Modul Praktikum

Alpro Lanjut

2

ARRAY

DASAR TEORI

Array adalah kumpulan dari variabel yang memiliki tipe data yang sama. Array ini bisa diakses dengan menggunakan index. Index dimulai dari 0.

Tiga hal penting dalam deklarasi array:

1. Tipe Data: Tipe data dari elemen array.
2. Nama Array: Nama dari array.
3. Ukuran Array: Jumlah elemen dari array.

Struktur deklarasi array adalah sebagai berikut:

```
tipe_data nama_array[ukuran_array];
```

Contoh:

```cpp
#include <iostream>
using namespace std;
int main() {
    // Deklarasi array dan inisialisasi elemen array
    string nama[4] = {"Haqi", "Dzaki", "Musang"};
    // Deklarasi array tanpa inisialisasi elemen array
    int angka[5];
    angka[0] = 10;
    angka[1] = 19;
    angka[2] = 30;
    angka[3] = 40;
    angka[4] = 50;
    return 0;
}
```

Operasi mengakses dan penyisipan elemen dengan data secara acak menggunakan array, operasi yang digunakan pada array, yaitu:

1. Operasi penyisipan atau insertion: Memasukkan data ke dalam elemen array, dengan menggunakan operator assignment.

   Contoh:

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

   Output:

```
Haqi
Dzaki
Musang
Rasyid
```

2. Operasi pengaksesan atau access: dengan indeks tertentu pada array dilakukan upaya pengambilan nilai dari elemen.

   Contoh:

```
cout << nama[0] << endl;
cout << angka[2] << endl;
for (string x : nama) {
    cout << x << endl;
}
```

   Output:

```
Haqi
30
Haqi
Dzaki
Musang
Rasyid
```

Selain itu, terdapat cara mengetahui panjang data pada array.

```
int angka[5] = {1, 2, 3, 4, 5};
cout << sizeof(angka);
```

Output:

```
20
```

Kenapa hasilnya 20? Karena tiap elemen array int butuh 4 bytes. Jadi, 5 elemen array int butuh 20 bytes (4 bytes * 5 elemen array). Untuk contoh setiap elemen array int butuh 4 bytes tertera pada contoh di bawah.

```
cout << sizeof(angka[0]);
```

Output:

```
4
```

Jadi gimana cara dapetin panjang array yang sebenarnya? Tinggal kita bagi aja dengan ukuran dari satu biji elemen array.

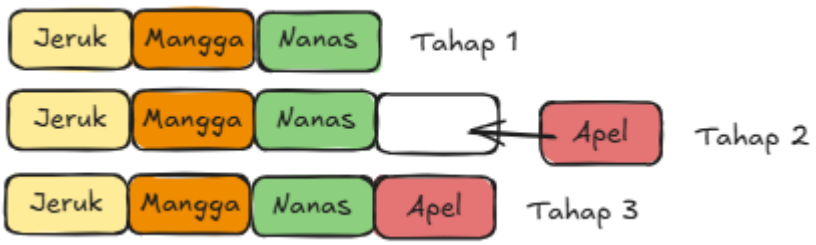
Misal contoh kita di atas, panjang bytes array angka adalah 20, dan ukuran dari satu biji elemen array int adalah 4 bytes. Jadi, panjang array angka adalah 20 / 4 = 5.

```
int angka[5] = {1, 2, 3, 4, 5};
int panjangArray = sizeof(angka) / sizeof(angka[0]);
cout << "Panjang dari array angka adalah: " <<panjangArray << endl;
```

I. ILUSTRASI ARRAY

Berikut adalah Ilustrasi dari operasi Create, Read, Update dan Delete pada sebuah Array.

Create



Tahap 1: Kondisi Awal Array Setelah ditambahkan 3 elemen

Tahap 2: Tambahkan Apel pada indeks ke 3 (indeks dimulai dari 0 yang mana diisi oleh Jeruk)

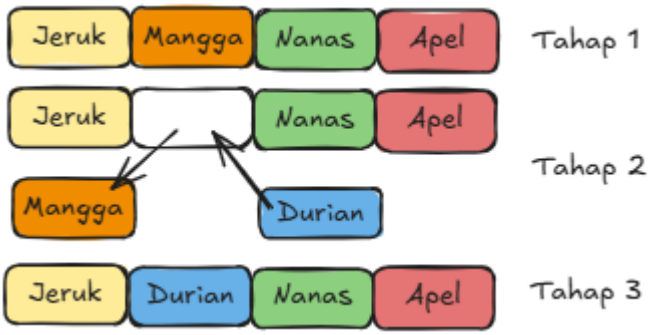
Tahap 3: Kondisi Array setelah ditambahkan Apel diindeks ke 3

