

ARRAY, STRING, DAN POINTER

OBJEKTIF :

1. Mahasiswa mampu memahami tentang array.
2. Mahasiswa mampu memahami tentang string.
3. Mahasiswa mampu memahami tentang pointer.

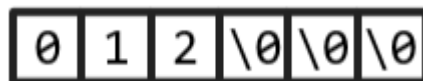
5.1 ARRAY

Array adalah kumpulan data yang memiliki tipe data yang sama yang ditampung di dalam nama variabel yang sama. Dengan array, data yang banyak akan ditampung di dalam penyimpanan yang disebut dengan variabel array. Nilai atau elemen atau data di dalam array dipisahkan dan diakses dengan subscript atau indeks. Indeks pada array diawali dengan `[0]` bukan `[1]`. Di bawah ini merupakan penggambaran tentang array:

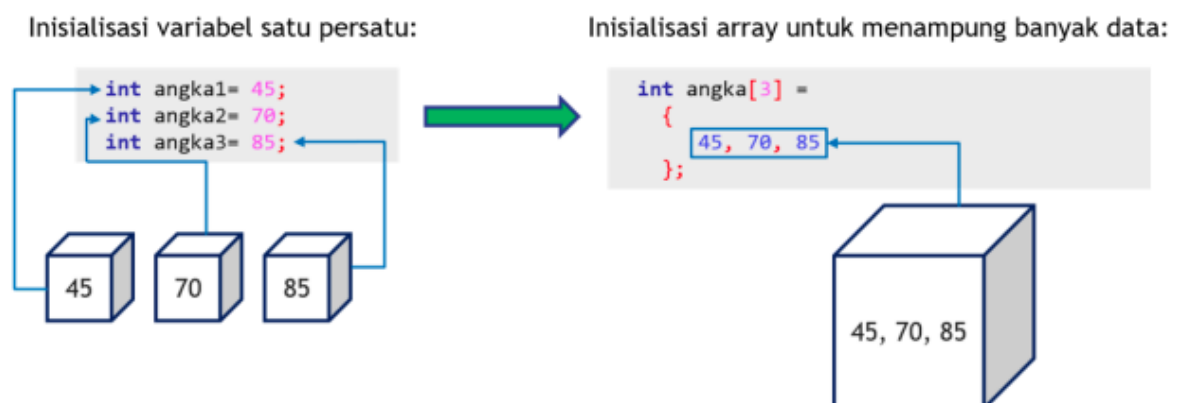
Diketahui pada suatu program terdapat deklarasi dan inisialisasi variabel array seperti di bawah ini:

```
int bil[6] =  
{  
    0, 1, 2  
};
```

Maka pengalokasian nilai di dalam memorinya adalah seperti di bawah ini:



Mengapa kita memerlukan array untuk menampung banyak data? Perhatikan gambaran di bawah ini:



Pada gambar di sebelah kiri ada banyak variabel yang diinisialisasikan untuk menampung suatu data, sedangkan pada gambar di sebelah kanan terdapat variabel array yang diinisialisasikan untuk ditugaskan menampung banyak data. Berdasarkan gambaran di atas, terlihat bahwa variabel array mempunyai manfaat dalam meminimalisir ukuran penyimpanan. Bayangkan jika kita menggunakan cara menginisialisasi 1000 angka dengan variabel secara satu persatu, tentu akan memakan banyak memori.

Pemberian nilai terhadap suatu variabel array dapat dilakukan dengan dua cara, diantaranya:

- Diinput oleh user (menggunakan perintah `scanf`)
- Diinisialisasikan pada program

Ketika suatu nilai telah berada di dalam variabel array, langkah selanjutnya adalah mengakses nilai-nilai tersebut. Untuk mencetak seluruh elemen atau nilai yang ditampung pada variabel array, dapat dilakukan dengan dua metode. Yang pertama adalah dengan memanggil secara satu persatu elemen di dalam array, lalu langkah yang kedua adalah dengan melakukan perulangan. Untuk itu kita perlu mengerti tentang alamat indeks suatu array. Karena, jika kita salah dalam menuliskan alamat indeks suatu nilai array, maka nilai yang ingin kita akses tidak akan sesuai dengan nilai yang ditampilkan pada output

Terdapat macam-macam array, diantaranya:

