BAB II

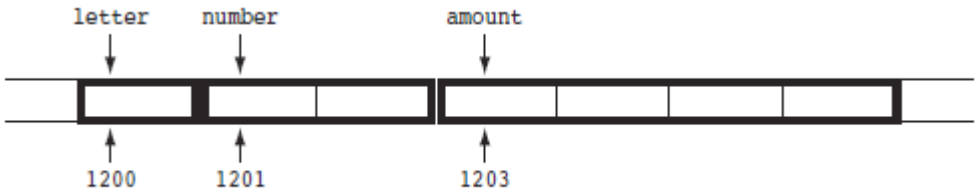
TIPE DATA POINTER

2.1 Pendahuluan

Dalam pembuatan program, setiap variabel yang telah didefinisikan akan disediakan tempat yang cukup untuk menyimpan nilai variabel tersebut di dalam memori komputer. Besarnya tempat yang disediakan tergantung dari tipe data yang disimpan dalam variabel dan komputer yang digunakan. Misalnya untuk komputer pribadi, variabel bertipe integer disediakan tempat sebanyak 2 byte, variabel bertipe real disediakan 4 byte dan variabel bertipe karakter hanya disediakan tempat sebanyak 1 byte saja. Sebagai contoh diberikan deklarasi variabel seperti berikut:

> Variabel letter bertipe karakter, variabel number bertipe integer dan variabel amount bertipe real.

Ilustrasi penyimpanan variabel-variabel tersebut di dalam memori komputer dapat dilihat pada gambar 2.1 di bawah ini. Diasumsikan variabel tersebut disimpan mulai di alamat 1200 dalam memori komputer.



Gambar 2.1 Ilustrasi penyimpanan variabel di dalam memori komputer

Dalam gambar di atas dapat dilihat, bahwa banyaknya tempat yang disediakan untuk suatu variabel sesuai dengan tipe variabelnya. Variabel letter ditempatkan pada alamat memori 1200, variabel number ditempatkan pada alamat memori 1201 dan 1202 (tipe data integer memerlukan 2 byte), dan variabel amount ditempatkan pada alamat memori 1203 sampai 1206 (tipe data real memerlukan 4 byte).

Untuk mengetahui alamat memori tempat variabel, digunakan operator **&** di depan nama variabelnya. Sebagai contoh:

Write (&amount) ==> menghasilkan alamat memori 1203

2.2 Pointer

Pointer adalah tipe data primitif yang menyimpan alamat memori tempat variabel. Variabel bertipe pointer atau disebut pointer saja, adalah variabel yang menyimpan alamat memori, bukan nilai seperti pada variabel yang sudah dibahas sebelum ini. Konsep pointer sebenarnya telah dibahas pada pembahasan sub algoritma yang lalu, yaitu pada pembahasan tentang hubungan antara parameter dan argumen fungsi secara *pass as variable* atau *pass as reference* yaitu dengan "menyerahkan" alamat memori dari argumen ke parameter fungsi. Dengan demikian, perubahan yang terjadi pada parameter fungsi tersebut akan direkam dan dapat dilihat di algoritma utama.

Dalam hal variabel pointer, perubahan nilai data direkam pada alamat memori komputer tempat data tersebut berada, bukan pada variabel pointer-nya. Hal ini sesuai dengan arti harfiah dari pointer yaitu penunjuk atau dalam hal ini, menunjukkan alamat dari data.

2.3 Mendeklarasikan dan Menggunakan Pointer

Sama seperti variabel lainnya, variabel pointer juga harus dideklarasikan dan ditentukan tipe datanya sebelum digunakan. Karena pointer berisi alamat memori komputer (selalu bertipe integer), maka yang dimaksud tipe data pointer adalah tipe dari data yang ditunjuk oleh pointer ini. Sebagai contoh, diketahui x adalah variabel integer, dan y adalah variabel real dan z adalah variabel karakter, maka:

1. pointer P yang berisi alamat memori dari variabel x adalah pointer integer ➔ integer *P
2. pointer Q yang berisi alamat memori dari variabel y adalah pointer real ➔ real *Q
3. pointer R yang berisi alamat memori dari variabel z adalah pointer karakter ➔ karakter *R.

Tanda * di depan nama variabel menunjukkan bahwa variabel tersebut adalah pointer. Sama seperti variabel yang hanya dapat menyimpan data yang tipenya sama dengan tipe data yang telah ditentukan ketika variabel dideklarasikan, maka variabel pointer juga demikan, yaitu hanya dapat digunakan untuk menunjuk data dengan tipe

yang sama. Jadi untuk contoh di atas, pointer P tidak dapat digunakan untuk menunjuk variabel Y atau Z yang berbeda tipe datanya.