Read

Tahap 1: Kondisi Awal array yang berisikan 4 elemen buah

Tahap 2: tampilkan masing-masing buah mudah dari indeks = 0 atau buah jeruk

Update

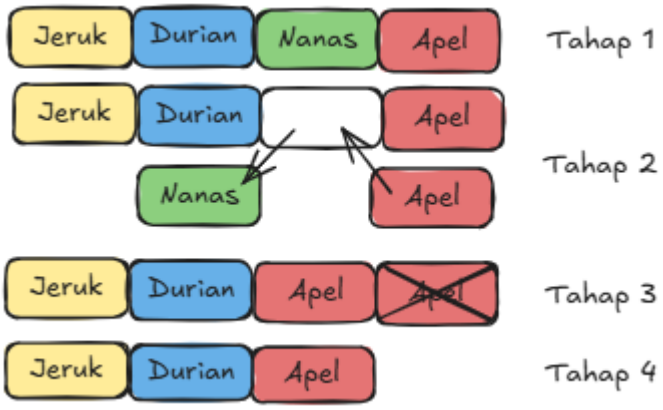


Tahap 1: Kondisi Awal Array yang berisikan 4 buah

Tahap 2: Jika kita ingin mengubah Mangga (indeks 1), maka Timpa buah Mangga dengan buah Durian

Tahap 3: Kondisi array buah setelah buah Mangga diubah menjadi buah Durian

Delete



Tahap 1: Kondisi Awal Array yang berisikan 4 buah

Tahap 2: Jika kita ingin menghapus Nanas pada indeks 2, maka timpa buah Nanas dengan Apel

Tahap 3: Setelah ditimpa, kita kurangi Panjang Array buah dari 4 menjadi 3 karena buah Nanas sudah dihapus

Tahap 4: Kondisi Akhir setelah buah Nanas dihapus

2. ARRAY SATU DIMENSI

Array satu dimensi adalah jenis array dasar yang terdiri dari beberapa kolom elemen. Dalam satu baris tersusun dari beberapa elemen-elemen yang sama. Keunggulan dari array satu dimensi adalah mudah digunakan dan mudah dibaca sehingga paling umum digunakan.

Syntax Array 1 dimensi :

TipeData NamaArray[jumlah elemen]

Contoh Array 1 Dimensi :

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

#define MAX_BUAH 100 // Ukuran maksimum array buah

int main() {
  int panjang = 0; // Jumlah elemen saat ini
  string buah[MAX_BUAH]; // Array dengan ukuran tetap

  int pilihan, index;
  do {
    system("clear");
    cout << "Menu Program" << endl;
    cout << "1. Tampilkan Buah" << endl;
    cout << "2. Tambah Buah" << endl;
    cout << "3. Ubah Buah" << endl;
    cout << "4. Hapus Buah" << endl;
    cout << "5. Keluar" << endl;
    cout << "Pilihan: ";
    cin >> pilihan;

    switch (pilihan) {
    case 1:
      // Tampilkan buah
      break;
    case 2:
      // Tambah buah
      break;
    case 3:
      // Ubah buah
      break;
```

```
    case 4:
      // Hapus buah
      break;
    case 5:
      cout << "Program selesai" << endl;
      break;
    default:
      cout << "Pilihan tidak valid" << endl;
      break;
    }
  } while (pilihan != 5);

  return 0;
}
```

Cara memanipulasi elemen dengan data secara acak menggunakan array, operasi yang digunakan pada array, yaitu:

1. Fungsi Create

```
if (panjang < MAX_BUAH) {
    cout << "Masukkan nama buah: ";
    cin.ignore();
    getline(cin, buah[panjang]);
    panjang++;
    cout << "Buah berhasil ditambahkan" << endl;
} else {
    cout << "Kapasitas penuh! Tidak bisa menambah buah lagi." << endl;
}
```

2. Fungsi Read

```
if (panjang == 0) {
    cout << "Belum ada buah" << endl;
} else {
    for (int i = 0; i < panjang; i++) {
        cout << "Buah ke-" << i + 1 << ": " << buah[i] << endl;
    }
}
```

3. Fungsi Update

```
if (panjang == 0) {
    cout << "Belum ada buah untuk diubah." << endl;
} else {
    for (int i = 0; i < panjang; i++) {
      cout << "Buah ke-" << i + 1 << ": " << buah[i] << endl;
    }
    cout << "Masukkan nomor buah yang akan diubah: ";
    cin >> index;

    if (index > 0 && index <= panjang) {
        cout << "Masukkan nama buah baru: ";
        cin.ignore();
```

```
        getline(cin, buah[index - 1]);
        cout << "Buah berhasil diubah" << endl;
    } else {
        cout << "Nomor buah tidak valid" << endl;
    }
}
```

4. Fungsi Delete