- Array satu dimensi
- Array dua dimensi
- Array multi dimensi

5.1.1 ARRAY SATU DIMENSI

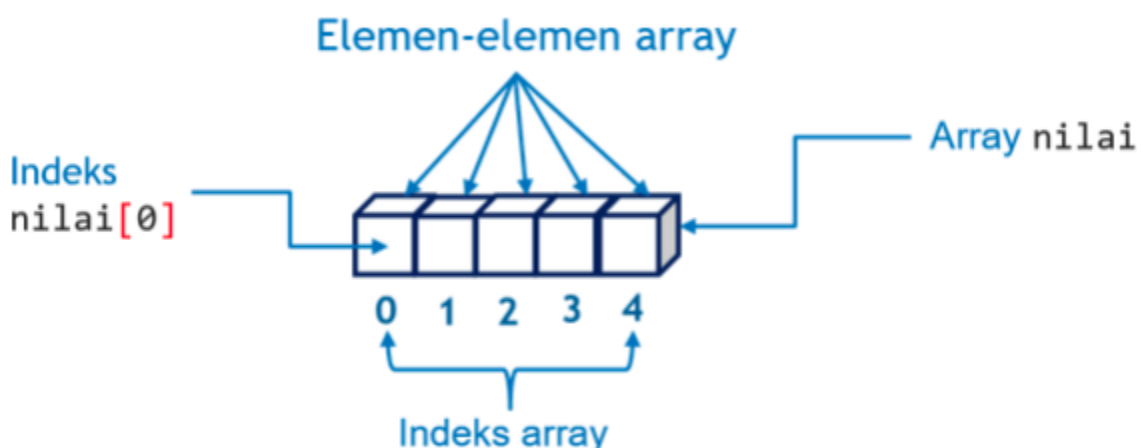
Array satu dimensi merupakan jenis array yang hanya mempunyai satu kurung indeks saja. Deklarasi variabel array berbeda dengan deklarasi variabel lainnya. Dalam mendeklarasikan atau menginisialisasi variabel array satu dimensi, diperlukan tambahan indeks `[indeks]` yang ditulis setelah nama variabelnya. Di bawah ini merupakan bentuk umum dari array satu dimensi:

tipeData namaVariabelArray[indeks];

Contohnya seperti di bawah ini:



Jika terdapat deklarasi variabel array seperti di atas, maka akan diciptakan suatu variabel array bernama `nilai` yang menampung lima elemen atau lima nilai di dalamnya, yang mana nilai-nilai tersebut bertipe data `integer`. Mari lihat gambar di bawah ini untuk melihat bagaimana array satu dimensi bekerja:



Jika pada suatu program dideklarasikan array seperti `int nilai[5];`, maka pengalokasian nilainya seperti di atas. Array diumpamakan sebagai *storage* atau penyimpanan, dimana masing-masing kotak penyimpanan ditandai dengan indeks dan dapat menampung nilai berbeda antar indeks.

Di bawah ini merupakan program `arraysatudimensi.c` yang menampung konsep array satu dimensi di dalamnya:

```
#include <stdio.h>

int main()
{
    char karakter[9] =
    {
        'G', 'U', 'N', 'A', 'D', 'A', 'R', 'M', 'A'
    };

    // Tampilkan isi dari variabel array
    printf ("Elemen yang ditampilkan adalah :\n%c", karakter[2]);

    return 0;
}
```

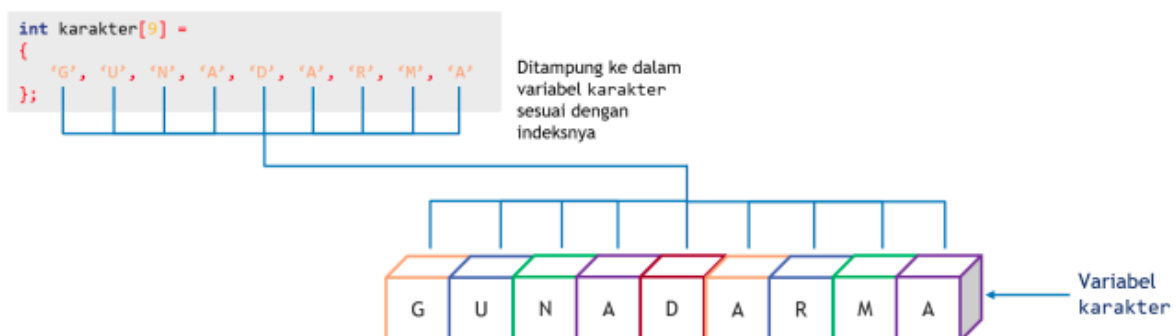
Output program `arraysatudimensi.c`:

```
Elemen yang ditampilkan adalah :
N
```

