PEMROGRAMAN BERORIENTASI OBJEK

Konsep Array dalam PBO

Telkom University

Budhi Irawan, S.Si, M.T

PENDAHULUAN

- ***Array* (Larik)** merupakan hal fundamental yang sering dijumpai dalam banyak kasus didunia pemrograman, maka dari itu sebagai pemrogram (programmer) perlu dibekali dengan pemahaman konsep ***array*** dan mampu mengimplementasikannya.

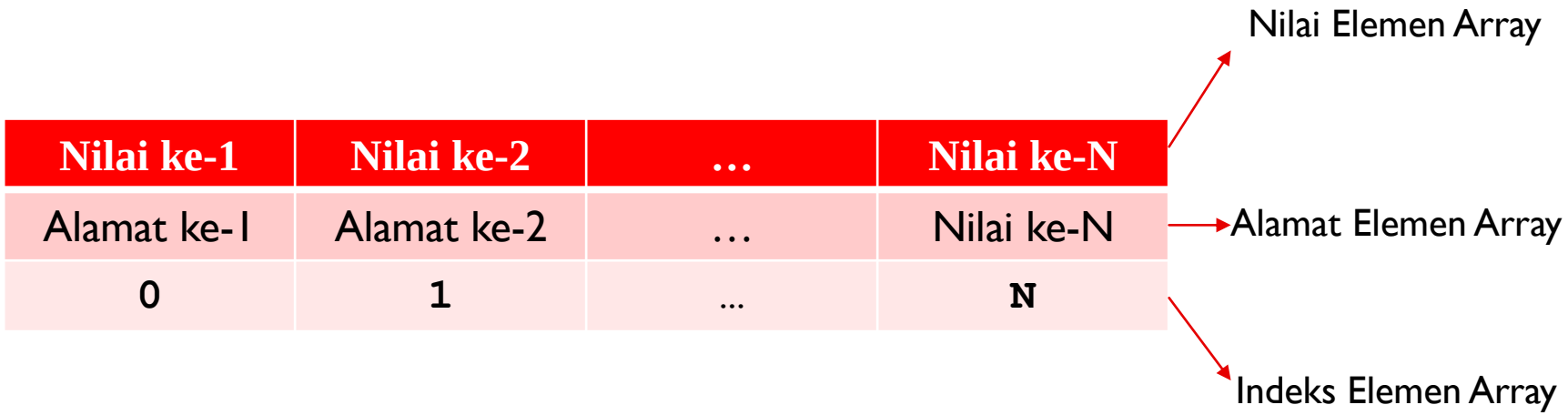
ARRAY

- ***Array*** adalah suatu ***variabel*** yang menyimpan sekumpulan data yang memiliki ***tipe data*** yang sama.
- Setiap data tersebut menempati ***lokasi*** atau ***alamat memory*** yang berbeda-beda dan selanjutnya disebut ***elemen array***.

ELEMEN ARRAY

- ***Elemen Array*** dapat diakses melalui ***indeks*** yang terdapat didalamnya.
- Namun penting sekali untuk diperhatikan dalam c++ ***indeks array*** selalui dimulai dari 0 bukan 1.

ILUSTRASI ARRAY



DEKLARASI ARRAY

- Untuk mendeklarasikan sebuah ***array*** dalam bahasa C++ maka digunakan ***tanda [] (bracket)***.
- Adapaun bentuk umum pendeklarasiannya adalah :

```
tipe_data nama_array[jumlah_elemen];
```

DEKLARASI ARRAY

- Sebagai contoh dideklarasikan sebuah ***array*** misalnya dengan nama **`LARIK`** yang memiliki **`25`** elemen dengan tipe data **`int`**, maka bentuk deklarasinya adalah sebagai berikut :

```
int LARIK[25];
```

RUANG MEMORY ARRAY

- Ruang memory yang dibutuhkan untuk deklarasi ***array*** tersebut adalah 100 byte yang berasal dari 25 x 4 byte (4 merupakan ukuran dari tipe data **`int`**).
- Cara yang digunakan untuk mengakses ***elemennya*** adalah dengan menuliskan ***indeksnya***.

