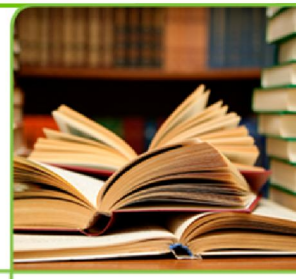
Kurikulum Qt

{ Basic OOP }

# Chapter 3

## Array dan String

NOKIA



# Agenda

- Pengantar Array
- Array 1 Dimensi
- Inisialisasi Array 1 Dimensi
- Pengalamatan dan Penyalinan Array 1 Dimensi
- Array multi dimensi
- Deklarasi Array 2 dimensi
- Pengakesan Array 2 dimensi
- String C-style
- Fungsi-fungsi String C-Style
- Class String C++
- Fungsi-fungsi class String C++

# Pengantar Array

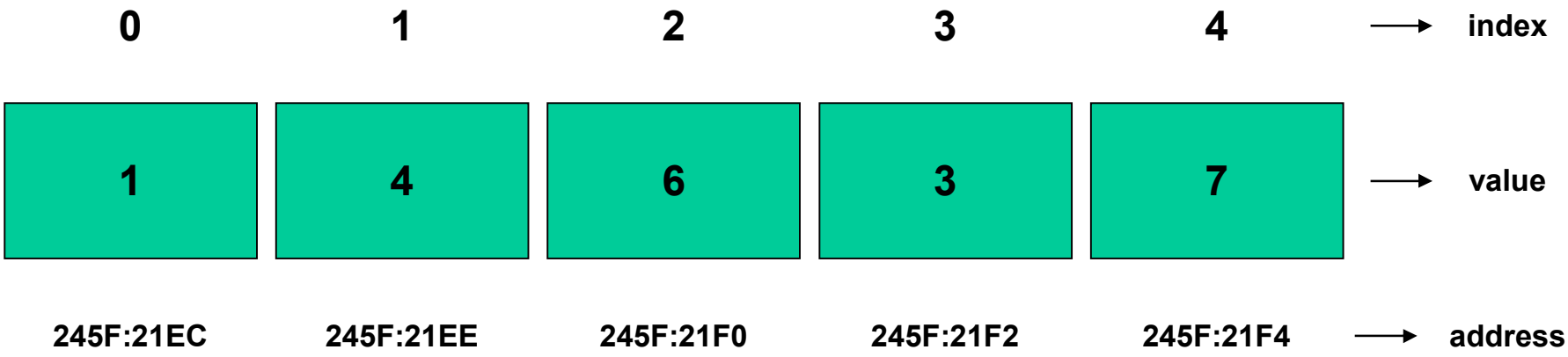
- Selama ini kita menggunakan satu variabel untuk menyimpan 1 buah nilai dengan tipe data tertentu.
  - int a1, a2, a3, a4, a5;
  - Deklarasi variabel diatas digunakan untuk menyimpan 5 data integer dimana masing-masing variabel diberi nama a1, a2, a3, a4, dan a5.
- Jika kita memiliki 10 data, 100 data integer bahkan mungkin data yang ingin kita proses tidak kita ketahui atau bersifat dinamis? Kita tidak mungkin menggunakan variabel seperti diatas.
- Bagaimana jika kita ingin menghitung total dari variabel biasa?

  **total = x1 + x2 + x3 + x4 + x5 + ... + xn;**

# Pengantar Array

- Di dalam C++, terdapat suatu fasilitas untuk menyimpan data-data yang bertipe **data sama** dengan suatu **nama tertentu** = ARRAY
- Array adalah suatu tipe data terstuktur yang berupa sejumlah data sejenis (bertipe data **sama**) yang jumlahnya tetap dan diberi suatu nama tertentu.
- Elemen-elemen array tersusun secara **sekuensial** di dalam memori sehingga memiliki alamat yang **berdekatan/bersebelahan**.
- Array dapat berupa array 1 dimensi, 2 dimensi, bahkan n-dimensi (multidimensi).
- Elemen-elemen array bertipe data sama tapi bisa bernilai **sama** atau **berbeda-beda**.