Pada program, elemen yang dideklarasikan pertama adalah huruf `G`, maka secara otomatis elemen `G` akan ditempatkan di indeks `karakter[0]`, kemudian elemen `U` akan diletakkan di indeks `karakter[1]`, dan elemen `N` akan diletakkan di indeks `karakter[2]`, dan seterusnya. Perlu diingat bahwa indeks array akan dimulai dari indeks ke `[0]` bukan ke `[1]`. `karakter[9]` merupakan deklarasi yang menyatakan bahwa elemen pada variabel `karakter` akan menampung sebanyak 9 nilai.

Setelah itu di bawah terdapat perintah untuk mencetak elemen yang ditampung di dalam indeks kedua dari variabel `karakter`. Maka, elemen yang akan dicetak ke layar adalah huruf `N` bukan `U`. Karena, elemen yang berada di variabel `karakter` di indeks kedua adalah `N`.

Di bawah ini merupakan gambaran bagaimana variabel array `karakter` menempatkan elemen-elemennya:



5.1.1.1 Array dengan elemen kosong

Bagaimana jika kita ingin memberi nilai kosong pada indeks [0] sampai indeks ke [9] ? Di bawah ini merupakan cara menginisialisasi variabelnya:

```
#include <stdio.h>

int main()
{
    char karakter[9] = {0};

    // Tampilkan isi dari variabel array
    printf ("Elemen yang ditampilkan adalah :\n%c", karakter[2]);

    return 0;
}
```

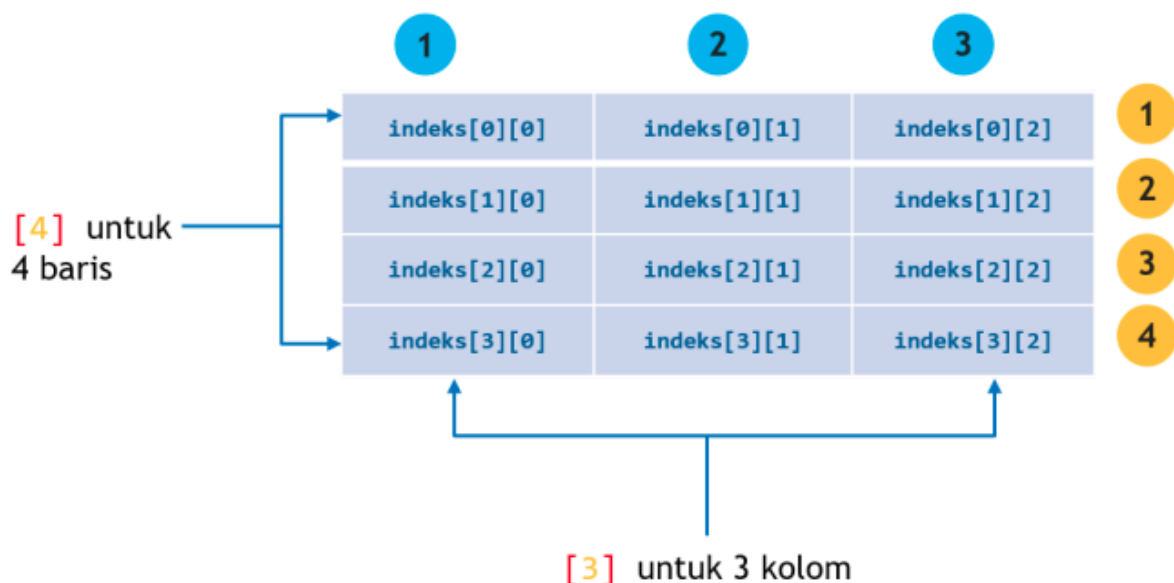
Berikut merupakan output yang akan ditampilkan apabila program di atas dijalankan:

Elemen yang ditampilkan adalah :

Meskipun kita memanggil nilai yang ditampung pada indeks `karakter[1]`, nilai yang akan tercetak tetap kosong. Karena, jika kita menuliskan angka `0` pada saat inisialisasi variabel array, secara otomatis akan mengisi nilai `0` pada masing-masing indeks array.