RUANG MEMORY ARRAY

- Misalnya ingin mengambil nilai yang terdapat pada ***elemen ke-10*** dan menampung nilai tersebut kedalam sebuah variabel yang bertipe **`int`** juga (misal **`X`**), maka kita perlu menuliskan kode seperti berikut :

```
X = LARIK[9]
```

- Kenapa `9` bukan `10` ? Perlu diingat bahwa indeks array selalu dimulai dari **0** sehingga untuk mengakses elemen ke `10` maka indek yang dibutuhkan adalah `10 – 1` yaitu **9**.

ISI NILAI KE ELEMEN ARRAY

- Cara mengisikan nilai ke dalam ***array*** yang paling umum dilakukan adalah dengan menggunakan ***pengulangan (looping)***.
- Cara ini akan jauh lebih cepat dibandingan cara manual satu persatu. Sebagai contoh apabila ingin dilakukan pengisian ***25 elemen array***, maka bentuk penulisannya adalah sebagai berikut :

```
for (int C=0; C<25; C++) {
cout<<"A["<<C<<"] = "; cin>>A[C];
}
```

ISI NILAI KE ELEMEN ARRAY

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

int main() {
  // Mendeklarasikan array A dengan 5 buah
  // elemen bertipe int
  int A[5];

  // Memasukkan nilai ke dalam elemen array
  for (int C=0; C<5; C++) {
    cout<<"A["<<C<<"] = "; cin>>A[C];
  }

  return 0;
}
```

MENAMPILKAN NILAI PADA ARRAY

- Setelah memahami cara mengisikan nilai kedalam ***elemen array***, selanjutnya dibahas bagaimana cara mengakses atau menampilkan nilai-nilai tersebut.
- Konsepnya sama saja, yaitu digunakan ***pengulangan*** untuk menampilkannya.

MENAMPILKAN NILAI PADA ARRAY

```
#include <iostream>
using namespace std;
int main() {
  // Mendeklarasikan array A dengan 5 buah elemen bertipe int
  int A[5];

  // Mengisikan nilai ke dalam elemen array
  cout<<"Masukkan nilai yang diinginkan"<<endl;
  for (int C=0; C<5; C++) {
    cout<<"A["<<C<<"] = "; cin>>A[C];
  }
  cout<<'\n';

  // Menampilkan nilai yang terdapat dalam elemen array
  cout<<"Menampilkan nilai yang telah dimasukkan"<<endl;
  for (int J=0; J<5; J++) {
    cout<<"Nilai yang terdapat pada elemen ke-";
    cout<<J+1<<": "<<A[J]<<endl;
  }

  return 0;
}
```

MELAKUKAN INISIALISASI ARRAY

- Pada saat ***array*** dideklarasikan maka dapat langsung dilakukan inisialisasi nilai terhadap ***elemen-elemen array*** didalamnya.
- Hal ini dimaksudkan untuk mengisikan ***nilai awal (default)*** pada ***elemen array*** sehingga jika elemen bersangkutan tidak diisi dengan nilai baru maka nilai yang digunakan adalah nilai yang telah ada.

MELAKUKAN INISIALISASI ARRAY

- Bentuk umum dari proses ***inisialisasi array*** adalah sebagai berikut :

```
tipe_data nama_array[N] = {nilai1, nilai2, …, nilaiN};
```

MELAKUKAN INISIALISASI ARRAY

```
#include <iostream>
using namespace std;
int main() {
  // Mendeklarasikan array dan langsung menginisialisasi nilainya
  int A[5] = { 10, 20, 30, 40, 50 };

  // Menampilkan nilai yang terdapat pada elemen array
  cout<<"Sebelum dilakukan perubahan nilai"<<endl;
  cout<<"A[0] = "<<A[0]<<endl;
  cout<<"A[1] = "<<A[1]<<endl;
  cout<<"A[2] = "<<A[2]<<endl;
  cout<<"A[3] = "<<A[3]<<endl;
  cout<<"A[4] = "<<A[4]<<endl;