```
if (panjang == 0) {
    cout << "Belum ada buah untuk dihapus." << endl;
} else {
    for (int i = 0; i < panjang; i++) {
      cout << "Buah ke-" << i + 1 << ": " << buah[i] << endl;
    }
    cout << "Masukkan nomor buah yang akan dihapus: ";
    cin >> index;

    if (index > 0 && index <= panjang) {
      for (int i = index - 1; i < panjang - 1; i++) {
        buah[i] = buah[i + 1]; // Ngegeser elemen ke kiri kek nimpa jadinya
      }
      panjang--;
      cout << "Buah berhasil dihapus" << endl;
    } else {
      cout << "Nomor buah tidak valid" << endl;
    }
}
```

3. ARRAY DUA DIMENSI

Array dua dimensi merupakan perluasan dari array satu dimensi sehingga array dua dimensi terdiri dari kolom dan baris atau berbentuk matrix. Maka harus perhatikan nomor baris dan kolom.

Sintaks Array 2 Dimensi :

TipeData NamaArray[jumlah baris][jumlah kolom]

Contoh Array 2 Dimensi

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

Looping dengan Array 2 Dimensi

```
for (int i = 0; i < 2; i++) { // Loop untuk baris
     for (int j = 0; j < 2; j++) { // Loop untuk kolom
         cout << matriks[i][j] << " ";
     }
     cout << endl;
}

for (auto &baris : matriks) { // Iterasi setiap baris (array 1D)
     for (auto &elemen : baris) { // Iterasi setiap elemen dalam baris
         cout << elemen << " ";
         }
      cout << std::endl;
}
```

Contoh Array 2 Dimensi:

Cara mengakses elemen pada array 2 dimensi sama saja dengan array 1 dimensi, hanya saja terdapat beberapa perbedaan pada penulisan indeksnya, contohnya sebagai berikut.

`Berikut adalah contoh CRUD dari Array 2 dimensi:

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

#define MAX_BUAH 100 // Ukuran maksimum array buah
#define INFO 2 // Kolom untuk menyimpan informasi tambahan (misal: nama dan
```

```
kategori)

int main() {
    int panjang = 0; // Jumlah elemen saat ini
    string buah[MAX_BUAH][INFO]; // Array 2 dimensi

    int pilihan, index;
    do {
        system("clear");
        cout << "Menu Program" << endl;
        cout << "1. Tampilkan Buah" << endl;
        cout << "2. Tambah Buah" << endl;
        cout << "3. Ubah Buah" << endl;
        cout << "4. Hapus Buah" << endl;
        cout << "5. Keluar" << endl;
        cout << "Pilihan: ";
        cin >> pilihan;

        switch (pilihan) {
        case 1:
            // Tampilkan semua buah
            break;
        case 2:
            // Tambah buah
            break;
        case 3:
            // Ubah buah
            break;
        case 4:
            // Hapus buah
            break;
        case 5:
            cout << "Program selesai" << endl;
            break;
        default:
            cout << "Pilihan tidak valid" << endl;
            break;
        }
    } while (pilihan != 5);

    return 0;
}
```

Cara memanipulasi elemen dengan data secara acak menggunakan array, operasi yang digunakan pada array, yaitu:

1. Fungsi Create

```
if (panjang < MAX_BUAH) {
    cout << "Masukkan nama buah: ";
    cin.ignore();
    getline(cin, buah[panjang][0]);
    cout << "Masukkan kategori buah: ";
    getline(cin, buah[panjang][1]);
    panjang++;
    cout << "Buah berhasil ditambahkan" << endl;
} else {
    cout << "Kapasitas penuh! Tidak bisa menambah buah lagi." << endl;
}
```

2. Fungsi Read

```
if (panjang == 0) {
    cout << "Belum ada buah" << endl;
} else {
    for (int i = 0; i < panjang; i++) {
        cout << "Buah ke-" << i + 1 << ": " << buah[i][0] << " (" << buah[i][1]
<< ")" << endl;
    }
}
```

3. Fungsi Update

```
if (panjang == 0) {
    cout << "Belum ada buah untuk diubah." << endl;
} else {
    for (int i = 0; i < panjang; i++) {
        cout << "Buah ke-" << i + 1 << ": " << buah[i][0] << " (" << buah[i][1]
<< ")" << endl;
    }
    cout << "Masukkan nomor buah yang akan diubah: ";
    cin >> index;
```

```
    if (index > 0 && index <= panjang) {
        cout << "Masukkan nama buah baru: ";
        cin.ignore();
        getline(cin, buah[index - 1][0]);
        cout << "Masukkan kategori baru: ";
        getline(cin, buah[index - 1][1]);
        cout << "Buah berhasil diubah" << endl;
    } else {
        cout << "Nomor buah tidak valid" << endl;
    }
}
```