5.1.2 ARRAY DUA DIMENSI

Setelah mempelajari array satu dimensi, pada Bahasa C, juga terdapat array dua dimensi. Array dua dimensi merupakan jenis array yang mempunyai dua indeks yang perlu dideklarasikan. Array dua dimensi dapat didefinisikan seperti tabel yang mempunyai dimensi baris dan dimensi kolom. Berikut penggambarannya:



Berdasarkan gambaran di atas, maka dapat disimpulkan di dalam array dua dimensi diperlukan dua indeks yang perlu dideklarasikan. Berikut merupakan bentuk umum dari deklarasi variabel array dua dimensi:



Jika di dalam suatu program terdapat potongan program seperti di bawah ini:

```
char *dataMhs[2][3] =
{
    {"Atika","36118987", "A+"},
    {"Kayla","36118986", "A"}
};
```

Berada di
dataMhs[0][0]

Atika	36118987	A+
Kayla	36118986	A

Jika diinisialisasikan variabel `dataMhs` seperti di atas, maka kita diminta untuk membuat suatu variabel array `dataMhs` yang mana variabel tersebut menampung nama mahasiswa, NPM, dan nilainya. Dimana data-data tersebut disusun dalam bentuk tabel 2 x 3.

Array identik dengan pengalamatan. Suatu nilai pada array dapat diakses apabila alamat yang ditulis pada saat pemanggilan variabel array ditulis dengan benar. Membahas soal alamat pada array, bagian yang ditunjuk oleh anak panah merupakan elemen `Atika` yang berada di indeks `dataMhs[0][0]`.

Di bawah ini merupakan program `arrayduadimensi.c`:

```
#include <stdio.h>

int main()
{
    printf("Nama\t\tNPM\t\tJurusan\n");
    char *dataMhs[3][3] =
    {
        {"Atika","36118987", "Akuntansi"},
        {"Kayla","36118986", "Teknik Informatika"},
        {"Andrean","36118985", "Manajemen"}
    };
    // Tampilkan data mahasiswa
    int i, j;
    for (i = 0; i < 3; i++) // Perulangan untuk indeks baris
```

```

{
    for (j = 0; j < 3; j++) // Perulangan untuk indeks kolom
        printf("%s\t\t", dataMhs[i][j]);

    printf("\n");
}

return 0;
}

```

Output program `arrayduadimensi.c`:

Nama	NPM	Jurusan
Atika	36118987	Akuntansi
Kayla	36118986	Teknik Informatika
Andrean	36118985	Manajemen

Pada program di atas, diinisialisasikan variabel `dataMhs` untuk ditugaskan menampung data berbentuk 2 dimensi dan dimensinya adalah 3 baris dan 3 kolom. Selain itu, terdapat perulangan bersarang. Perulangan pertama merupakan perulangan yang digunakan untuk melakukan iterasi terhadap variabel `i` dan perulangan kedua merupakan perulangan yang digunakan untuk melakukan iterasi terhadap variabel `j`.

Variabel `i` digunakan untuk melakukan iterasi untuk indeks baris. Artinya, jika kondisi dalam perulangan `for i` ini terpenuhi, maka akan menjalankan perulangan di dalamnya yaitu perulangan `for j`. Sedangkan variabel `j` digunakan untuk melakukan iterasi untuk indeks kolom. Artinya, jika kondisi dalam perulangan `for j` ini terpenuhi, maka akan menjalankan perintah di dalamnya yaitu perintah untuk mencetak nilai dari array berdasarkan indeks baris (`[i]`) dan indeks kolomnya (`[j]`).

Variabel `i` digunakan untuk melakukan iterasi untuk indeks baris. Artinya, jika kondisi dalam perulangan `for i` ini terpenuhi, maka akan menjalankan perulangan di dalamnya yaitu perulangan `for j`. Sedangkan variabel `j` digunakan untuk melakukan iterasi untuk indeks kolom. Artinya, jika kondisi dalam perulangan `for j` ini terpenuhi, maka akan menjalankan perintah di dalamnya yaitu perintah untuk mencetak nilai dari array berdasarkan indeks baris (`[i]`) dan indeks kolomnya (`[j]`).