Setelah dideklarasikan, pointer dapat digunakan dalam algoritma seperti variabel lainnya. Contoh penggunaan pointer dapat potongan algoritma di bawah ini, jika diketahui ptr adalah variabel pointer yang bertipe integer dan x adalah variabel integer:

1. x = 25
2. ptr = &x → pointer ptr berisi alamat dari variabel x
3. write (x) → output: 25
4. write (ptr) → output: 7a00
5. x = x + 5 → nilai x = …..
6. *ptr = *ptr + 10 → nilai x = …..
7. Write (x) → output:
8. Write (ptr) → output:

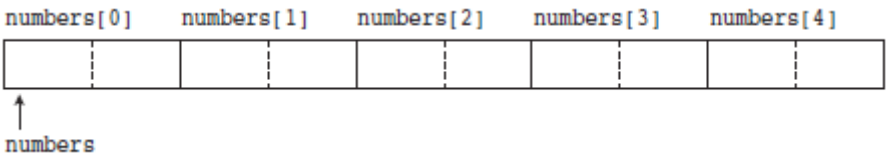
Perhatikan statement di langkah ke 6, statement ini berarti nilai variabel yang alamatnya disimpan oleh pointer ptr (lihat langkah 2), ditambah dengan nilai 10. Dalam hal ini, nilai variabel x akan bertambah 10.

2.4 Hubungan antara Array dan Pointer

Secara tidak langsung nama array dapat digunakan untuk mereferensikan data (komponen) yang disimpan dalam array karena nama array selalu menunjuk pada komponen array yang pertama. Dengan demikian, nama array dapat dianggap sebagai variabel pointer juga. Perhatikan algoritma di bawah ini dan ilustrasi penyimpanan elemen array numbers di memori pada gambar 2.2.

Algoritma Contoh

Algoritma yang menunjukkan bahwa nama array adalah pointer, numbers adalah array bertipe integer.
[deklarasi array]
Integer numbers[5]
1. numbers = {10, 20, 30, 40, 50}
2. write(”Elemen pertama dari array adalah ”)
3. write (*numbers)
4. halt



Gambar 2.2 Ilustrasi penyimpanan array numbers[5]

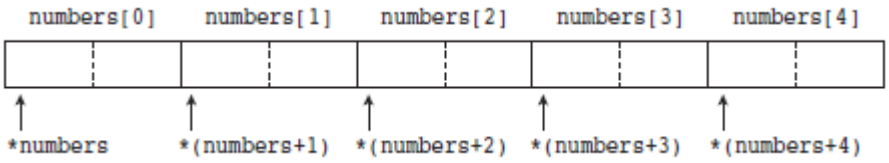
➔output: Elemen pertama dari array adalah 10

Pada gambar 2.2 dapat dilihat bahwa komponen array disimpan secara berurutan mulai dari komponen pertama sampai terakhir. Nama array (numbers) secara otomatis menunjuk pada komponen yang pertama. Dengan demikan statement write (*numbers) pada langkah 3 dari algoritma Contoh akan menghasilkan angka 10. Atau dengan kata lain *numbers adalah pointer yang menunjuk pada komponen array yang pertama.

Jika *numbers adalah pointer yang menunjuk komponen array yang pertama, maka untuk mengakses komponen berikutnya adalah dengan menambahkan pointer tersebut dengan banyaknya memori yang digunakan untuk menyimpan 1 komponen array, dalam hal ini 2 byte (lihat gambar 2.3 dibawah ini). Atau dengan kata lain:

*(numbers+1) ➔ untuk komponen kedua, alamat numbers ditambah 1 komponen, atau (numbers[1]),

*(numbers+2) ➔ untuk komponen ketiga, alamat numbers ditambah 2 komponen, atau (numbers[2]), dan seterusnya



Gambar 2.3 Cara mengakses komponen array dengan pointer

Perhatikan pada gambar di atas, terdapat kesamaan penulisan indeks array dengan banyaknya penambahan komponen pada penulisan array dengan pointer.

*numbers ↔ numbers[0]
*(numbers+1) ↔ numbers[1]
*(numbers+2) ↔ numbers[2]

2.5 Dynamic Memory Allocation

Selama ini, variabel yang digunakan pada algoritma atau program, dideklarasikan di awal algoritma/program dan terus digunakan selama algoritma/program dieksekusi. Sebagai contoh, algoritma untuk menghitung luas persegi panjang, maka diperlukan 3 variabel yaitu variabel untuk menyimpan panjang, lebar dan luas. Algoritma untuk menghitung nilai akhir dari suatu kelas maka diperlukan beberapa variabel array untuk menyimpan komponen data nilai dari seluruh siswa. Pada umumnya, variabel yang akan digunakan sudah dapat diperkirakan ketika program dibuat. Dalam beberapa kasus, sangat sulit untuk memperkirakan banyaknya variabel, misalnya menghitung rata-rata dari beberapa nilai tes untuk sekelompok siswa. Disini tidak diketahui banyaknya nilai tes dan banyaknya siswa sehingga sulit untuk menentukan banyaknya variabel dan juga tipe datanya.

