Pertemuan 4

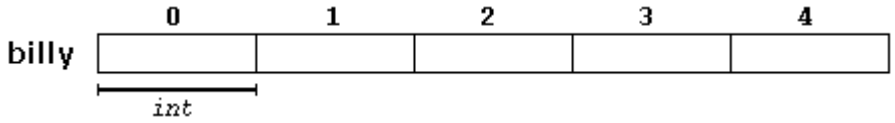
4. Arrays

Obyektif Praktikum :

1. Memahami penggunaan Array
2. Dapat menerapkan array dengan elemen multidimensi
3. Memahami penggunaan statement pada character dan string

P. 4.1 Array

Array adalah himpunan elemen (variable) dengan tipe yang sama dan disimpan secara berurutan dalam memory yang ditandai dengan memberikan index pada suatu nama variable. Contohnya, kita dapat menyimpan 5 nilai dengan tipe **`int`** tanpa harus mendeklarasikan 5 identifier variabel yang berbeda. Perhatikan contoh dibawah ini :



Bagian kosong diatas merepresentasikan *elemen* array, dalam kasus ini adalah nilai integer. Angka **`0 - 4`** merupakan index dan selalu dimulai dari **`0`**. Seperti penggunaan variable pada umumnya, array harus dideklarasikan terlebih dahulu, dengan format sbb :

```
type name [elements];
```

Maka contoh array diatas dideklarasikan sbb :

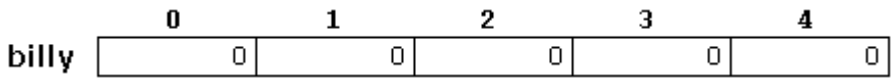
```
int billy [5];
```

Inisialisasi array

Ketika mendeklarasikan array lokal (didalam fungsi), jika tidak diberikan nilai maka isi dari array tidak akan ditentukan (*undetermined*) sampai nilai diberikan. Jika mendeklarasikan array global array (diluar semua fungsi) maka isi dari array akan di-iniisialisasikan sebagai 0 :

```
int billy [5];
```

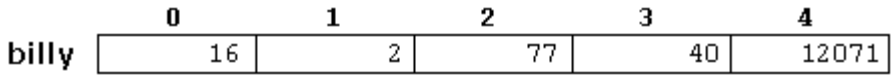
maka setiap elemen array *billy* akan di-inisialisasikan sebagai **`0`** :



Atau dideklarasikan dengan memberikan nilai array yang dituliskan dalam kurung kurawal :

```
int billy [5] = { 16, 2, 77, 40, 12071 };
```

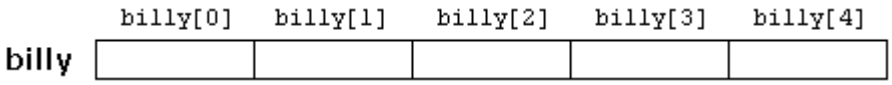
Maka elemen array *billy* akan berisi :



Access to the values of an Array.
Nilai array dapat diakses secara individual, dengan format :

name[*index*]

Maka dari contoh sebelumnya nama yang digunakan untuk mengakses masing-masing elemen:



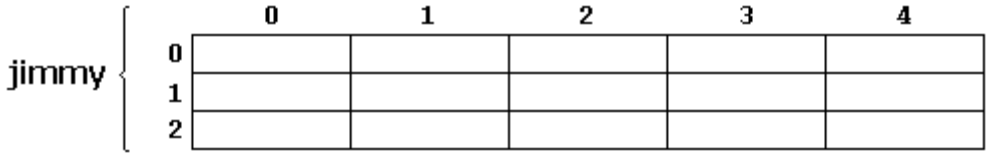
Misalkan akan disimpan nilai 75 pada elemen ketiga, maka intruksinya :

```
billy[2] = 75;
```

Dan jika nilai elemen ketiga tadi akan diberikan ke variable **a,** maka dapat dituliskan:

```
a = billy[2];
```

Array Multidimensi
Array Multidimensi dapat dikatakan sebagai array dari array. Contoh dibawah ini adalah array berdimensi 2 :



Maka pendeklarasiannya :

```
int jimmy [3][5];
```

Contoh:

```
// multidimensional array
#include <iostream.h>
#define WIDTH 5
#define HEIGHT 3
int jimmy [HEIGHT][WIDTH];
int n,m;
int main ()
{
  for (n=0;n<HEIGHT;n++)
```

```
    for (m=0;m<WIDTH;m++)
    {
      jimmy[n][m]=(n+1)*(m+1);
    }
  return 0;
}

// pseudo-multidimensional array
#include <iostream.h>
#define WIDTH 5
#define HEIGHT 3

int jimmy [HEIGHT * WIDTH];
int n,m;
int main ()
{
  for (n=0;n<HEIGHT;n++)
    for (m=0;m<WIDTH;m++)
    {
      jimmy[n * WIDTH +
m]=(n+1)*(m+1);
    }
  return 0;
}
```

Program diatas tidak akan menghasilkan tampilan, tetapi akan menyimpan nilai dalam memory seperti dibawah ini :

jimmy

	0	1	2	3	4
0	1	2	3	4	5
1	2	4	6	8	10
2	3	6	9	12	15

Penggunaan konstanta *defined* (**`#define`**) untuk mempermudah jika akan melakukan perubahan.