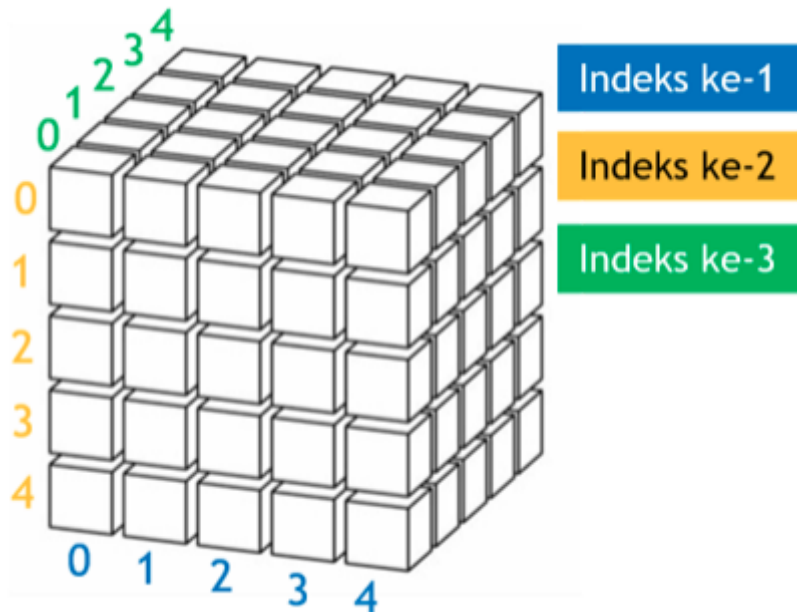
Ketika program ini dijalankan, dan tahap perulangan dilakukan, perulangan pertama yang akan dijalankan adalah perulangan variabel `i`. Dimana nilai `i` awal adalah `0`, dan kondisi pada perulangan `i` terpenuhi karena kondisi yang ditulis pada perulangan adalah `i < 3`. `0` ini merupakan indeks untuk baris pertama. Langkah selanjutnya adalah melakukan iterasi kolom atau perulangan `for j`. Nilai awal dari variabel `j` adalah `0`. `0` merupakan nilai yang memenuhi kondisi untuk melakukan perintah di dalam `for j`. Dikarenakan kondisi perulangan `j` terpenuhi, maka langkah selanjutnya adalah mencetak nilai dari variabel array `dataMhs` yang berada di indeks `dataMhs[0][0]` yaitu `Atika`. Karena `Atika` merupakan nilai yang berada di indeks `dataMhs[0][0]`.

Dilanjut dengan melakukan iterasi kedua pada variabel `j`, yang mana nilai `j` berubah menjadi `1` dikarenakan adanya operator *increment*. Dan selanjutnya menjalankan perintah untuk mencetak nilai dari variabel `dataMhs` kembali. Nilai yang dicetak adalah `36118987`. Karena, nilai `36118987` berada di indeks `dataMhs[0][1]` atau berada di dalam baris pertama, kolom kedua. Dan cara yang sama untuk mencetak nilai dari `dataMhs[0][2]`. Ketika nilai `j` bertambah menjadi `3` dan iterasi berhenti, maka langkah selanjutnya adalah menjalankan perulangan pertama yang mana

nilai **i** berubah menjadi **1**. Perulangan ini digunakan untuk mencetak dan memposisikan nilai sesuai dengan indeks baris dan kolomnya.

5.1.3 ARRAY MULTI DIMENSI

Array multi dimensi merupakan jenis array yang memiliki lebih dari satu indeks. Array tiga dimensi termasuk di dalamnya. Array tiga dimensi memiliki tiga indeks yang perlu di deklarasikan. Di bawah ini merupakan gambaran tentang array tiga dimensi:



Array tiga dimensi sering digambarkan sebagai bentuk kubus. Karena, kubus memiliki tiga dimensi yaitu dimensi panjang, dimensi lebar, dan dimensi tinggi. Di bawah ini merupakan bentuk umum dari deklarasi array tiga dimensi:

tipeData namaVarArray[**indeks1**][**indeks1**][**indeks1**];

