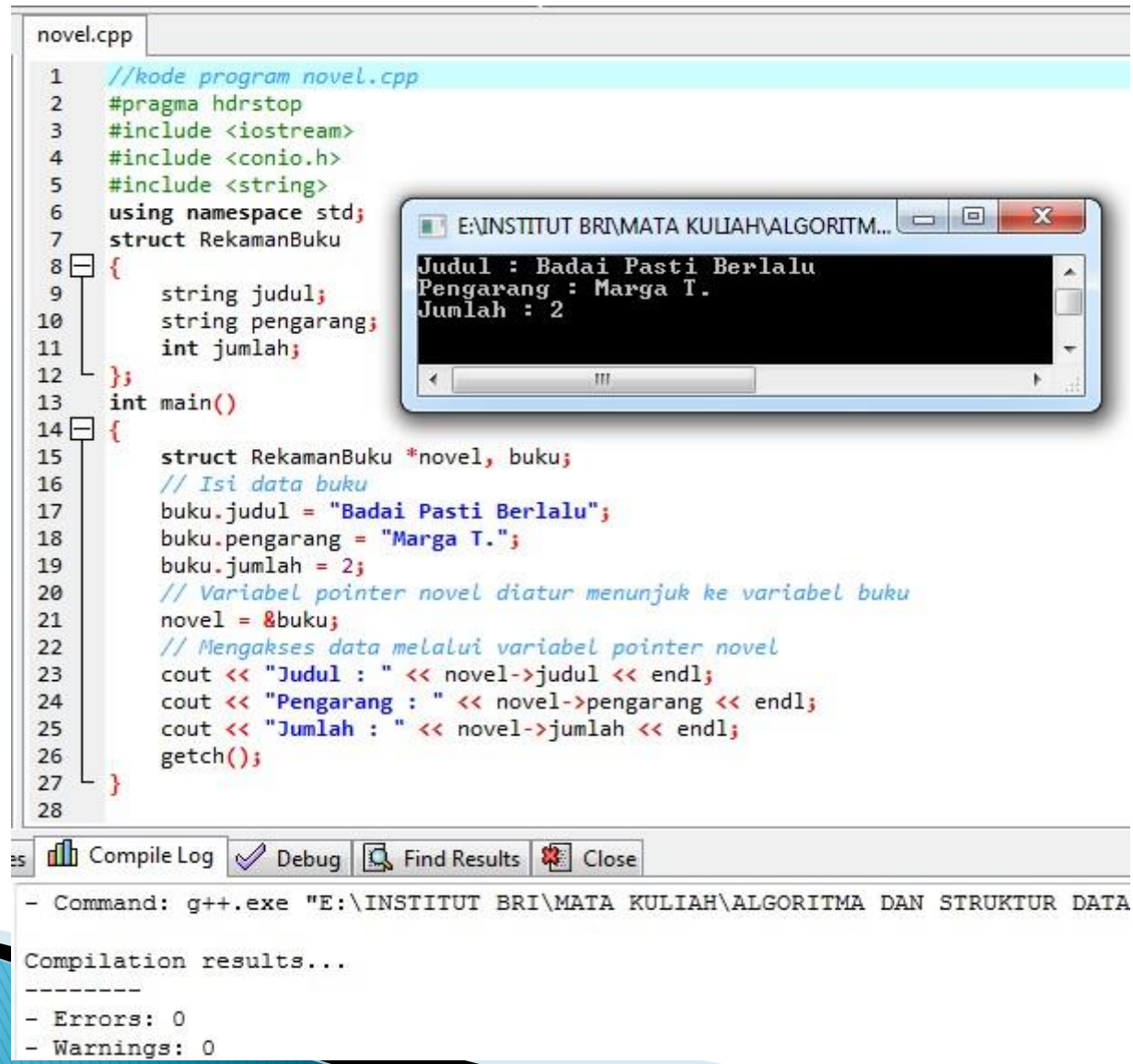
Supaya suatu variabel pointer menunjuk ke suatu variabel data, penugasan seperti berikut diperlukan :

```
Variabel_pointer = &variabel_data
```

- Simbol & berarti alamat
- Pernyataan di atas berarti bahwa : **variabel\_pointer** diisi dengan alamat **variabel\_data**

# Mengakses Data Via Pointer

Untuk melihat cara pengaksesan data melalui pointer, cobalah kode novel.cpp berikut :