  // Mengubah elemen ke-1 dan ke-2
  A[0] = 12;
  A[1] = 25;

  // Menampilkan kembali nilai yang terdapat pada elemen array
  cout<<"Setelah dilakukan perubahan nilai"<<endl;
  cout<<"A[0] = "<<A[0]<<endl;
  cout<<"A[1] = "<<A[1]<<endl;
  cout<<"A[2] = "<<A[2]<<endl;
  cout<<"A[3] = "<<A[3]<<endl;
  cout<<"A[4] = "<<A[4]<<endl;
  return 0;
}
```

PENCARIAN PADA ARRAY

- Salah satu permasalahan yang sering muncul pada saat kita menggunakan ***array*** adalah tuntutan untuk melakukan ***pencarian elemen array***.
- Contoh : pencarian data mahasiswa disalah satu perguruan tinggi, pencarian data nasabah bank, pencarian nilai terbesar atau terkecil dari sekumpulan bilangan dll

PENCARIAN PADA ARRAY

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

int main() {
  // Mendeklarasikan array dgn melakukan inisialisasi nilai ke dalamnya
  int A[10] = { 12, 24, 14, 25, 10,
                13, 21, 20, 15, 18 };
  int CARI;   // Variabel untuk menampung nilai yang akan dicari

  // Menampilkan nilai yang terdapat pd elemen-elemen array di atas
  for (int C=0; C<10; C++) {
    cout<<"A["<<C<<"]: "
        <<A[C]<<endl;
  }
  cout<<endl;

  // Memasukkan nilai yang akan dicari
  cout<<"Masukkan nilai yang akan dicari : ";
  cin>>CARI;

  // Melakukan pencarian data
  for (int J=0; J<10; J++) {
    if (A[J] == CARI) {
      cout<<"Nilai yang dicari "
          <<"terdapat pada indeks ke-"<<J;
      break;
    }
  }

  return 0;
}
```

MENGURUTKAN ELEMEN ARRAY

- Apabila kita sudah mempelajari ***algoritma pemrograman*** maka pasti sudah dipelajari bahwa ***array*** dapat ***diurutkan*** dengan beberapa metoda diantaranya ***metoda gelembung (bubble short), maksimum-minimum (maximum-minimum short) , heap short,*** dll.
- Salah satu kegunaan dari ***pengurutan*** adalah ***untuk mempermudah dan mempercepat proses pencarian data.***

ARRAY KONSTAN

- Nilai dari ***elemen array*** dapat dibuat tetap, yaitu dengan mendefinisikannya sebagai ***konstanta***.
- Caranya adalah dengan menggunakan keyword **`const`** didepan ***nama array*** yang akan didefinisikan.

ARRAY KONSTAN

```
#include <iostream>

using namespace std;

int main() {

  // Mendeklarasikan array yang bersifat konstan
  const int A[5] = { 10, 20, 30, 40, 50 };

  // Menampilkan nilai yang terdapat pada array A
  for (int C=0; C<5; C++) {
    cout<<"A["<<C<<"] = "<<A[C]<<endl;
  }

  return 0;
}
```

ARRAY TIPE DATA BENTUKAN

- Dalan C++, array juga dapat digunakan sebagai ***tipe data*** bentukan seperti halnya ***struktur*** dan ***enumerasi***.
- Untuk melakukan ini perlu digunakan ***keyword*** **`typedef`** yang berfungsi untuk memberikan nama lain dari ***array*** yang dideklarasikan.
- Bentuk umum penulisanya adalah sebagai berikut :

```
Typedef tipe_data nama_array[jumlah_elemen]
```

ARRAY TIPE DATA BENTUKAN

```
#include <iostream>

using namespace std;

int main() {

  // Mendeklarasikan tipe data berbentuk array dengan nama LARIK
  typedef int LARIK[5];

  // Menggunakan tipe data LARIK untuk mendeklarasikan variabel A
  LARIK A;

  for (int C=0; C<5; C++) {
    // Mengisikan nilai elemen ke dalam variabel A
    A[C] = (C+1) * 100;

    // Menampilkan nilai elemen yang terdapat pada variabel A
    cout<<"A["<<C<<"] = "<<A[C]<<endl;
  }

  return 0;
}
```