Array sebagai parameter

Adakalanya array diberikan kedalam fungsi sebagai parameter. Dalam C++ tidak memungkinkan untuk *pass by value* satu blok memory sebagai parameter kedalam suatu fungsi. Untuk menggunakan array sebagai parameter maka yang harus dilakukan saat pendeklarasian fungsi adalah spesifikasi tipe array pada argumen, Contoh :

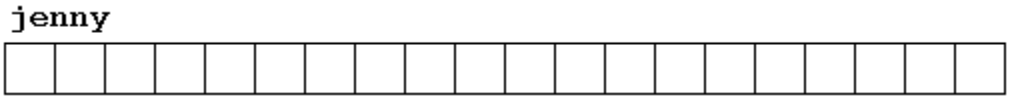
```
void procedure (int arg[])
```

String & Character

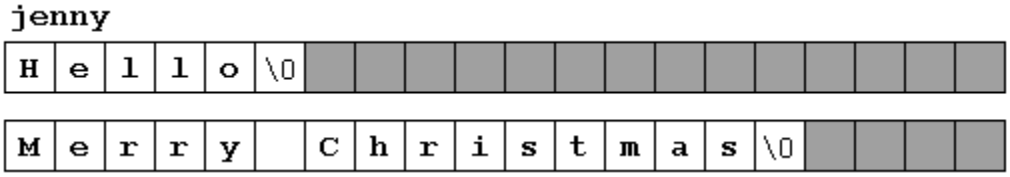
Pada C++ tidak ada tipe variable elemen yang spesific untuk menyimpan string. Untuk keperluan ini dapat digunakan array dengan tipe **char**, dimana berisi elemen dengan tipe **char**. Perlu di ingat bahwa tipe **char** digunakan untuk menyimpan 1 karakter, karena itu array dari char digunakan untuk menyimpan string. Contoh :

```
char jenny [20];
```

Dapat menyimpan sampai dengan 20 karakter :



Penyimpanan karakter-nta dapat direpresentasikan seperti dibawah ini :



Perhatikan, karakter NULL (**`'\0'`**) selalu disertakan diakhir string untuk indikasi akhir dari string.

Inisialisasi string

Sama halnya seperti array-array sebelumnya, inisialisasi pada string sbb :

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

Contoh diatas, merupakan inisialisasi 6 buah elemen bertipe **char,** yaitu **Hello** dan karakter null **'\0'**. Untuk menentukan nilai konstan, pada string digunakan tanda kutip ganda ("), sedangkan untuk karakter kutip tunggal ('). String yang diapit oleh kutip ganda sudah mengandung karakter Null pada akhir string, contoh :

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

contoh diatas merupakan deklarasi array **mystring** yang berisi 6 elemen.

Pemberian nilai pada string

Sama halnya seperti pemberian nilai pada array-array sebelumnya, untuk array dengan tipe char dapat dituliskan :

```
mystring[0] = 'H';
mystring[1] = 'e';
mystring[2] = 'l';
mystring[3] = 'l';
mystring[4] = 'o';
mystring[5] = '\0';
```

Cara diatas sangat tidak praktis. Umumnya untuk pemberian nilai pada array bertipe char digunakan fungsi **strcpy**. **strcpy** (**str**ing **c**o**py**) mendefinisikan **cstring** (string.h) library dan dapat dipanggil dengan cara :

```
strcpy (string1, string2);
```

instruksi diatas menyebabkan isi dari *string2* di-*copy* ke *string1*. *string2* dapat berupa array, pointer, atau konstanta string.

Metode lain yang dapat digunakan untuk inisialisasi nilai yaitu input stream (**cin**). Dalam kasus ini, nilai string ditentukan oleh user saat eksekusi program. Ketika menggunakan **cin,** biasanya digunakan metode **getline**, Pemanggilannya sbb :

```
cin.getline ( char buffer[], int length, char
delimiter = ' \n');
```

dimana, ***buffer*** adalah alamat untuk menyimpan input, ***length*** adalah maksimum panjang buffer,dan ***delimiter*** adalah karakter yang digunakan untuk menentukan input akhir, dengan default – atau dengan (**'\n'**).

Perhatikan kedua pemanggilan **cin.getline**, menggunakan identifier yang sama (**mybuffer**). Sama halnya seperti penggunaan operator extraction, sehingga dapat dituliskan :

```
cin >> mybuffer;
```

Instruksi diatas dapat berjalan, hanya saja mempunyai keterbatasan bila dibandingkan dengan **`cin.getline`**, diantaranya :

- Dapat menerima 1 kata saja (bukan kalimat lengkap).
- Tidak diperkenankan untuk memberikan ukuran buffer. Akan menyebabkan program tidak stabil jika user meng-input lebih besar dari kapasitas array yang ada.

