JURNAL PRAKTIKUM
(LAB. ACTIVITY)

STRUKTUR DATA
SI025

Materi 2:
PENGANTAR STRUKTUR DATA
(Array/Larik)

Dosen:

Agung Nugroho, M.Kom
Prof. Dr. Ema Utami, S.Si, M.Kom
Ikmah, M.Kom
Ninik Tri H, M.Kom
Lilis Dwi Farida, M.Kom
Windha Mega PD, M.Kom

S1 – SISTEM INFORMASI
UNIVERSITAS AMIKOM YOGYAKARTA
2018

Struktur Data
Pendahuluan

A. Tujuan

Setelah praktikum ini, praktikan diharapkan dapat:

1. Memahami tipe data dasar
2. Memahami tipe data bentukan
3. Menerapkan larik menggunakan bahasa C++

B. Peralatan

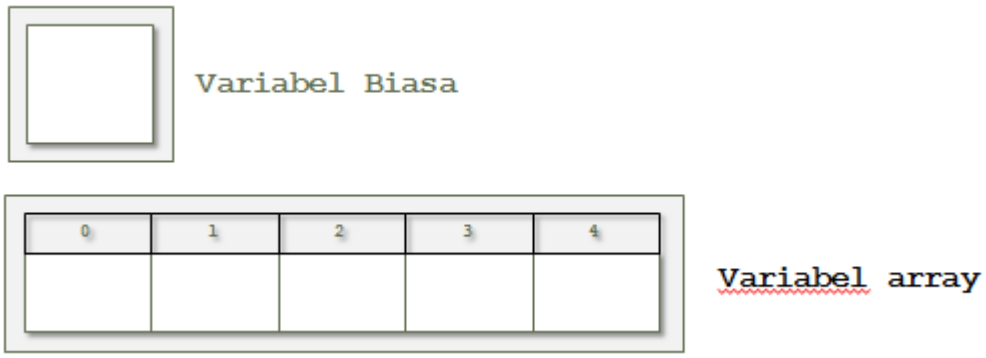
1. PC Desktop
2. Windows 7
3. MinGW

C. Teori

PENGANTAR C++

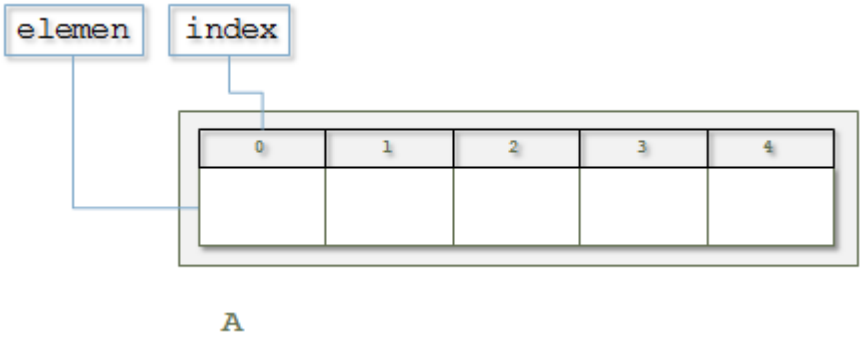
Arrays adalah serial dari elemen-elemen (variable-variabel) yang memiliki type yang sama dan ditempatkan secara berurutan pada memory sehingga dapat dinyatakan dengan menambah index pada nama variable tersebut.

Hal ini berarti bahwa, sebagai contoh, kita dapat menempatkan 5 nilai yang bertype **`int`** tampa harus mendeklarasikan 5 variabel yang berbeda untuk masing-masing nilai. Sebagai gantinya kita dapat menggunakan *array* untuk menyimpan kelima nilai tersebut.



