Modul
Pemrograman Terstruktur

27.02.2023

Jurusan Ilmu Komputer
Fakultas Matematika dan Ilmu Pengetahuan Alam
Universitas Lampung
2023

Array

Array adalah tipe data yang digunakan untuk menyimpan sejumlah variabel dengan tipe data yang **sama** dalam satu lokasi memori, yang diidentifikasi oleh sebuah nama. Array dapat digunakan untuk mempermudah penggunaan variabel dalam program.

Capaian Pembelajaran

1. Mahasiswa dapat memahami konsep Array dan cara menggunakannya
2. Mahasiswa dapat mengimplementasikan Array ke dalam program sesuai dengan kebutuhan program.

Materi

ARRAY SATU DIMENSI

Array Satu Dimensi atau array linier) adalah sekumpulan 'n' bilangan homogen hingga elemen data misalnya :

```
int myArray[5];
```

tipe_data nama_array[ukuran_array] = {nilai_pertama, nilai_kedua, ...,};

```
int myArray[5] = {1, 2, 3, 4, 5};
```

Contoh dalam array dengan tipe data char

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

int main() {
  char name[20];

  cout << "Please enter your name: ";
  cin >> name;

  cout << "Hello, " << name << "!" << endl;
```

Kode program tersebut merupakan tipe data yang sama. Berikut contoh program menggunakan array.

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

int main() {
   int myArray[5] = {10, 20, 30, 40, 50};
   for(int i = 0; i < 5; i++) {
      cout << "Value of element " << i << " is " << myArray[i] << endl;
   }
   return 0;
}
```

Pada contoh di atas, array myArray berisi 5 elemen dengan nilai awal 10, 20, 30, 40, dan 50. Dalam loop for, kita mengakses setiap elemen dalam array dengan menggunakan indeks i dari 0 hingga 4, dan menampilkan nilai setiap elemen dengan menggunakan cout. Dalam hal ini struktur data.

Array Dua Dimensi

Berikut ini adalah contoh penggunaan array dua dimensi dalam bahasa C++:

```cpp
// Deklarasi array 2 dimensi dengan ukuran 3 x 4
int arr[3][4];
```

Memasukkan value ke dalam array satu persatu dengan inputan :

```cpp
for(int i = 0; i < 3; i++)
{
    for(int j = 0; j < 4; j++)
    {
        arr[i][j] = i * j;
    }
```

Menampilkan output array yang telah diberi masukkan:

```cpp
for(int i = 0; i < 3; i++)
{
    for(int j = 0; j < 4; j++)
    {
        cout << arr[i][j] << " ";
    }
    cout << endl;
}
```

Studi kasus :

1. Bagaimana anda menjumlahkan array tersebut?

2. Berikan contoh program yang anda buat pada lembar praktikum.

Array passing by value

Berikut adalah contoh penggunaan passing by value untuk array pada bahasa C++ :

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

void printArray(int *arr, int size) {
  for(int i = 0; i < size; i++) {
    cout << arr[i] << " ";
  }
  cout << endl;
}

int main() {
  int arr[] = {1, 2, 3, 4, 5};
  int size = sizeof(arr)/sizeof(int);

  printArray(arr, size);

  return 0;
}
```

Pointer

Sebuah pointer adalah variabel yang nilainya adalah alamat dari variabel lain, yaitu alamat langsung dari lokasi memori. Seperti variabel atau konstanta lainnya, Anda harus mendeklarasikan sebuah pointer sebelum menggunakannya untuk menyimpan alamat variabel apapun.

Capaian Pembelajaran

1. Mahasiswa dapat memahami konsep Pointer dan cara menggunakannya
2. Mahasiswa dapat mengimplementasikan Pointer ke dalam program sesuai dengan kebutuhan program.

Materi

Ada beberapa operasi penting yang akan kita lakukan dengan bantuan pointer secara sangat sering. (a) Kita mendefinisikan variabel pointer, (b) memberikan alamat dari sebuah variabel ke pointer, dan (c) akhirnya mengakses nilai pada alamat yang tersedia di dalam variabel pointer.

Pointer adalah variabel yang menyimpan alamat variabel lain dalam memori komputer. Dalam penggunaan pointer di bahasa C++ digunakan dua operator yaitu Address-of Operator (&) dan Dereference operator (*). Pointer berkaitan dengan Array, tetapi pointer membawa banyak keuntungan ketimbang Array. Adapun keuntungannya itu sebagai berikut

Bentuk umum dari deklarasi variabel pointer adalah:

```
tipe_data* nama_pointer;
```

Contohnya, jika ingin mendeklarasikan sebuah pointer untuk menunjuk ke sebuah integer, Anda dapat menuliskannya seperti ini:

```
int* ptr1, *ptr2, *ptr3;
```

Contoh inisialisasi pointer dengan memberikan alamat memori dari sebuah variabel:

```
int a = 5;
int* ptr = &a;  // inisialisasi pointer dengan alamat memori variabel a
```

Contoh inisialisasi pointer dengan alokasi memori secara dinamis menggunakan tipe data lain sebagai berikut :

```
int *ip;     // pointer to an integer
double *dp;  // pointer to a double
float *fp;   // pointer to a float
char *ch;    // pointer to a character
```

Cara menggunakan Pointer

Berikut adalah contoh program yang menggunakan deklarasi pointer untuk integer, double, float:

```
int num = 10;
double dbl = 3.14;
float flt = 2.5f;
```

```
// assign address of variables to pointers
ip = &num;
dp = &dbl;
fp = &flt;
ch = &chr;
```

Berikut cara print value and memory address of variables :

```
cout << "Integer value: " << *ip << ", memory address: " << ip << endl;
cout << "Double value: " << *dp << ", memory address: " << dp << endl;
cout << "Float value: " << *fp << ", memory address: " << fp << endl;
cout << "Char value: " << *ch << ", memory address: " << (void*)ch << endl;
```

Output program ini akan menampilkan nilai dan alamat memori dari variabel yang dideklarasikan sebagai pointer. Perhatikan bahwa operator & digunakan untuk mendapatkan alamat memori dari variabel, sedangkan operator * digunakan untuk mendapatkan nilai yang disimpan di alamat yang ditunjuk oleh pointer. Output dari program ini akan terlihat seperti ini:

```
Integer value: 10, memory address: 0x61fe1c
Double value: 3.14, memory address: 0x61fe10
Float value: 2.5, memory address: 0x61fe14
Char value: A, memory address: 0x61fe1b
```

Dereferencing

Jika kita menggunakan operator `&` pada sebuah pointer, maka hasilnya adalah alamat memori dari variabel atau objek yang ditunjuk oleh pointer tersebut. Namun, jika Anda menggunakan operator `&` pada sebuah variabel atau objek, maka hasilnya adalah alamat memori dari variabel atau objek tersebut. Dalam contoh kode diberikan sebelumnya, kita memiliki variabel **x** dan pointer **ptr** yang menunjuk ke alamat memori dari variabel tersebut. Jika kita ingin mendapatkan alamat memori dari pointer **ptr**.

kita dapat menggunakan operator & seperti ini:

```
int x = 10;
int* ptr = &x;
cout << "Alamat memori dari pointer ptr = " << &ptr << endl;
```

Dalam kedua contoh di atas, kita menggunakan operator & untuk mendapatkan alamat memori, namun hasilnya berbeda tergantung pada apakah operator & digunakan pada pointer atau variabel. Maka outputnya adalah alamat memori dari variabel tersebut.

Penggunaan Pointer dengan Array

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

int main() {
    int arr[5] = {10, 20, 30, 40, 50};
    int *ptr = arr; // ptr menunjuk ke alamat memori awal dari array

    for(int i = 0; i < 5; i++) {
        cout << "Value of arr[" << i << "] = " << *ptr << endl;
        ptr++; // pindah ke alamat memori berikutnya
    }

    return 0;
}
```

Pada contoh di atas, ptr awalnya menunjuk ke alamat memori awal dari array arr. Kemudian, kita menggunakan operator dereferencing * untuk mengakses nilai dari setiap elemen dalam array. Pada setiap iterasi loop, ptr ditingkatkan untuk menunjuk ke alamat memori elemen array berikutnya. Dengan menggunakan pointer dan dereferencing, kita dapat melakukan manipulasi array yang lebih fleksibel dan efisien.

Maka outputnya :

```
Value of arr[0] = 10
Value of arr[1] = 20
Value of arr[2] = 30
Value of arr[3] = 40
Value of arr[4] = 50
```

Daftar Pustaka

1. Learn C programming Language by Tutorials Point
2. Geek for Geeks
3. W3School
4. https://pythontutor.com/visualize