# Bentuk Array dalam Memory (int)



# Array 1 Dimensi

- Elemen-elemen array dapat diakses oleh program menggunakan suatu **indeks** tertentu
- Pengaksesan elemen array dapat dilakukan **berurutan** atau **random** berdasarkan indeks tertentu secara langsung.
- Pengisian dan pengambilan nilai pada indeks tertentu dapat dilakukan dengan mengeset nilai atau menampilkan nilai pada **indeks** yang dimaksud.
- Dalam C++, **tidak terdapat** error handling terhadap batasan nilai indeks, apakah indeks tersebut berada di dalam indeks array yang sudah didefinisikan atau belum.
  - Hal ini merupakan tanggung jawab programmer, sehingga jika programmer mengakses indeks yang salah, maka nilai yang dihasilkan akan berbeda atau rusak karena mengakses alamat memori yang tidak sesuai.

# Deklarasi Array 1 Dimensi

```
tipe_data nama_var_array[ukuran];
```

tipe_data : menyatakan jenis tipe data elemen larik (int, char, float, dll)

nama_var_array : menyatakan nama variabel yang dipakai.

ukuran : menunjukkan jumlah maksimal elemen larik.

-Tipe data sejenis
-Ada indeks yang teratur dan berurutan
-Bersifat statis, harus diketahui ukurannya terlebih dahulu

# Contoh Array 1 Dimensi

Contoh:

```
char huruf[9];
int umur[10];
int kondisi[2] = {0,1}
int arr_dinamis[] = {1,2,3}
```

# Penjelasan Lebih Lanjut

- Tanda [] disebut juga “elemen yang ke- „.
  - Misalnya “kondisi[0]“ berarti elemen yang ke nol.
- Array yang sudah dipesan, misalnya 10 tempat tidak harus diisi semuanya, bisa saja hanya diisi 5 elemen saja, baik secara berurutan maupun tidak.
- Namun pada kondisi yang tidak sepenuhnya terisi tersebut, tempat pemesanan di memori tetap sebanyak 10 tempat, jadi tempat yang tidak terisi tetap akan terpesan dan dibiarkan kosong.

# Demo Manipulasi Array 1D

# Inisialisasi Array 1D

- Cara menginisialisasi data pada array adalah dengan menuliskannya secara langsung pada source code program
- **// An array of 5 integers, all elements initialized to 0**

  **int IntegerArray[5] = {0};**
- **// An array of 5 integers initialized to zero, except the last one**

  **int IntegerArray[5] = { 0, 0, 0, 0, 6 };**
- Untuk semua array pada C++, inisialisasi satu buah elemen saja pada array akan membuat semua elemen array lainnya berisi nilai 0.

# Inisialisasi Array 1D

- Kita tidak dapat melakukan inisialisasi pada array melebihi batas jumlah elemen array yang dipesan.
- Pada array satu dimensi, kita dapat membuat array 1 dimensi tanpa menyebutkan jumlah elemen array yang dipesan.
  - Namun semua elemen harus diinisialisai terlebih dahulu.
- Contoh:
- **int data[5] = {1,2,3,4,5,6};  //error**
- **int data2[] = {10,20};  //terpesan 2 tempat dimemory**
- Inisialisasi pada elemen array yang dideklarsikan **SANGATLAH PENTING** untuk menghindari nilai **ACAK**!

# Demo Inisialisasi Array 1 Dimensi

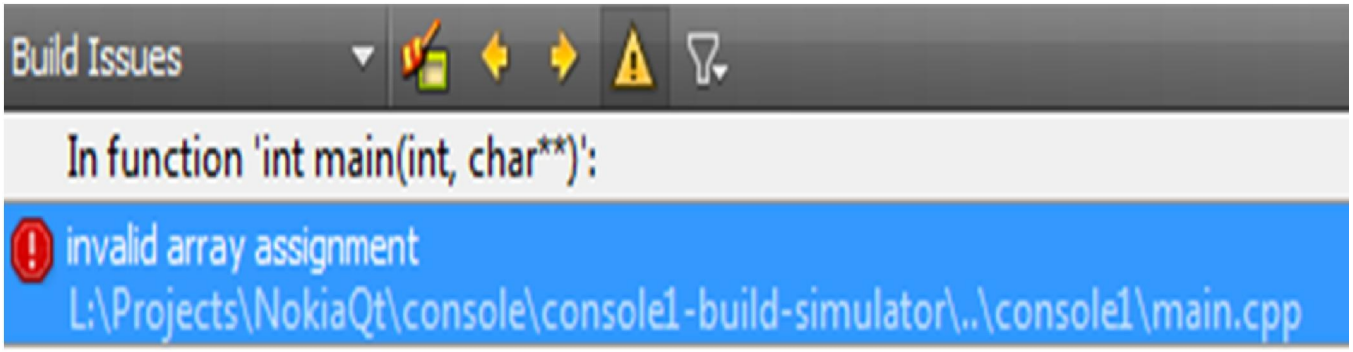
# Pengalamatan dan Penyalinan Array

- Array tidak bisa disalin begitu saja antara array satu yang ada nilainya ke array lain yang kosong.
  - Karena array bukan tipe data primif, array terdiri dari banyak elemen data.
- Compiler mencatat alamat array indeks pertamanya saja
  - Untuk mengakses elemen selanjutnya compiler menghitung jarak berdasarkan lebar tipe data yang digunakan