Di bawah ini merupakan contoh inisialisasi suatu variabel dengan nilai array tiga dimensi di dalamnya:

```
char *dataMhs[2][3][3] =
{
    {"Atika","36118987", "Akuntansi"},
    {"Kayla","36118986", "Teknik Informatika"},
    {"Andrean","36118985", "Manajemen"}},
    {"Layla","36118984", "Perbankan"},
    {"Firman","36118983", "Kedokteran"},
    {"Jonathan","36118982", "Farmasi"},}
};
```

```
char *dataMhs[2][3][3] =
{
    {"Atika", "36118987", "Akuntansi"},
    {"Kayla", "36118986", "Teknik Informatika"},
    {"Andrean", "36118985", "Manajemen"}},
    {"Layla", "36118984", "Perbankan"},
    {"Firman", "36118983", "Kedokteran"},
    {"Jonathan", "36118982", "Farmasi"}}
};
```

Kurung kurawal yang diberi warna highlight kuning merupakan pengelompokkan data untuk indeks pertama yaitu [2]. Dimana pada masing-masing kurung kurawal, terdapat 3 kelompok data lagi di dalamnya yaitu data-data yang diberi warna highlight hijau. Selanjutnya, pada kurung kurawal yang diberi warna highlight hijau merupakan pengelompokkan data untuk indeks kedua yaitu [3]. Ada 3 kelompok data yang berada di dalam indeks pertama. Yang terakhir, tulisan berwarna biru merupakan data yang terdapat di dalam indeks kedua. Ada 3 data di dalam kurung berwarna hijau, untuk itu dideklarasikan indeks [3] pada indeks ketiga.

Di bawah ini merupakan program `arraytigadimensi.c` dengan menggunakan konsep array tiga dimensi:

```
#include <stdio.h>

int main()
{
    char *dataMhs[2][3][3]=
    {
        {"Atika", "Ferd", "Budi"},
        {"Kayla", "Jesica", "Michael"},
        {"Andrean", "Karina", "Julian"}}
        {"Layla", "Farida", "Meily"},
        {"Firman", "Anita", "Gita"},
        {"Jonathan", "Lola", "Fikri"}}
    };
    // Tampilkan data mahasiswa
    int i, j, k;
    printf("\nNama-nama peserta lolos seleksi asisten komputer :\n");
    for (i = 0; i < 2; i++)
        for (j = 0; j < 3; j++)
        {
            printf("\n");
            for (k = 0; k < 3; k++)
                printf("Hari ke-%d | Sesi ke-%d = %s\n", i+1, j+1, dataMhs[i][j][k]);
        }

    return 0;
}
```

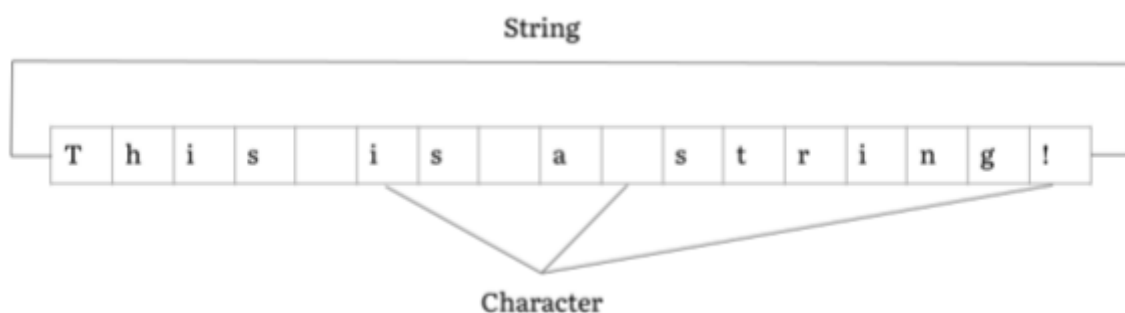
Dan berikut adalah output yang ditampilkan:



Konsep kerja program di atas sama dengan konsep kerja array dua dimensi. Dimana perulangan yang dilakukan terlebih dahulu adalah perulangan terluar, dan lanjut ke perulangan di dalamnya. Contohnya, nilai `Atika` berada di dalam indeks `dataMhs[0][0][0]`.

5.2 STRING

String merupakan array satu dimensi yang terdiri dari kumpulan karakter dan diakhiri dengan karakter kosong (null). String dapat digunakan untuk banyak hal, salah satunya untuk menampilkan pesan kesalahan (`error`) ke layar. Seperti halnya tipe data lain string dapat berupa konstanta dan variabel.