The screenshot shows a C++ IDE with a file named `novel.cpp` open. The code defines a `RekamanBuku` struct with fields `judul`, `pengarang`, and `jumlah`. In the `main` function, a `RekamanBuku` object `buku` is created and populated with the title "Badai Pasti Berlalu", author "Marga T.", and count 2. A pointer `novel` is then assigned the address of `buku`. The program uses `cout` to print the data accessed through the pointer `novel`. A separate window displays the output of the program.

```
1 //kode program novel.cpp
2 #pragma hdrstop
3 #include <iostream>
4 #include <conio.h>
5 #include <string>
6 using namespace std;
7 struct RekamanBuku
8 {
9     string judul;
10    string pengarang;
11    int jumlah;
12 };
13 int main()
14 {
15     struct RekamanBuku *novel, buku;
16     // Isi data buku
17     buku.judul = "Badai Pasti Berlalu";
18     buku.pengarang = "Marga T.";
19     buku.jumlah = 2;
20     // Variabel pointer novel diatur menunjuk ke variabel buku
21     novel = &buku;
22     // Mengakses data melalui variabel pointer novel
23     cout << "Judul : " << novel->judul << endl;
24     cout << "Pengarang : " << novel->pengarang << endl;
25     cout << "Jumlah : " << novel->jumlah << endl;
26     getch();
27 }
28
```

Output window content:

```
Judul : Badai Pasti Berlalu
Pengarang : Marga T.
Jumlah : 2
```

Command: g++.exe "E:\INSTITUT BRI\MATA KULIAH\ALGORITMA DAN STRUKTUR DATA

Compilation results...

-----

- Errors: 0

- Warnings: 0



# Variabel Pointer

Contoh lain Suatu variable pointer didefinisikan dengan bentuk :

**TipeData \*NamaVariabel**

Contoh :

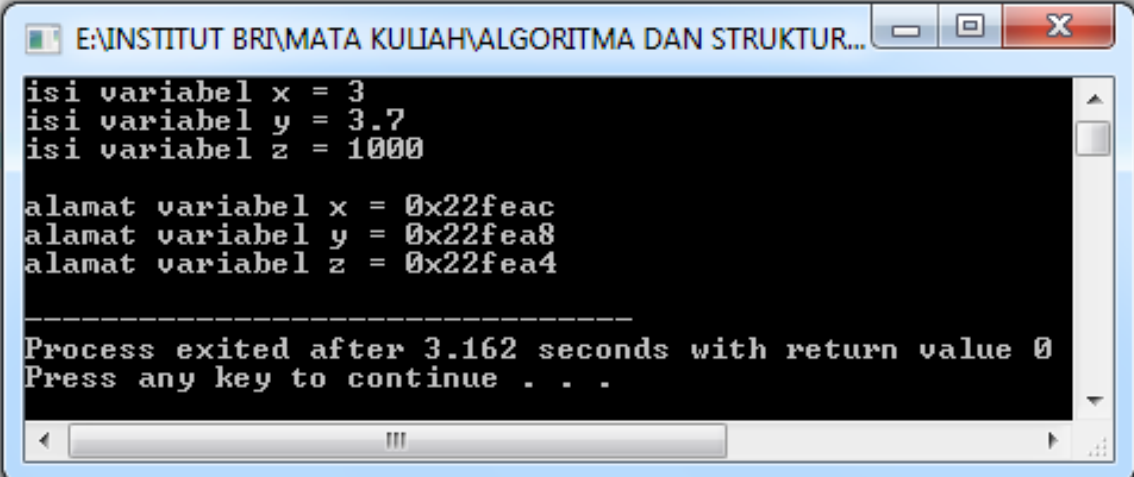
a	*c	b	*d	Var
2	*	3	*	Value
A	*	B	*	Address

Dari contoh di atas terlihat bahwa address pada variabel pointer dapat berubah – ubah, apabila address suatu variabel pointer berubah maka valuenya akan berubah sesuai address yang ditunjuk oleh pointer tersebut. Apabila pada address yang ditunjuk oleh pointer tersebut mengalami perubahan value, maka value pada pointer juga akan berubah.