# Pengalamatan dan Penyalinan Array

```
int A[6]={1,2,3,4,5,6};
int B[6];
B = A;
```

- Error:



# Demo Penyalinan Array 1 Dimensi

# Penghapusan Array

- Elemen array tidak dapat dihapus saat runtime
- Untuk penghapusan gunakan trik:
  - Buat array baru yang tidak berisi elemen yg dihapus
  - Timpa elemen array yang dihapus dengan data elemen belakangnya
    - Jadi seolah-olah elemen-elemen datanya maju satu persatu kedepan
    - Index i = index i+1

# Array Dimensi 2

- Sering kali digambarkan/dianalogikan sebagai sebuah matriks.
- Jika array berdimensi satu hanya terdiri dari 1 baris dan banyak kolom, array berdimensi dua terdiri dari banyak baris dan banyak kolom yang bertipe sama
- Gambar array berdimensi (baris x kolom = 3 x 4)

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 5 | 20 | 1 | 11 |
| 1 | 4 | 7 | 67 | -9 |
| 2 | 9 | 0 | 45 | 3 |

# Deklarasi Array 2 Dimensi

- **Bentuk Umum:**

```
tipe_data nama_var_array[batas_baris][batas_kolom];
```

- **Contoh:**

```
int matriks[3][4];
int matriks2[3][4] = { {5,20,1,11}, {4,7,67,-9}, {9,0,45,3} };
```

$$x = \begin{pmatrix} 8 & 5 & 9 & 8 \\ 8 & 2 & 1 & 0 \end{pmatrix}$$

## Deklarasi:

int x[2][4];

**X[0][0]=8; x[0][1]=5; x[0][2]=9; x[0][3]=8;**

**X[1][0]=8; x[1][1]=2; x[1][2]=1; x[1][3]=0;**

atau

int x[2][4]= {{8, 5, 9, 8},{8, 2, 1, 0}};

# Deklarasi array 2 dimensi

- int matriks[3][5] = {{5,12,17,10,7},
  {15,6,25,2,19},
  {4,9,20,22,11}};

- Jika data yang diinputkan kurang dari deklarasi
  int matriks[3][5] = {{5,12,17,10,7},
  {15,6,25,2,19},
  {4,9 }};  //kurang 3 angka
- Maka tiga data yang kurang akan diisi dengan **0**

- Jika data yang diinputkan lebih dari deklarasi
  int matriks[3][5] = {{5,12,17,10,7},
  {15,6,25,2,19},
  {4,9,20,22,11,14,19 }};  //lebih 2 angka
- Matriks yang jumlah datanya lebih akan menyebabkan **ERROR**

# Pengaksesan Array 2 Dimensi

- Diakses indeksnya dengan menggunakan looping
  - Outer loop
  - Inner loop
  - Akses elemen pada indeks tersebut
    - Data[i][j]
- Pemrosesan ada 2:
  - Baris-demi-baris
  - Kolom-demi-kolom

# PROSES MATRIKS

## Matriks

Program Proses_Matrik_BarisdemiBaris

**Definisi**

```
#define baris 2
#define baris 3

int A[baris][kolom];
```

**ALGORITMA**

```
For   i  ← 0 to baris-1   do
        For   j  ← 0 to kolom-1   do
                PROSES MATRIK
        Endfor
Endfor
```

# PROSES MATRIKS

Matriks

Program Proses_Matrik_KolomdemiKolom

**KAMUS**

```
#define baris 2
#define kolom 3
int A[M][N];
```

**ALGORITMA**

```
For   i  ← 0 to kolom-1   do
        For   j  ← 0 to baris-1   do
                PROSES MATRIK
        Endfor
Endfor
```