Untuk mengatasi masalah ini, beberapa bahasa pemrograman menyediakan metode untuk mengalokasikan memori kapan saja, dan dimana saja diperlukan termasuk tipe dan banyaknya data. Metode ini disebut pengalokasian memori secara dinamik atau Dynamic Memory Allocation dengan menggunakan pointer. Dengan cara ini, programmer tidak perlu bersusah payah memikirkan seluruh variabel yang akan digunakan di awal program, tetapi dapat mengalokasikan variabel ketika diperlukan dan menghapusnya jika sudah tidak diperlukan. Untuk mengalokasikan memori secara dinamik digunakan perintah:

```
tipe_data *nama_variabel
nama_variabel = new tipe_data
```

Pada perintah di atas, deklarasikan lebih dulu pointer (baris pertama). Lalu, jika diperlukan baru dialokasikan memori untuk pointer tersebut (baris kedua). Perintah **new** pada baris kedua berarti akan dialokasikan memori untuk variabel tersebut sesuai dengan tipe datanya. Sebagai contoh lihat potongan algoritma berikut ini:

```
Integer *ptr
1. ptr = new integer
2. *ptr = 25
3. Write (”Isi variabel = ”, *ptr)          ➔ output: .....
4. Read (*ptr)                              ➔ input:  .....
5. Write (”Isi variabel = ”, *ptr)          ➔ output: .....
6. Total = *ptr * 2
7. Write (”Isi variabel total = ”, total)   ➔ output: .....
```

Selain untuk mengalokasikan variabel tunggal, perintah **new** di atas juga dapat digunakan untuk mengalokasikan sekelompok memori untuk array secara dinamik. Sebagai contoh, pointer ptr digunakan untuk mengalokasikan array dengan 100 komponen:

```
ptr = new integer[100]
```

Setelah dialokasikan, selanjutnya array **ptr[ ]** dapat diproses seperti biasanya.

Ketika suatu variabel dialokasikan secara dinamik, maka variabel tersebut dapat dihapus dengan perintah **delete** jika tidak diperlukan lagi keberadaannya. Contoh perintah untuk menghapus keberadaan variabel dinamik:

```
delete ptr        ➔ menghapus keberadaan pointer ptr
delete [ ] ptr    ➔ menghapus keberadaan array ptr
```

Setelah perintah di atas, variabel ptr tidak dapat diakses lagi. Jadi pastikan lebih dulu variabel-variabel tersebut tidak digunakan lagi dalam program.

2.6 Pointer sebagai Parameter Fungsi

Seperti telah dibahas sebelumnya, salah cara untuk menghubungkan parameter fungsi dengan argumen fungsi adalah dengan cara *pass as variable* atau *pass as a reference*. Selain dengan menggunakan variabel biasa, dapat juga digunakan variabel pointer sebagai parameter fungsi. Sebagai contoh dapat dilihat pada fungsi dan algoritma di bawah ini. Fungsi Angka dan Dobel menggunakan cara *pass as variable* dengan parameter yang bertipe pointer.

```
Void Angka(integer *input)
Fungsi untuk memasukkan angka integer. Variabel input adalah pointer yang bertipe
integer
 {
 1. Write (”Masukkan sebuah angka integer:  ”)
 2. Read (*input)
 }

 Void Dobel (integer *nilai)
 Fungsi untuk mengalikan dengan  2. Variabel nilai adalah pointer bertipe integer
 {
    *nilai = *nilai * 2
 }

 Algoritma Referensi
```

Algoritma yang menunjukkan penggunaan variabel pointer untuk menghubungkan argumen dan parameter dengan cara pass as reference. Fungsi angka untuk menerima data integer, fungsi dobel untuk mengkalikan data dengan 2. Kedua fungsi ini menggunakan pointer sebagai parameternya. Variabel bil bertipe integer.

```
[deklarasi fungsi]
Void Angka (integer *)
Void Dobel (integer *)
1. Angka(&bil)
2. Dobel (&bil)
3. Write (”Angka setelah dikalikan  2 = ”, bil )
4. Halt
```

Output dari algoritma di atas:

```
Masukkan sebuah angka integer : 5
Angka setelah dikalikan 2 : 10
```

Perhatikan pemanggilan fungsi di langkah 1 dan 2. Dalam definisi dan deklarasi fungsi digunakan parameter yang bertipe pointer, tetapi dalam pemanggilan fungsi, digunakan *reference argument*. Dengan demikian, dapat disimpulkan bahwa variabel pointer dapat digantikan *reference variable*.