5.2.1 KONSTANTA DAN VARIABEL STRING

Programmer sudah sering menggunakan string sebagai konstanta, salah satu penggunaannya adalah seperti berikut:

```
printf("Hello");
```

Konstanta string seperti contoh di atas akan disimpan dalam memori secara berurutan seperti berikut:



Setiap karakter pada konstanta string akan menempati memori berukuran satu byte dan setelah karakter terakhir akan terdapat null. String yang hanya berisi karakter null disebut string kosong. Di bawah ini merupakan contoh deklarasi string:

```
char teks[10];
```

Pada contoh diatas, terdapat *statement* mendeklarasikan variabel string dengan panjang ukuran 10 karakter. *Statement* di atas juga merupakan *statement* untuk mendeklarasikan array satu dimensi, karena string merupakan array satu dimensi. Inisialisasi string sama seperti array satu dimensi seperti berikut:

```
char teks[]={ 'H', 'e', 'l', 'l', 'o', '\0' };
```

Inisialisasi string juga dapat dilakukan seperti berikut:

```
char teks[] = "Hello";
```

5.2.2 FUNGSI `toupper()` PADA STRING

Nilai string pada variabel array dapat dimodifikasi. Salah satunya dengan fungsi `toupper()`. `toupper()` berfungsi untuk memperoleh atau mengkonversi huruf kecil pada nilai suatu variabel array menjadi huruf kapital. *Return value* dari `toupper()` akan berupa seperti argumennya, apabila argumen berisi huruf kapital. Untuk mendapat gambaran tentang cara kerja fungsi `toupper()`, perhatikan program di bawah ini:

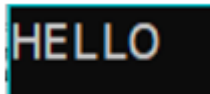
```
#include <stdio.h>
int main()
{
    char st[] = "hello";

    int i;
    for (i = 0; st[i]; i++)
        st[i] = toupper(st[i]);

    printf("%s\n", st);

    return 0;
}
```

Di bawah ini merupakan output dari program di atas:



Dengan fungsi `toupper()`, variabel `st[]` yang awalnya mengandung kata `hello` dan setiap hurufnya merupakan huruf kecil, menjadi huruf kapital saat ditampilkan di layar.

5.2.3 FUNGSI `tolower` PADA STRING

Fungsi `tolower()` merupakan kebalikan dari fungsi `toupper()`. `tolower()` berfungsi untuk memperoleh atau mengkonversi huruf kapital pada nilai suatu variabel array menjadi huruf kecil. *Return value* dari `tolower()` akan berupa seperti argumennya, apabila argumen berisi huruf kecil. Untuk mendapat gambaran tentang cara kerja fungsi `tolower()`, perhatikan program di bawah ini:

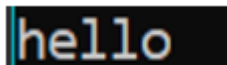
```
#include <stdio.h>
int main()
{
    char st[] = "HELLO";

    int i;
    for (i = 0; st[i]; i++)
        st[i] = tolower(st[i]);

    printf("%s\n", st);

    return 0;
}
```

Output program:



`tolower()` berfungsi untuk mengubah nilai huruf kapital pada nilai di dalam suatu variabel array menjadi huruf kecil.

5.3 POINTER

Pointer merupakan fitur yang memungkinkan pengaksesan data melalui variabel lain. Pointer berisi alamat suatu data. Untuk itu pointer berfungsi untuk mengakses data yang berada di tempat lain. Setiap byte di dalam memori komputer memiliki alamat. Alamat memori dimulai dari 0. Di dalam memori inilah suatu variabel disimpan. Alamat suatu variabel dapat diketahui dengan cara menambahkan operator alamat, yaitu simbol ampersand (&). Dengan mengirimkan ke `printf()`, alamat suatu variabel akan ditampilkan ke layar.

Untuk menyatakan pointer (alamat) pada `printf()`, dapat menggunakan kode format `%p`. Contohnya:

```
printf("alif = %p\n",&alif);
```

Suatu variabel pointer dideklarasikan dengan bentuk seperti berikut:



Di bawah ini merupakan contoh deklarasi pointer:

```
int *pint;  
char *pch;  
float *pfl;
```

Dengan mengisi nilai variabel pointer dengan variabel lain berarti menunjuk variabel pointer ke variabel lain. Contoh:

```
int vint = 55;  
int *pint;
```

Diinisialisasikan variabel `vint` untuk ditugaskan dengan nilai `55`. Kemudian, dideklarasikan variabel pointer `*pint`. Setelah itu jika ditambah dengan *statement* di bawah ini:

```
pint = &vint;
```

Maka, variabel pointer `pint` akan menunjuk variabel `vint`. Untuk mengakses nilai suatu variabel nilai `vint` dapat langsung menggunakan pointer dengan menuliskan `*pint` pada program.