Contoh

Array A mempunyai ukuran 5 index penyimpanan



dimana setiap kotak kosong mewakili *element* dari array, dimana dalam hal ini adalah nilai integer dari type **`int`**. Index array tersebut dimulai dari **`0`** sampai **`4.`**

Sama seperti variable lainnya, suatu array harus dideklarasikan terlebih dahulu sebelum pemakaiannya, deklarasi array pada C++:

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

dima *`type`* adalah type data (**int**, **float**...), *`name`* adalah nama variabel dan *`elements`* adalah jumlah element yang diinginkan, dalam hal ini diapit oleh brackets (kurung siku) **`[]`**

Selanjutnya, contoh penulisan deklarasi:

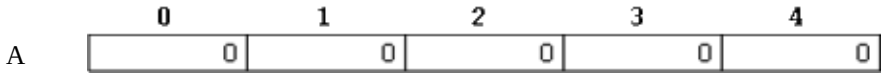
```
int A [5];
```

CATATAN: *Elements* harus berupa nilai constant (konstanta), karena array merupakan blok dari memory static yang harus diberikan, sehingga compiler dapat mengetahui jumlah memory yang diperlukan array sebelum instruksi.

Jika kita mendeklarasi array global (diluar dari function), nilai awalnya adalah nol. Maka jika kita mendeklarasikan array global berikut:

```
int A [5];
```

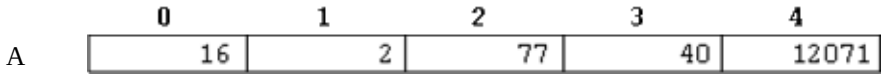
setiap elemen dari **A** adalah 0:



Kita juga dapat mendeklarasikan suatu Array, berikut dengan nilainya dengan menuliskan nilainya dalam tanda kurawal { }. Sebagai contoh:

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

Deklarasi diatas akan menghasilkan array sebagai berikut :



Atau pada C++, boleh juga ditulis sebagai berikut :

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

Access to the values of an Array.

Dalam hal ini, setiap element array yang ada dapat diakses satu persatu nilainya dengan format penulisan berikut:

name[*index*]

Sebagai contoh, untuk menyimpan nilai 75 pada elemen kelima pada ***A***, penulisan yang sesuai adalah:

```
A[4] = 75;
```

dan, untuk menyimpan nilai elemen ketiga dari variable ***A*** ke variable **b**, kita dapat menulis:

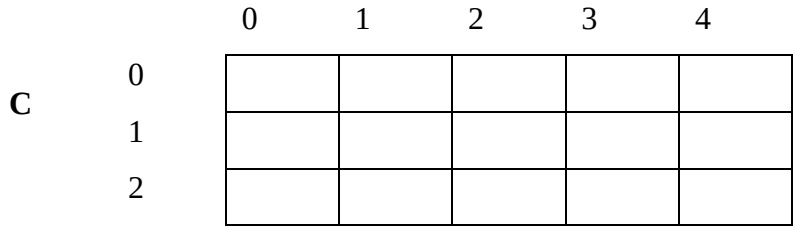
```
b = A[2];
```

```
// arrays example
#include <iostream.h>
int A [] = {16, 2, 77, 40, 12071};
int n, result=0;
int main ()
 {
  for ( n=0 ; n<5 ; n++ )
  {
   result += A[n];
  }
  cout << result;
  return 0;
 }
```

Output : 12206

Multidimensional Arrays

Array multidimensional dapat digambarkan sebagai array dari array.



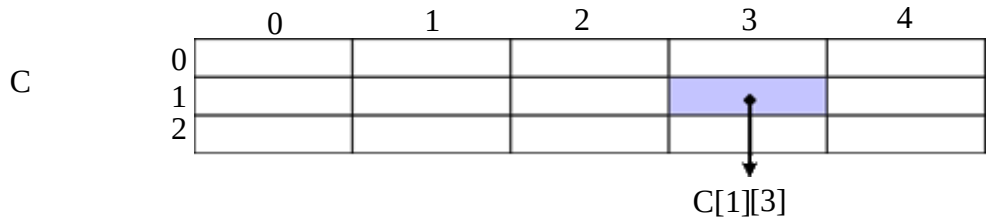
Misalnya **C** adalah suatu bidimensional array 3 baris 5 kolom dengan type **int**.