Konversi string ke tipe lainnya

String dapat berisi data dengan tipe lain seperti angka. Contoh **`"1977"`**. **`cstdlib`** (**`stdlib.h`**) library menyediakan 3 fungsi yang dapat menangani hal tersebut :

- **atoi:** converts string to **`int`** type.
- **atol:** converts string to **`long`** type.
- **atof:** converts string to **`float`** type.

Fungsi-fungsi ini menerima 1 parameter dan mengembalikan nilainya kedalam tipe yang diminta (`int`, `long` or `float`). Fungsi ini dikombinasikan dengan metode **`getline`** pada **`cin`**.

Fungsi untuk manipulasi string

cstring library (`string.h`) mendefinisikan banyak fungsi untuk operasi manipulasi, diantaranya:

strcat: **`char* strcat (char*`** *`dest`*`, `**`const char*`** *`src`*`);`
Appends *src* string at the end of *dest* string. Returns *dest*.

strcmp: **`int strcmp (const char*`** *`string1`*`, `**`const char*`** *`string2`*`);`
Compares strings *string1* and *string2*. Returns `0` is both strings are equal.

strcpy: **`char* strcpy (char*`** *`dest`*`, `**`const char*`** *`src`*`);`
Copies the content of *src* to *dest*. Returns *dest*.

strlen: **`size_t strlen (const char*`** *`string`*`);`
Returns the length of *string*.

Cttn : **`char*`** sama dengan **`char[]`**

P.4.2 Contoh Kasus

- *arrays example*

```
// arrays example
#include <iostream.h>
int billy [] = {16, 2, 77, 40, 12071};
int n, result=0;

int main ()
{
  for ( n=0 ; n<5 ; n++ )
  {
    result += billy[n];
  }
  cout << result;
  return 0;
}
```

Output :

12206

- *cin with strings* :

```
// cin with strings
#include <iostream.h>

int main ()
{
  char mybuffer [100];
  cout << "What's your name? ";
  cin.getline (mybuffer,100);
  cout << "Hello " << mybuffer << ".\n";
  cout << "Which is your favourite team? ";
  cin.getline (mybuffer,100);
  cout << "I like " << mybuffer << " too.\n";
  return 0;
}
```

Output :

```
What's your name? Juan
Hello Juan.
Which is your favourite team? Inter Milan
I like Inter Milan too.
```

- *setting value to string* :

```
// setting value to string
#include <iostream.h>
#include <string.h>

int main ()
{
  char szMyName [20];
  strcpy (szMyName,"J. Soulie");
  cout << szMyName;
  return 0;
}
```

Output :

J. Soulie

Perhatikan, header **<string.h>** harus disertakan agar bisa menggunakan fungsi **strcpy**.
Bisa juga menggunakan fungsi sederhana seperti **setstring**, dengan operasi yang sama seperti **strcpy.**

- *setting value to string* :

```
// setting value to string
#include <iostream.h>

void setstring (char szOut [], char szIn [])
{
  int n=0;
  do {
    szOut[n] = szIn[n];
  } while (szIn[n++] != '\0');
}

int main ()
{
  char szMyName [20];
  setstring (szMyName,"J. Soulie");
  cout << szMyName;
  return 0;
}
```

Output :

J. Soulie

- *arrays as parameters* :

```
// arrays as parameters


#include <iostream.h>
void printarray (int arg[], int length)
{
  for (int n=0; n<length; n++)
    cout << arg[n] << " ";
  cout << "\n";
}

int main ()
{
  int firstarray[] = {5, 10, 15};
  int secondarray[] = {2, 4, 6, 8, 10};
  printarray (firstarray,3);
  printarray (secondarray,5);
  return 0;

}
```

Output :

5 10 15

2 4 6 8 10

Dari contoh diatas, instruksi (**`int arg[]`**) menjelaskan bahwa semua array bertipe **`int`**, berapapun panjangnya. oleh sebab itu dideklarasikan parameter kedua dengan sifat yang sama seperti parameter pertama.

P.4.3 Latihan

`1. Carilah output program di bawah ini :`

```
// cin and ato* functions
#include <iostream.h>
#include <stdlib.h>

int main ()
{
  char mybuffer [100];
  float price;
  int quantity;
  cout << "Enter price: ";
  cin.getline (mybuffer,100);
  price = atof (mybuffer);
```

```
  cout << "Enter quantity: ";
  cin.getline (mybuffer,100);
  quantity = atoi (mybuffer);
  cout << "Total price: " << price*quantity;
  return 0;
}
```

P. 4.4 Daftar Pustaka

1. Ayuliana, modul pengenalan bahasa C++, Gunadarma Jakarta, February 2004

2. Hari, Konsep Dasar Objek Oriented Programming, FTI budiluhur Jakarta, 2003

3. r.hubbard, John , schaum's outline of theory and problems of programming with C++ second edition, mcgraw-hill, New York 2000

4. http://www.cplusplus.com/

5. http://cs.binghamton.edu/~steflik/

6. http://en.wikipedia.org/wiki/c++