5.3.1 POINTER DAN ARRAY

Pointer dan array dapat terlibat dalam suatu program yang sama. Suatu pointer di dalam program dimaksudkan untuk menunjuk ke suatu alamat memori. Nilai pada variabel array dapat diakses dengan pointer. Seperti contoh:

```
int tgl_lahir[] = { 24, 6, 1965};  
int *ptgl;
```

Agar suatu variabel pointer menunjuk ke suatu variabel array, diperlukan *statement* berupa:

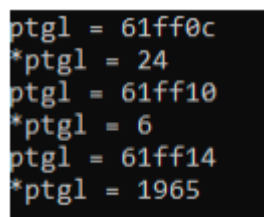
```
ptgl = tgl_lahir;
```

Dikarenakan suatu variabel array sudah menyatakan alamat, maka saat menunjuk variabel array untuk mengisi variabel pointer, tidak lagi diperlukan karakter *ampersand* (&).

Perhatikan program di bawah ini:

```
#include <stdio.h>  
  
int main()  
{  
    int tgl_lahir[] = { 24, 6, 1965};  
    int *ptgl;  
  
    ptgl = tgl_lahir; // ptgl menunjuk ke array  
  
    // Menampilkan isi array via pointer  
    int i;  
    for(i = 0; i < sizeof(tgl_lahir)/sizeof(int); i++)  
    {  
        printf("ptgl = %x\n", ptgl);  
        printf("*ptgl = %d\n", *ptgl);  
        ptgl++;  
    }  
  
    return 0;  
}
```

Output program:



```
ptgl = 61ff0c  
*ptgl = 24  
ptgl = 61ff10  
*ptgl = 6  
ptgl = 61ff14  
*ptgl = 1965
```

Pada output terdapat perbedaan. Jika kita hanya memanggil variabel `ptgl` tanpa menggunakan simbol asterisk (*), maka output yang ditampilkan adalah alamat dari suatu nilai yang ada pada variabel array `tgl_lahir`. Sedangkan jika kita menggunakan simbol *, maka nilai yang ditampilkan adalah nilai suatu variabel array atau nilai dari pointer.

5.3.2 POINTER DAN STRING

Perhatikan program di bawah ini:

```
char *ptokoh = "Gatotkaca";
```

C akan mengalokasikan sebagai variabel pointer yang merujuk ke data bertipe `char` dan menempatkan string `Gatotkaca` ke suatu lokasi di memori. `ptokoh` akan menunjuk ke lokasi string `Gatotkaca`.

Statement di atas serupa dengan *statement* berikut:

```
char tokoh[] = "Gatotkaca";
```

Perbedaan contoh di atas dan di bawah adalah pada contoh pertama dalam mendeklarasikan nilainya menggunakan pointer, sedangkan pada contoh kedua menggunakan array.

Perhatikan program di bawah ini:

```
#include <stdio.h>

int main()
{
    char tokoh[] = "Gatotkaca";
    char *ptokoh = "Gatotkaca";

    printf("tokoh = %s\n", tokoh);
    printf("ptokoh = %s\n", ptokoh);

    //tokoh++ tidak diperkenankan
    ptokoh++;
    printf("ptokoh = %s\n", ptokoh);

    return 0;
}
```

Output program:

```
tokoh = Gatotkaca
ptokoh = Gatotkaca
ptokoh = atotkaca
```

Pada output terdapat perbedaan pada nilai variabel `ptokoh`. Sebelum mengalami *increment*, nilai awal dari `ptokoh` adalah `Gatotkaca`, lalu setelah terdapat operator increment `ptokoh++`, nilai dari variabel `ptokoh` berubah menjadi `atotkaca`. Hal ini menunjukkan bahwa pointer dapat diubah dengan mudah.

REFERENSI

- [1] Abdul Kadir. 2015. *From Zero to a Pro*. Yogyakarta. Andi