Cara deklarasinya adalah sebagai berikut:

```
int C [3][5];
```

dan, cara untuk menyebutkan elemen baris kedua, kolom keempat adalah dengan ekspresi berikut :

```
C[1][3]
```



(sesuatu yang harus anda ingat adalah index dari array selalu dimulai dengan **0).*

Array Multidimensional tidak hanya terbatas pada dua dimensi saja. Mereka dapat memiliki dimensi sesuai dengan kebutuhan, walaupun sebenarnya jarang melebihi 3 dimensi.

Contoh:

```
char century [100][365][24][60][60];
```

Menentukan type **char** untuk setiap detik dalam satu abad, yang terdiri lebih dari 3 milyar **chars**! Hal ini akan menggunakan memory sebesar 3000 *megabytes* dari RAM jika kita mendeklarasikannya.

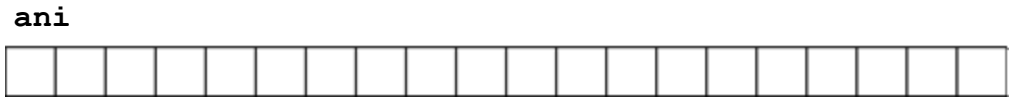
Strings of Characters.

Dalam semua program yang telah kita buat, kita hanya menggunakan variable numeric, dan menggunakan ekspresi numeric. Selain numeric sebenarnya masih ada karakter string Pada C untuk menyimpan karakter string kita dapat menggunakan type **char**, yang mana merupakan urutan dari elemen **char**

Sebagai contoh, array berikut (atau karakter string):

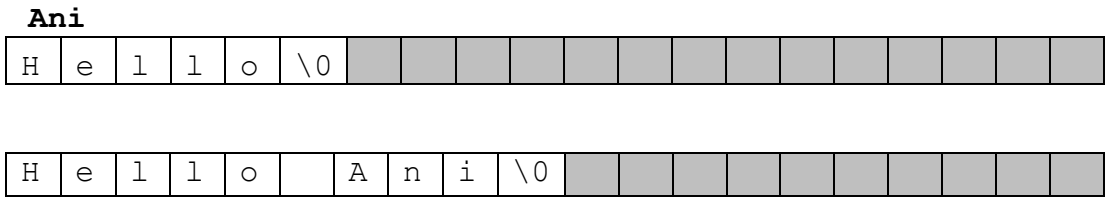
```
char ani [20];
```

dapat menyimpan 20 karakter. Anda dapat membayangkannya sebagai:



Ukuran maksimum dari 20 karakter tidak selamanya akan digunakan secara keseluruhan. Dapat saja hanya berisi tulisan "Hello" ataupun "Happy holiday", sehingga perlu diakhiri dengan suatu karakter null, yang ditulis sebagai **0** atau **'\0'**.

Sehingga dapat digambarkan sebagai berikut :



Initialization of strings

Karena sebenarnya karakter string adalah array, sehingga cara mengisinya sama dengan array biasanya, contoh :

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

Dalam hal ini, kita akan mendeklarasikan suatu karakter string (array) yang terdiri dari 6 elemen type **`char`** dengan nilai **`Hello`** diikuti dengan suatu karakter null **`'\0'`**.
Berikut ini adalah cara lain untuk deklarasi karakter string dengan nama **`mystring`** dengan hasil yang sama :

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

Assigning values to strings

Karena merupakan suatu array, untuk memberikan nilai kepada karakter string kita dapat menggunakan metode berikut:

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

Tetapi tentu saja cara demikian adalah sangat tidak praktis. Umumnya untuk memberikan nilai pada suatu array, khususnya untuk karakter string kita dapat menggunakan fungsi seperti **`strcpy`**. **strcpy** (**str**ing **c**o**py**) dimana didefinisikan dalam library **`cstring`** (`string.h`) yang dapat digunakan dengan penulisan berikut:

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

Hal ini akan menduplikasi isi dari *string2* kedalam *string1*. *string2* dapat berupa suatu array, suatu pointer, atau suatu constant string, contoh :

```
strcpy (mystring, "Hello");
```

Contoh :

// setting value to string

```
#include <iostream.h>
#include <string.h>
int main ()
{
 char szMyName [20];
 strcpy (szMyName,"Agung N");
 cout << szMyName;
 return 0;
}
```

Output : `Agung N`

Perhatikan bahwa kita perlu melakukan include **<string.h>** pada bagian header agar dapat digunakan function **strcpy**.