# Variabel Pointer

```
pointerdata.cpp  dinamis.cpp  pointer.cpp  AlokasiPointer.cpp

1  #include <iostream>
2  #include <conio.h>
3  #include <string>
4  #pragma hdrstop
5
6  using namespace std;
7  int main()
8  {
9      int x;
10     float y;
11     long z;
12
13     x = 3;
14     y = 3.7;
15     z = 1000;
16
17     cout<<"isi variabel x = "<<x<<endl;
18     cout<<"isi variabel y = "<<y<<endl;
19     cout<<"isi variabel z = "<<z<<endl;
20
21     cout<<endl;
22
23     cout<<"alamat variabel x = "<<&x<<endl;
24     cout<<"alamat variabel y = "<<&y<<endl;
25     cout<<"alamat variabel z = "<<&z<<endl;
26 }
```



The screenshot shows a console window titled "E:\INSTITUT BRI\MATA KULIAH\ALGORITMA DAN STRUKTUR...". The output of the program is as follows:

```
isi variabel x = 3
isi variabel y = 3.7
isi variabel z = 1000

alamat variabel x = 0x22feac
alamat variabel y = 0x22fea8
alamat variabel z = 0x22fea4

-----
Process exited after 3.162 seconds with return value 0
Press any key to continue . . .
```

# Mengakses dan Mengubah isi Pointer

---

Program berikut memberikan gambaran tentang pengubahan isi suatu variable secara tak langsung (yaitu melalui pointer). Mula-mula **pd** dideklarasikan sebagai pointer yang menunjuk ke suatu data bertipe *float* dan **d** sebagai variabel bertipe *float*. Selanjutnya.

```
d = 54.5;
```

digunakan untuk mengisi nilai 54,5 secara langsung ke variabel **d**. Adapun

```
pd = &d;
```

alamat dari **d** ke **pd**. Dengan demikian **pd** menunjuk ke variabel **d**. Sedangkan pernyataan berikutnya

merupakan instruksi untuk mengubah nilai variabel **d** secara tak langsung.

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

menunjuk **pd** dengan 10 kemudian berikan ke yang ditunjuk oleh **pd**, atau identik dengan pernyataan

```
d = d + 10;
```

# Mengakses dan Mengubah isi Pointer

Akan tetapi, seandainya tidak ada instruksi

```
pd = &d;
```

maka pernyataan

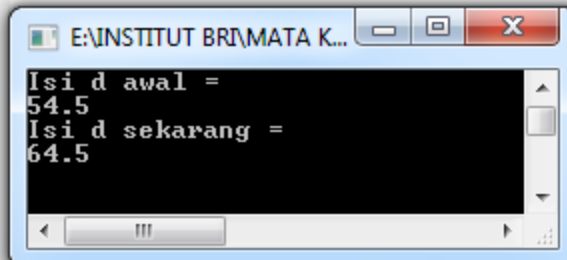
```
*pd = *pd + 10;
```

tidaklah sama dengan

```
d = d + 10;
```

UbahPointer.cpp

```
1  #include<iostream>
2  #include<conio.h>
3
4  using namespace std;
5  main()
6  {
7      float d = 54.5f, *pd;
8      cout <<"Isi d awal = \n"<< d<<endl;
9      pd = &d;
10     *pd += 10;
11     cout <<"Isi d sekarang = \n"<< d<<endl;
12     getch();
13     return 0;
14 }
```



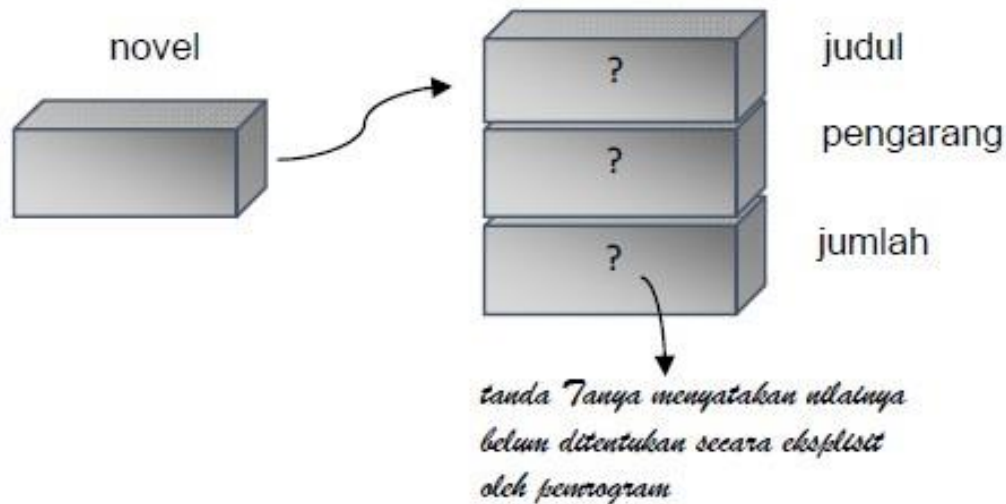
```
E:\INSTITUT BRI\MATA K...
Isi d awal =
54.5
Isi d sekarang =
64.5
```