ARRAY TIPE DATA BENTUKAN

- Seperti terlihat di atas bahwa sebuah ***array*** dapat digunakan untuk mendeklarasikan ***variabel*** lain
- Pada kasus ini dibuat sebuah tipe data bentukan dengan nama **`LARIK`**, selanjutnya digunakan tipe tersebut untuk mendeklarasikan variabel A.
- Apabila diamati sebenarnya dapat langsung dideklarasikan varibel A tersebut dengan tipe array :

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

ARRAY TIPE DATA BENTUKAN

- Mungkin bisa jadi bingung kenapa harus dituliskan sebagai tipe data ? Jawabannya adalah karena tipe tersebut dapat digunakan untuk mendeklarasikan variabel dengan cepat.
- Sebagai contoh : misalnya akan dideklarasikan 5 buah variabel (dengan nama A, B, C, D, dan E) yang bertipe ***array*** dari tipe **`int`** dan terdiri dari 5 elemen.

ARRAY TIPE DATA BENTUKAN

- Jika ditulis dengan deklarasi biasa maka programnya akan tampak sebagai berikut :

```
Int A[5];
Int B[5];
Int C[5];
Int D[5];
Int E[5];
```

ARRAY TIPE DATA BENTUKAN

- Namun jika sebelumnya kita mendefinisikan sebuah tipe data yang berbentuk array (misalnya dengan nama LARIK), maka kodenya akan terlihat lebih rapi yaitu sebagai berikut :

```
// Membuat tipe data dengan nama LARIK
Typedef LARIK[5];
//Mendeklarasikan variabel bertipe LARIK
LARIK A, B, C, D, E;
```

ARRAY DARI KARAKTER

- Dalam C++, kumpulan karakter disebut dengan string (teks) dan untuk mendeklarasikan ***array*** dari ***tipe karakter*** maka tentu akan menuliskannya ke dalam bentuk seperti dibawah ini :

```
char nama_array[jumlah_elemen]
```

- Dengan demikian apabila akan dideklarasikan ***variabel string*** (misalkan dengan nama **TEKS**) yang terdiri dari 4 buah karakter, maka sintaksnya :

```
char TEKS[4] = {'B', 'I', 'S', '\0};
```

ARRAY DARI KARAKTER

- Karakter terakhir, `'\0'`, disebut dengan karakter ***null,*** yaitu karakter yang digunakan sebagai terminator dari sebuah string didalam C++.
- Namun karena bentuk tersebut susah ditulis dan riskan terhadap terjadinya sebuah kesalahan, maka C++ memperbolehkan kita untuk dapat menuliskan karakter-karakter tersebut dalam sebuah string yaitu sebagai berikut :

```
char TEKS[4] = "BIS";
```

ARRAY DARI KARAKTER

```
#include <iostream>
using namespace std;
int main() {
  // Mendeklarasikan array A dari tipe karakter
  char A[4] = { 'B','I','S','\0' };
  // Mendeklarasikan B sebagai string
  char B[4] = "BIS";
  // Menampilkan nilai dari elemen array A
  for (int C=0; C<4; C++) {
     cout<<A[C];
  }
  cout<<endl;
  // Menampilkan nilai dari variabel B
  cout<<B;
  cout<<"\n\n";
  // Menampilkan ukuran dari A dan B
  cout<<"Ukuran A: "<<sizeof(A)<<endl;
  cout<<"Ukuran B: "<<sizeof(B);
  return 0;
}
```

ARRAY DARI STRUKTUR DAN STRUKTUR DARI ARRAY

- Bagi kebanyakan programmer pemula terkadang istilah ***array dari struktur*** (***array of structure***) dan ***struktur dari array (structure of array)*** ini agak sedikit membingungkan.
- Sebenarnya konsepnya sederhana, ***array*** dari ***struktur*** berarti dideklarasikan sebuah ***array*** yang nilai dari setiap elemennya bertipe ***struktur***, sedangkan ***struktur*** dari ***array*** berarti dideklarasikan sebuah ***struktur*** yang anggotanya bertipe ***array.***