4. Fungsi Delete

```
if (panjang == 0) {
    cout << "Belum ada buah untuk dihapus." << endl;
} else {
    for (int i = 0; i < panjang; i++) {
        cout << "Buah ke-" << i + 1 << ": " << buah[i][0] << " (" << buah[i][1]
<< ")" << endl;
    }
    cout << "Masukkan nomor buah yang akan dihapus: ";
    cin >> index;

    if (index > 0 && index <= panjang) {
        for (int i = index - 1; i < panjang - 1; i++) {
            buah[i][0] = buah[i + 1][0];
            buah[i][1] = buah[i + 1][1];
        }
        panjang--;
        cout << "Buah berhasil dihapus" << endl;
    } else {
        cout << "Nomor buah tidak valid" << endl;
    }
}
```

4. ARRAY MULTI DIMENSI

Array multidimensi digunakan untuk ke array yang lebih dari dua dimensi atau lebih. Bentuknya memiliki banyak dimensi sehingga untuk menentukan posisi elemen data tidak menggunakan indeks namun menggunakan key atau string. String merupakan array dari karakter.

Sintaks Array Multidimensi (Array 3 Dimensi) :

TipeData NamaArray[Elemen 1][Elemen 2][Elemen 3]

Contoh array multidimensi (Array 3 dimensi) :

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

int main() {
    int arr[2][2][2];
    arr[0][0][0] = 10;
    arr[0][0][1] = 20;
    arr[0][1][0] = 30;
    arr[0][1][1] = 40;
    arr[1][0][0] = 11;
    arr[1][0][1] = 22;
    arr[1][1][0] = 33;
    arr[1][1][1] = 44;

    cout << "Isi variabel arr:" << endl;
    cout << "=================" << endl;
    cout << endl;
    cout << "Element di [0][0][0]: "<< arr[0][0][0] << endl;
    cout << "Element di [0][0][1]: "<< arr[0][0][1] << endl;
    cout << "Element di [0][1][0]: "<< arr[0][1][0] << endl;
    cout << "Element di [0][1][1]: "<< arr[0][1][1] << endl;
    cout << "Element di [1][0][0]: "<< arr[1][0][0] << endl;
    cout << "Element di [1][0][1]: "<< arr[1][0][1] << endl;
    cout << "Element di [1][1][0]: "<< arr[1][1][0] << endl;
    cout << "Element di [1][1][1]: "<< arr[1][1][1] << endl;

    return 0;
}
```

STUDI KASUS

1. Studi Kasus: Rotasi Jadwal Kerja Karyawan

a. Deskripsi:

Sebuah perusahaan memiliki 7 karyawan dengan jadwal kerja bergilir selama 7 hari. Jadwal mereka disimpan dalam array 1D:

string jadwal[7]; // Menyimpan nama karyawan yang bekerja setiap hari

Karena sistem kerja bergilir, setiap hari jadwal akan **bergeser ke kiri,** dan karyawan yang sebelumnya bekerja di hari pertama akan pindah ke hari terakhir.

b. Tugas:

1. Input Jadwal Awal: Masukkan nama 7 karyawan untuk jadwal selama 7 hari pertama.
2. Rotasi Jadwal: Setiap hari, jadwal bergeser ke kiri, dan karyawan hari pertama pindah ke hari terakhir.
3. Cek Pola Jadwal: Setelah 7 hari, pastikan setiap karyawan mendapatkan giliran yang sama.
4. Hitung Total Shift Tiap Karyawan: Tampilkan jumlah shift yang diterima masing-masing karyawan setelah N hari.

c. Konsep yang Dipakai:

1. Array 1D dengan rotasi (shift left)
2. Looping untuk pergeseran data
3. Pengecekan pola dalam array

2. Studi Kasus: Manajemen Stok Barang di Gudang

a. Deskripsi:

Sebuah gudang menyimpan 10 jenis barang, masing-masing memiliki jumlah stok tertentu. Stok barang ini disimpan dalam array 1D:

int stok[10]; // Stok 10 jenis barang

Setiap harinya ada barang yang masuk dan keluar, sehingga jumlah stok berubah.

b. Tugas:

1. Input Data Stok Awal: Masukkan jumlah stok untuk 10 jenis barang.
2. Update Stok Harian: Untuk setiap barang, lakukan update stok berdasarkan jumlah barang masuk dan keluar.
3. Cek Barang Hampir Habis: Jika stok suatu barang < 5, tampilkan peringatan "Stok Hampir Habis".
4. Hitung Total Barang di Gudang: Tampilkan jumlah total semua stok yang tersisa.

c. Konsep yang Dipakai:

1. Array 1D untuk menyimpan data stok
2. Perubahan nilai elemen array berdasarkan input harian\
3. Menjumlahkan elemen dalam array