Walaupun kita juga dapat menulis suatu fungsi sederhana seperti **setstring** berikut dengan operasi yang menyerupai **strcpy**:

// setting value to string

```
#include <iostream>
Using namespace std;

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,"Agung.N");
 cout << szMyName;
 return 0;
}
```

Output : `Agung.N`

Metode lain yang sering dipergunakan untuk memasukan nilai ke suatu array adalah dengan langsung memasukannya melalui input stream (**cin**). Dalam hal ini nilai string diberikan oleh user pada saat eksekusi program.

Ketika **cin** digunakan dengan string atau karakter biasanya menggunakan metode **getline**, yang dapat dipanggil dengan prototype berikut:

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

dimana ***buffer*** adalah alamat untuk menyimpan input (dalam hal ini adalah array), ***length*** adalah panjang maksimum dari buffer (ukuran dari array) dan ***delimiter*** adalah karakter yang digunakan untuk mengakhiri user input, dimana defaultnya adalah newline character (**'\n'**).

Berikut ini adalah contoh sederhana pemakaian **cin.getline** pada string:

```
// cin with strings
#include <iostream>
Using namespace std;
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;
}
```

```
What's your name? Agung
Hello Juan.
Which is your favourite team?
Manchester United
I like Manchester United too.
```

Sebenarnya kita dapat saja menggunakan perintah berikut untuk membaca string ke suatu variable array:

```
cin >> mybuffer;
```

Tetapi jika dibandingkan dengan **cin.getline**, perintah diatas memiliki keterbatasan:

- Hanya dapat menerima satu kata tunggal (bukan kalimat lengkap) karena metode ini

menggunakan karakter kosong sebagai delimeternya, seperti spasi, tabulator,

- Tidak dimungkinkan untuk membatasi ukuran buffer. Hal ini akan membuat program anda menjadi tidak stabil jikalau input oleh user melebihi ukuran array.

Untuk alas an ini, kami menyarankan anda untuk menggunakan **`cin.getline`** sebagai penganti dari **`cin >>`**.

Converting strings to other types

Kadang-kadang kita perlu melakukan konversi string ke type numerik. Misalnya suatu string seperti **`"1977"`**, dan kita ingin mengkonversinya ke suatu type data integer, untuk keperluan tersebut kita membutuhkan library **`cstdlib`** (**`stdlib.h`**) yang mengandung tiga fungsi untuk keperluan ini :

- **atoi:** melakukan konversi string ke type **`int`**.
- **atol:** melakukan konversi string ke type **`long`**.
- **atof:** melakukan konversi string ke type **`float`**.

Semua fungsi diatas membutuhkan satu parameter dan mengembalikan suatu nilai berdasarkan tyoe yang dikehendaki (`int`, `long` or `float`). contoh:

```
// 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;
}
```

```
Enter price: 2.75
Enter quantity: 21
Total price: 57.75
```

Functions to manipulate strings

Library **cstring** (`string.h`) mendefinisikan banyak fungsi untuk melakukan operasi manipulasi terhadap string pada C (seperti yang diterangakan pada strcpy). Berikut ini adalah ringkasan fungsi yang sering digunakan:

strcat: **`char* strcat (char*`** *`dest`*`, const char*` *`src`*`);`

Menambahkan *src* string pada akhir dari *dest* string. Mengembalikan *dest*.

strchr: **`char* strchr (const char*`** *`string`*`, int` *`c`* **`);`**

Mengembalikan pointer pada karakter pertama ditemukan dan null jika tidak ditemukan.

strcspn: **`size_t strcspn ( const char *`** *`string1`*`, const char *`** *`string2`* **`);`** Mencari dalam *string1* character demi character, mengembalikan posisi pertama yang mengandung salah satu character pada string 2, fungsi akan mengembalikan nilai panjang dari *string1* kalau tidak ada character *string2* didalam *string1*, karena masing-masing diakhiri dengan null.