# Variabel Dinamis

Variabel dinamis adalah variabel yang bisa dialokasikan di dalam memori atau dihapus dari memori ketika program dieksekusi. Menciptakan variabel dinamis butuh variabel pointer, kuncinya yaitu operator new. Misalnya terdapat variabel pointer novel yang bertipe pointer. Agar tercipta variabel dinamis yang akan ditunjuk oleh novel, perintahnya adalah :

```
Novel = new RekamanBuku;
```

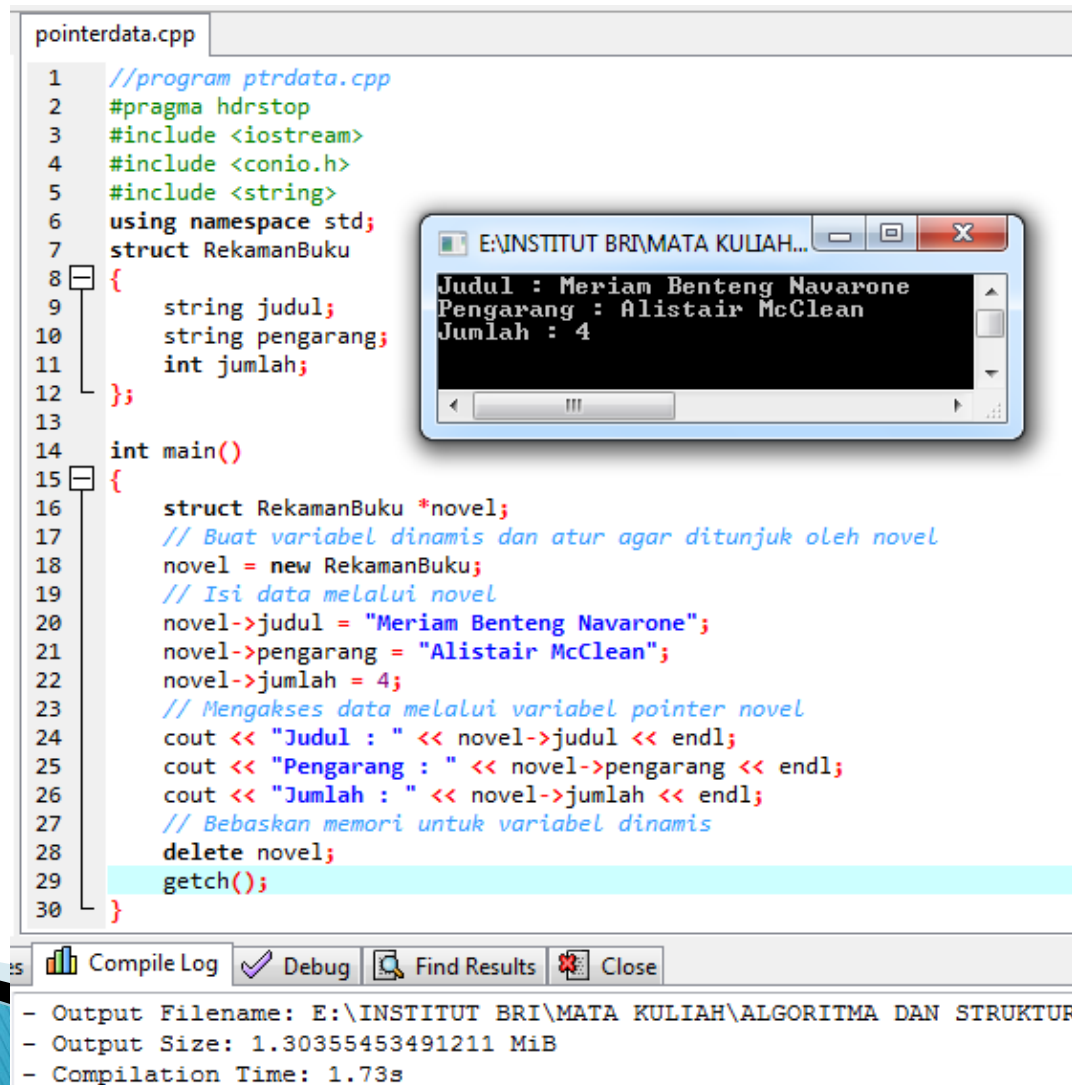
Gambar berikut menunjukkan keadaan setelah pernyataan tersebut dieksekusi :