ARRAY DARI POINTER DAN POINTER DARI ARRAY

- Dalam C++, ***elemen array*** dapat berbentuk ***pointer*** dan ***pointer*** juga dapat menunjuk ke ***tipe array***.
- Apabila dibuat ***array*** dari ***pointer***, artinya akan dideklarasikan ***pointer*** sebanyak jumlah ***elemen array*** yang ditentukan, misal :

```
int *P[5];
```

- Sintaks tersebut menunjukan bahwa dimiliki ***5 buah alamat memori*** yang akan ditunjuk oleh ***pointer P*** yang tentunya tidak menunjuk semua alamat secara sekaligus melainkan harus digunakan ***operator*** **`new`**

ARRAY DARI POINTER DAN POINTER DARI ARRAY

- Sekarang apabila dideklarasikan ***pointer ke array***, itu artinya dideklarasikan sebuah pointer yang akan digunakan untuk menunjuk ke ***variabel*** yang bertipe ***array***.

```
Int A[5]; // Array yang terdiri dari 5 buah
          // elemen bertipe int
Int *P;   // Mendeklarasikan pointer P
P = A;    // Mendeklarasikan P untuk menunjuk
          // array A
```

ARRAY DARI POINTER DAN POINTER DARI ARRAY

- Mungkin muncul kebingungan, kenapa **P =A**, bukan **P = &A**; ? Hal ini disebabkan karena dalam C++, ***array*** sebenarnya adalah suatu pointer. Jadi kode tadi sama saja jika dituliskan sbb :

```
int A[5];
int *P;
P = &A[0];
```

- Pada kode diatas, pointer P akan menunjuk ke elemen array yaitu A[0]

POINTER DARI ARRAY

```
#include <iostream>

using namespace std;

int main() {

   int A[5];
   int *P = A;

   cout<<"Nilai &A[0] : "<<&A[0]<<endl;
   cout<<"Nilai P   : "<<P;

   return 0;
}
```

ARRAY MULTIDIMENSI

- ***Array multidimensi*** yaitu ***array*** yang terdiri dari beberapa ***subskrip***.
- Sebagai contoh, ***array 2 dimensi*** adalah ***array yang mempunyai 2 subskrip***, ***3 dimensi mempunya 3 subskrip***, dan seterusnya.
- ***Array*** seperti ini sering digunakan untuk ***pemrosesan matrik***.

ARRAY DUA DIMENSI

- ***Array dua dimensi*** adalah array yang mempunyai ***dua buah subskrip***, yaitu baris dan kolom.
- Bentuk umum pendeklarasiannya adalah sebagai berikut :

```
Type_data nama_array[jml_elemen_bar] [jml_elemen_kol]
```

INNISIALISASI PADA ARRAY MULTIDIMENSI

- Sama halnya seperti array satu dimensi, pada array multidimensi juga dapat dilakukan inisialisasi nilai kedalam elemen-elemennya.
- Contoh :

```
Int a[3][3] = {1,2,3,4,5,6,7,8,9};
```

- Namun untuk memudahkan proses inisialisasi, C++ mengizinkan untuk melakukan pengelompokan untuk setiap baris, yaitu sbb :

```
Int a[3][3] = {{1,2,3}, {4,5,6} ,{7,8,9}};
```

INNISIALISASI PADA ARRAY MULTIDIMENSI

```
#include <iostream>

using namespace std;

int main() {

  // Melakukan inisialisasi nilai
  // ke dalam elemen-elemen array dua dimensi
  int A[3][3] = { {1,2,3}, {4,5,6}, {7,8,9} };

  // Menampilkan nilai yang tersimpan
  // dalam elemen array
  for (int j=0; j<3; j++) {
    for (int k=0; k<3; k++) {
      cout<<"A["<<j<<"]["<<k<<"] = "
          <<A[j][k]<<endl;
    }
    cout<<endl;
  }

  return 0;
}
```