Selain sebagai parameter input, pointer juga dapat digunakan sebagai parameter output dengan catatan bahwa data yang ditunjuk oleh pointer tersebut masih tetap ada setelah fungsi dieksekusi. Untuk jelasnya dapat dilihat pada algoritma berikut ini.

```
Fungsi *Bil_Acak (num)
Fungsi yang mengembalikan pointer ke array yang berisi bilangan acak. Parameternya
menunjukkan banyaknya bilangan acak yang akan dibangkitkan lalu disimpan ke array.
Semua variabel dan array bertipe integer. Pointer Ar bertipe integer.
{
    1. If (num <= 0)
           Return NULL
    2. [mengalokasikan array Arr secara dinamik]
       Ar = new integer[num]
    3. [memanggil fungsi Srand() untuk membangkitkan awal bilangan acak]
       Srand (time(0))
    4. [memanggil fungsi rand() untuk membangkitkan bilangan acak sebanyak num]
       For ( c = 0 ; c < num : c++)
       {
       Ar[c] = rand()
       }
    5. Return (Ar)
}

Algoritma Demo
Contoh algoritma dengan fungsi yang mengembalikan pointer.
[deklarasi pointer]
```

```
Integer *bil
1. [memanggil fungsi Bil_Acak]
   Bil = Bil_Acak(5)
2. [menampilkan isi array]
   For ( c = 0 ; c < num : c++)
   {
      Write (Bil[c])
   }
3. [menghapus array]
    Delete [] bil
4. halt
```

LATIHAN SOAL BAB 2

1. Tuliskan hasil dari statement berikut ini:

a. Algoritma soal_1a
   Variabel x, y dan z adalah integer, ptr adalah pointer integer
   1. x = 50; y = 60; z = 70
   2. ptr =&x
   3. *ptr *= 10
   4. ptr =&y
   5. *ptr *= 5
   6. ptr =&y
   7. *ptr *= 2
   8. Write (“Nilai x = “, x, “ Nilai y = “, y, “ Nilai z = “, z)
   9. Halt

b. Algoritma soal_1b
   Variabel x, y dan z adalah integer, ptr adalah pointer integer
   1. x = 25; y = 50; z = 75
   2. ptr =&x
   3. *ptr += 10
   4. ptr =&y
   5. *ptr += 5
   6. ptr =&y
   7. *ptr += 2
   8. Write (“Nilai x = “, x, “ Nilai y = “, y, “ Nilai z = “, z)
   9. Halt

2. Tentukan apakah statement berikut ini benar atau salah, jika salah jelaskan apa yang salah lalu perbaiki.

2.a.
```
integer vrb
Integer *ptr = &vrb
```
2.b.
```
float vrb
Integer ptr = &vrb
```
2.c.
```
integer num[50]
*ptr = num
```

2.d.
```
integer *ptr = &vrb
Integer vrb;
```

3. Buat fungsi untuk mengalokasikan array integer secara dinamik. Fungsi menerima angka sebagai argument yang menunjukkan banyaknya elemen array. Fungsi menghasilkan pointer ke array tersebut

3.b. Buat pula algoritma utama yang memanggil fungsi, mengisi komponen array dan menampilkan hasilnya.

4.a. Dalam statistik, modus adalah himpunan nilai yang sering muncul sehingga mempunyai nilai frekuensi yang besar. Buatlah fungsi yang menerima array integer sebagai argument-nya lalu menentukan modus dari array. Jika array input tidak mempunyai modus, maka dikembalikan nilai -1. Gunakanlah notasi pointer untuk array.

4.b. Buat pula algoritma utamanya

5.a. Buatlah fungsi yang menerima array integer lalu mengkopi array tersebut sehingga elemen array hasil kopi mempunyai urutan kebalikan dari elemen array semula. Fungsi menghasilkan pointer ke array hasil kopi

5.b. Buat pula algoritma utamanya.

6.a. Buatlah fungsi yang menerima array integer lalu membuat array baru yang kapasitasnya 2 kali dari array semua. Fungsi akan mengkopi array mula-mula ke array yang baru dan sisa elemen di array baru diisi dengan angka 0.

6.b. Buat pula algoritma utamanya.

7.a. Buatlah fungsi yang menerima array integer lalu membuat array baru yang berisi penggeseran elemen dari array semula. Fungsi akan mengisi angka 0 di indeks 0 array baru lalu mengkopi elemen array lama indeks 0 ke indeks 1 array baru, mengkopi elemen array lama indeks 1 ke indeks 2 array baru dan seterusnya sampai semua elemen array lama dikopi ke array baru.

7.b. Buat pula algoritma utamanya.