Gambar 4. Keadaan setelah variabel dinamis diciptakan

# Variabel Dinamis

**pointerdata.cpp** memberikan gambaran tentang penciptaan variable dinamis dan pengaksesannya. Bandingkan dengan kode yang serupa, tetapi tidak memakai variabel dinamis yaitu **novel.cpp**.



```
pointerdata.cpp
1  //program ptrdata.cpp
2  #pragma hdrstop
3  #include <iostream>
4  #include <conio.h>
5  #include <string>
6  using namespace std;
7  struct RekamanBuku
8  {
9      string judul;
10     string pengarang;
11     int jumlah;
12 };
13
14 int main()
15 {
16     struct RekamanBuku *novel;
17     // Buat variabel dinamis dan atur agar ditunjuk oleh novel
18     novel = new RekamanBuku;
19     // Isi data melalui novel
20     novel->judul = "Meriam Benteng Navarone";
21     novel->pengarang = "Alistair McClean";
22     novel->jumlah = 4;
23     // Mengakses data melalui variabel pointer novel
24     cout << "Judul : " << novel->judul << endl;
25     cout << "Pengarang : " << novel->pengarang << endl;
26     cout << "Jumlah : " << novel->jumlah << endl;
27     // Bebaskan memori untuk variabel dinamis
28     delete novel;
29     getch();
30 }
```

Output window showing the program's results:

```
E:\INSTITUT BRI\MATA KULIAH...
Judul : Meriam Benteng Navarone
Pengarang : Alistair McClean
Jumlah : 4
```

Compile Log:

- Output Filename: E:\INSTITUT BRI\MATA KULIAH\ALGORITMA DAN STRUKTUR
- Output Size: 1.30355453491211 MiB
- Compilation Time: 1.73s

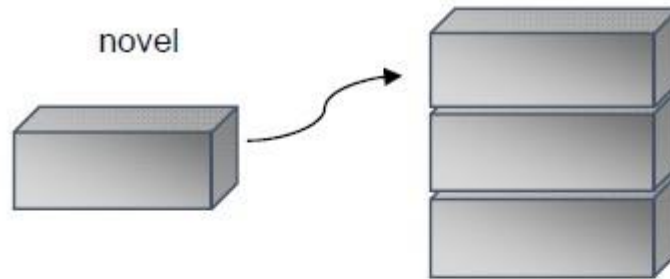
# Menghapus Variabel Dinamis

Bila suatu variabel dinamis tidak diperlukan lagi, memori yang digunakannya bisa dihapus. Perintah yang diperlukan untuk keperluan tersebut berupa prosedur delete. Perintahnya adalah :

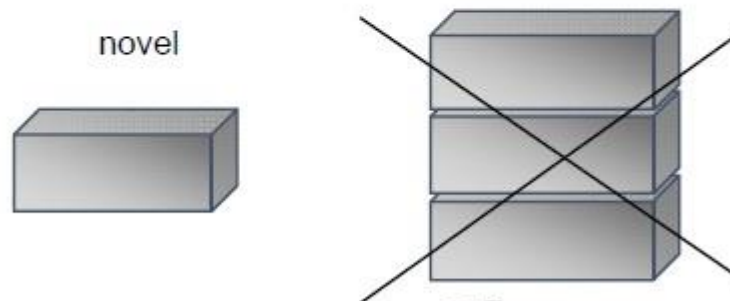
```
Deletevariabel_pointer;
```

Dengan cara tersebut, memori yang ditunjuk oleh variabel\_pointer dihapus.