Contoh

```
      /* strcspn example */
#include <stdio.h>
#include <string.h>
int main ()
      {
```

```
    char str1[] = "fcba73";
    char str2[] = "1234567890";
    int i;
    i = strcspn (str1,str2);
    printf ("The first number in str1 is str1[%d]\n",i);
    return 0;
}
```

Output:

```
The first number in str1 is str1[4]
```

strcmp: **`int strcmp (const char*`** *`string1`*`,` **`const char*`** *`string2`*`);`

Membandingkan string *string1* dan *string2*. Mengembalikan `0` jika keduanya serupa.

strcpy: **`char* strcpy (char*`** *`dest`*`,` **`const char*`** *`src`*`);`

Menduplikasi isi dari *src* ke *dest*. Mengembalikan *dest*.

strlen: **`size_t strlen (const char*`** *`string`*`);`

Mengembalikan panjang dari *string*.

CATATAN: **`char*`** adalah sama dengan **`char[]`**

D. Praktikum

Latihan 1

1. Diberikan program seperti di bawah ini:

```cpp
#include <iostream>

using namespace std;

int main()
{
    string nama;
    string MK;
    int nilai;

    //input data
    cout<<"masukkan nama mahasiswa: ";
    cin>>nama;
    cout<<"masukkan matakuliah yang diambil: ";
    cin>>MK;
    cout<<"masukkan nilai mahasiswa: ";
    cin>>nilai;

    //menampilkan data
    cout<<"\ntampilkan data"<<endl;
    cout<<"nama: "<<nama<<" Nilai Matakuliah "<<MK<<" adalah = "<<nilai<<endl;

    return 0;
}
```

Output Program:

```
masukkan nama mahasiswa: Windha
masukkan matakuliah yang diambil: StrukturData
masukkan nilai mahasiswa: 90

tampilkan data
nama: Windha Nilai Matakuliah StrukturData adalah = 90
```

Code program di atas hanya dapat digunakan untuk menyimpan data 1 mahasiswa. Apabila akan digunakan untuk menyimpan data beberapa mahasiswa, maka perlu diubah sedikit code programnya, yaitu dengan membuat array pada tiap variabel.

2. Tambahkan code program agar dapat digunakan untuk menyimpan data beberapa mahasiswa:
   a. Mengubah variabel nama, MK dan nilai yang semula variabel biasa menjadi variabel array

```cpp
string nama[100];
string MK[100];
int nilai[100];
```

   b. Tambahkan variabel untuk menyimpan jumlah mahasiswa yang akan diinputkan: int jml;

c. Berikan baris program untuk menginputkan jumlah mahasiswa, dan disimpan pada variabel jml

```
cout<<"Masukkan jumlah mahasiswa = ";
cin>>jml;
```

d. Buat perulangan untuk proses input data mahasiswa, dan disimpan di tiap elemen array

```
//input data
for(int i=1; i<=jml;i++){
cout<<"masukkan nama mahasiswa: ";
cin>>nama[i];
cout<<"masukkan matakuliah yang diambil: ";
cin>>MK[i];
cout<<"masukkan nilai mahasiswa: ";
cin>>nilai[i];
}
```

e. Buat perulangan untuk menampilkan beberapa data yang telah diinputkan

```
    for(int i=1; i<=jml;i++){
    cout<<"\ntampilkan data"<<endl;
    cout<<"nama: "<<nama[i]<<" Nilai Matakuliah "<<MK[i]<<" adalah = "<<nilai[i]<<endl;
    }
```

Output:

```
Masukkan jumlah mahasiswa = 2
masukkan nama mahasiswa: Windha
masukkan matakuliah yang diambil: SD
masukkan nilai mahasiswa: 100
masukkan nama mahasiswa: Mega
masukkan matakuliah yang diambil: SD
masukkan nilai mahasiswa: 90

tampilkan data
nama: Windha Nilai Matakuliah SD adalah = 100
nama: Mega Nilai Matakuliah SD adalah = 90
```

E. Tugas

Buat Program untuk kasus di bawah ini:

1. Jumlahkan isi array yang anda inputkan
2. Buat data buku