# Demo Penyalinan Array 2 Dimensi

# String

- Nilai String adalah kumpulan dari nilai-nilai karakter yang berurutan dalam bentuk satu dimensi, nilai string ini haruslah ditulis didalam tanda petik dua (“) misalnya: “*ini string*”.
- Suatu nilai string disimpan di memori dengan diakhiri oleh nilai ‘\0’(*null*), misalnya nilai string “ANTO” disimpan dimemori dalam bentuk:
- ‘A’ ‘N’ ‘T’ ‘O’ ‘\0’
- Untuk mendeklarasikan sebuah string terdapat dua cara:
  - Menggunakan array of character (C-style string)
  - Menggunakan tipe data class string pada C++

# Array of Character

- Cara menggunakan array of character sama seperti mendeklarasikan variabel bertipe array namun bertipe data character
- Array of character memiliki sifat-sifat array lainnya yaitu bersifat statis dan letaknya berurutan di dalam memory komputer.

```
char nama[6];                 //tanpa inisialisasi
char nama2[6] = “anton”;      //langsung diinisialisasi
char nama2[6] = {‘a’,’n’,’t’,’o’,’n’};      //langsung diinisialisasi
```

# Demo Array of Character

# Fungsi String C-style

**strlen()**

- Berfungsi untuk menentukkan panjang suatu nilai string.
- Bentuk umum: **int strlen(<identifier string>);**

**length()**

- Berfungsi untuk menentukan panjang suatu nilai tipe data class string
- Bentuk umum method: **<nama_var_string>.length()**

**strcpy()**

- Bentuk umum: **void strcpy(<stringhasil>,<stringsumber>);**

**strncpy()**

- Bentuk umum: **void strncpy(<stringhasil>,<stringsumber>);**

# Fungsi String C-style

**strcat()**

- Berfungsi untuk menggabungkan dua string
- Bentuk umum: **strcat(<string hasil>, <string sumber>);**

**strchr()**

- Berfungsi untuk mencari suatu karakter dalam suatu string.
- Hasil dari fungsi ini adalah alamat letak dari karakter pertama di nilai string yang sama dengan karakter yang dicari.
- Bentuk umum: **strchr(stringsumber,karakter yang dicari)**

**strcmp()**

- Berfungsi untuk membandingkan string satu dengan string lain
  - Hasil < 0, Jika string1 < string2
  - Hasil = 0, Jika string1 = string2
  - Hasil > 0, Jika string1 > string2
- Bentuk umum: **strcmp(string1,string2);**

# Demo Fungsi String C-style

# Class string pada C++

- C++ library standar memiliki kelas string yang membuat bekerja dengan string lebih mudah dengan menyediakan satu set encapsulasi dari data, dan fungsi untuk memanipulasi data string
- std::string
- Menggunakan #include <string>
- Class string dapat menangani rincian alokasi memori dan membuat kopi string, atau menempatkan mereka di memory dengan lebih mudah

# Class String

- Mengurangi kesulitan dalam upaya penciptaan dan memanipulasi string
- Meningkatkan stabilitas aplikasi yang sedang diprogram dalam pengelolaan dan alokasi memori internal
- Mudah dalam menyalin, memotong, menemukan, dan penghapusan string
- Memberikan kesempatan pada programmer untuk lebih fokus pada pengembangan aplikasi daripada kesulitan dalam manipulasi string

# Class String

- Deklarasi:
  - string stringku1;
  - string stringku2(“ini string”);
  - string stringku3 = “anton”;
  - string stringku4(string2);
- string dapat diakses juga per karakter pembentuknya
  - Karena string pada dasarnya adalah array of character

# String concatenation

- Menggunakan opeator + dan method append
  - stringku = string1 + string2;
  - stringku.append(string2);

# Pengaksesan String

- Menggunakan loop

```
for(size_t i=0;i<stringku.length();i++){
        cout<<stringku[i]<<endl;
 }
```

NOKIA

# Demo Class String

# Fungsi-fungsi class string

- Menemukan substring pada string besar
  - Menggunakan method **find(<string yg dicari>,<offset index>);**
- Membalik kalimat:
  - Menggunakan header #include<algorithm>
  - Menggunakan method **reverse (<string>.begin (), <string>.end ());**
- Konversi case:
  - Huruf besar ke kecil: **transform(strInput.begin(),strInput.end(),strInput.begin(),(int(*)(int))toulower);**
  - Huruf kecil ke besar: **transform(strInput.begin(),strInput.end(),strInput.begin(),(int(*)(int))toupper);**

# Demo Fungsi class String

# Thank You

NOKIA