*Keadaan awal :*



*Setelah delete novel :*



*Memori untuk data ini dibebaskan  
atau didealokasikan*

# Pointer dan Array Dinamis

Array juga dapat dipesan secara dinamis melalui new. Contoh kode program **dinamis.cpp**

```
pointerdata.cpp  dinamis.cpp
1  //kode program dinamis.cpp
2  #pragma hdrstop
3  #include <iostream>
4  #include <conio.h>
5  #include <string>
6  using namespace std;
7  struct RekamanBuku
8  {
9      string judul;
10     string pengarang;
11     int jumlah;
12 };
13
14 int main()
15 {
16     struct RekamanBuku *novel;
17     // Buat variabel dinamis berupa array dan atur agar ditunjuk oleh novel
18     novel = new RekamanBuku[5];
19     // Isi data melalui novel
20     novel[0].judul = "Meriam Benteng Navarone";
21     novel[0].pengarang = "Alistair McClean";
22     novel[0].jumlah = 4;
23     novel[1].judul = "Octopussy";
24     novel[1].pengarang = "Ian Flemmings";
25     novel[1].jumlah = 2;
26     novel[2].judul = "Badai Pasti Berlalu";
27     novel[2].pengarang = "Marga T.";
28     novel[2].jumlah = 2;
29     novel[3].judul = "Twilight";
30     novel[3].pengarang = "Stephenie Meyer";
31 }
```

Compile Log   Debug   Find Results   Close

- Output Filename: E:\INSTITUT BRI\MATA KULIAH\ALGORITMA DAN STRUKTUR DA
- Output Size: 1.30404281616211 MiB
- Compilation Time: 1.83s



# Pointer dan Array Dinamis

The image shows a C++ IDE with two tabs: `pointerdata.cpp` and `dinamis.cpp`. The `dinamis.cpp` tab is active, displaying the following code:

```
17 // Buat variabel dinamis berupa array dan atur agar ditunjuk oleh novel
18 novel = new RekamanBuku[5];
19 // Isi data melalui novel
20 novel[0].judul = "Meriam Benteng Navarone";
21 novel[0].pengarang = "Alistair McClean";
22 novel[0].jumlah = 4;
23 novel[1].judul = "Octopussy";
24 novel[1].pengarang = "Ian Flemmings";
25 novel[1].jumlah = 2;
26 novel[2].judul = "Badai Pasti Berlalu";
27 novel[2].pengarang = "Marga T.";
28 novel[2].jumlah = 2;
29 novel[3].judul = "Twilight";
30 novel[3].pengarang = "Stephenie Meyer";
31 novel[3].jumlah = 3;
32 novel[4].judul = "Harry Potter Deadly Hallows";
33 novel[4].pengarang = "JK Rowlings";
34 novel[4].jumlah = 4;
35 // Mengakses data melalui variabel pointer novel
36 for (int j = 0; j < 5; j++)
37 {
38     cout << "Judul : " << novel[j].judul << endl;
39     cout << "Pengarang : " << novel[j].pengarang << endl;
40     cout << "Jumlah : " << novel[j].jumlah << endl;
41     cout << endl;
42 }
43 // Bebaskan memori untuk variabel dinamis
44 delete [] novel;
45 getch();
46 }
```

Overlaid on the code is a screenshot of the program's output window, titled `E:\INSTITUT BRI\MATA KULIAH\ALGORI...`. It displays the output for each of the five novels:

```
Judul : Meriam Benteng Navarone
Pengarang : Alistair McClean
Jumlah : 4

Judul : Octopussy
Pengarang : Ian Flemmings
Jumlah : 2

Judul : Badai Pasti Berlalu
Pengarang : Marga T.
Jumlah : 2

Judul : Twilight
Pengarang : Stephenie Meyer
Jumlah : 3

Judul : Harry Potter Deadly Hallows
Pengarang : JK Rowlings
Jumlah : 4
```

At the bottom of the IDE, a status bar shows the following information:

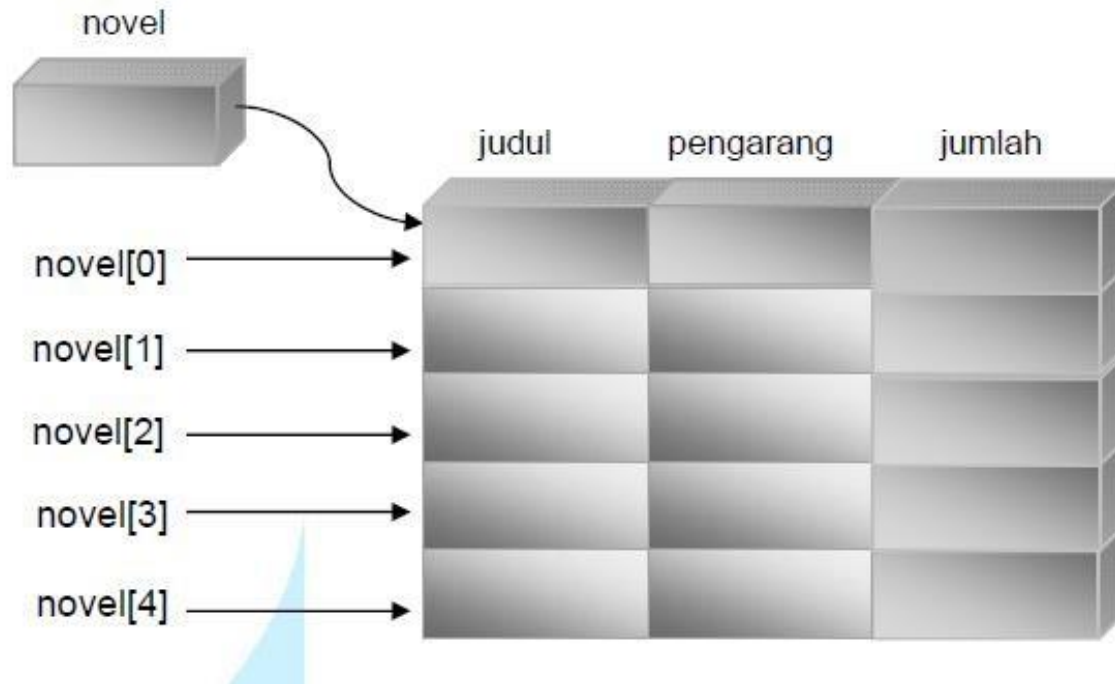
- Compile Log
- Debug
- Find Results
- Close
- Output Filename: E:\INSTITUT BRI\MATA KULIAH\ALGORITMA DAN STRUKTUR DATA\PRAKTIKUM\dinamis.exe
- Output Size: 1.30404281616211 MiB
- Compilation Time: 1.75s

# Pointer dan Array Dinamis

Hal yang terpenting dalam menggunakan array dinamis adalah penyebutan jumlah elemen array dilakukan dengan menuliskan jumlah elemen dalam tanda [ ]. Pada program di atas :

```
novel = new RekamanBuku[5];
```

menyatakan bahwa jumlah elemen array yang diciptakan dan ditunjuk oleh novel sebanyak 5 buah. Gambaran setelah pernyataan di atas dijalankan dapat dilihat pada gambar berikut :



# Pemberian Memori Alokasi Pada Pointer

Sebuah pointer itu tidak memiliki alamat, sehingga pointer harus menumpang pada variabel lain. Namun sekarang kita memberikan alamat kepada variabel pointer sehingga pointer tidak lagi menumpang pada variabel lain. Untuk membuat alamat menggunakan malloc yang disesuaikan dengan panjang data.

```
VariabelPointer = (TipeData *) malloc(sizeof(TipeData));
```

```
pointerdata.cpp  dinamis.cpp  pointer.cpp  AlokasiPointer.cpp
1  #include<iostream>
2  #include<conio.h>
3  #include<malloc.h>
4
5  using namespace std;
6  int main()
7  {
8      int *p;
9
10     p =(int *)malloc(sizeof(int)); //membuat alamat
11
12     *p=5; //deklarasi nilai
13
14     cout <<"isi P "<< *p<<endl;
15     cout <<"alamat P "<<p<<endl;
16 }
```

```
E:\INSTITUT BRI\MATA KULIAH\ALGORITMA DAN STRUKTUR DATA\PR...
isi P 5
alamat P 0x5c6148
```

```
-----
Process exited after 2.748 seconds with return value 0
Press any key to continue . . .
```

# Kesimpulan

---

- 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



Terima  
kasih

# Latihan Soal

1. Apa yang dimaksud dengan pointer pada bahasa C++?
  2. Gambarkan pengaturan memori pemrograman menggunakan pointer?
  3. Bagaimana cara mendeklarasikan pointer?
  4. Bagaimana cara mengubah dan mengisi isi pointer?
  5. Apakah perbedaan array dan pointer?
  6. Apakah perbedaan variable pointer dengan variable dinamis?
- 