# RANGKUMAN TEORI ARRAY

## PENGERTIAN ARRAY

Array dapat diartikan sebagai sesuatu yang berbaris atau berderet-deret. Dalam bahasa pemrograman array adalah variabel bertipe data sama yang berderet-deret sedemikian rupa sehingga alamatnya saling bersambung atau bersebelahan / berdampingan (contiguous).

Sebuah array digunakan untuk memproses kumpulan data yang semuanya berjenis sama, seperti daftar suhu atau daftar nama. Array termasuk dalam struktur data statis, artinya adalah lokasi memori untuk suatu array tidak dapat ditambah atau dikurangi selama program dijalankan.

## MENDEKLARASIKAN ARRAY

Pada program C++, berikut adalah contoh array yang terdiri dari lima variabel dan bertipe data int, dapat dideklarasikan sebagai berikut:

int score[5];

Kemudian untuk mendefinisikannya sebagai berikut:

score[0], score[1], score[2], score[3], score[4];

Nomor dalam tanda kurung siku disebut indeks atau subskrip. Dalam C++, indeks diberi nomor dimulai dengan 0. Bilangan variabel yang berada di indeks array adalah ukuran array. Ketika sebuah array dideklarasikan, ukuran dari array tersebut diberikan tanda kurung siku setelah nama array. Variabel yang di indeks kemudian diberi nomor (juga menggunakan tanda kurung siku), dimulai dengan 0 dan diakhiri dengan bilangan bulat yaitu satu kurang dari ukuran array.

Semua variabel yang di indeks untuk satu array adalah bertipe data sama. Tipe ini disebut tipe dasar dari array. Contoh array skor, tipe dasarnya adalah int. Anda dapat mendeklarasikan array dan variabel biasa bersama-sama. Misalnya, mendeklarasikan dua variabel int next dan max bersamaan dengan skor array:

int score[5], next, max;

Misalnya, skor[0] sampai skor[4] adalah variabel yang di indeks. Indeks di dalam tanda kurung siku tidak perlu diberikan sebagai konstanta bilangan bulat. Anda dapat menggunakan ekspresi apa pun dalam tanda kurung siku selama ekspresi tersebut menilai ke salah satu bilangan bulat 0 sampai bilangan bulat yang kurang dari ukuran himpunan. Misalnya, berikut ini akan menetapkan nilai skor[3] sama dengan 99:

int n = 2;

skor[n + 1] = 99;

Meskipun terlihat berbeda, skor[n + 1] dan skor[3] adalah variabel sama yang di indeks dalam kode di atas. Itu karena n + 1 bernilai 3.

1. Array dalam Memori

Memori komputer terdiri dari daftar lokasi bernomor yang disebut byte. Jumlah byte dikenal sebagai address. Variabel sederhana diimplementasikan sebagai bagian dari memori yang terdiri dari beberapa byte yang berurutan. Jumlah byte ditentukan oleh jenis variabel.

Variabel sederhana dalam memori dijelaskan oleh dua bagian informasi: alamat dalam memori (memberikan lokasi byte pertama untuk variabel tersebut) dan jenis variabel, yang memberi tahu berapa byte memori yang dibutuhkan variabel. Ketika program Anda menyimpan nilai dalam variabel, yang sebenarnya terjadi adalah bahwa nilai (dikodekan sebagai nol dan satu) ditempatkan dalam byte memori yang ditetapkan ke variabel tersebut. Demikian pula, ketika sebuah variabel diberikan sebagai argumen (panggilan-oleh-referensi) ke suatu fungsi, itu adalah alamat variabel yang sebenarnya diberikan ke fungsi pemanggil.

Variabel yang di indeks array direpresentasikan dalam memori dengan cara yang sama seperti variabel biasa, tetapi dengan array ada sedikit lebih berbeda. Lokasi berbagai variabel yang di indeks array selalu ditempatkan bersebelahan dalam memori. Misalnya, lihat contoh berikut:

int a[6];

Ketika mendeklarasikan array, komputer akan mencadangkan cukup memori untuk menampung enam variabel bertipe int sesuai contoh di atas. Selain itu, komputer selalu menempatkan variabel satu per satu dalam memori. Komputer kemudian mengingat alamat variabel terindeks a[0], tetapi tidak mengingat alamat variabel terindeks lainnya. Ketika program Anda membutuhkan alamat dari beberapa variabel terindeks lainnya, komputer menghitung alamat untuk variabel terindeks lainnya dari alamat a[0].

1. Array dalam Fungsi
2. Variabel Terindeks sebagai Argumen Fungsi

Variabel indeks array dan seluruh array dapat digunakan sebagai argumen untuk fungsi dengan cara yang sama persis seperti variabel apa pun yang dapat menjadi argumen. Misalnya, program berisi deklarasi berikut:

int i, n, a[10];

Jika my\_function mengambil satu argumen bertipe int, maka bentuk di bawah ini diperbolehkan:

my\_function(n);

Karena variabel terindeks dari array a juga merupakan variabel berjenis int, seperti n, berikut ini sama-sama diperbolehkan:

my\_function(a[3]);

Ada satu kehalusan yang berlaku untuk variabel terindeks yang digunakan sebagai argumen. Misalnya, dengan mempertimbangkan pemanggilan fungsi berikut:

my\_function(a[i]);

Jika nilai i adalah 3, maka argumennya adalah a[3]. Di sisi lain, jika nilai i adalah 0, maka panggilan ini setara dengan berikut:

my\_function(a[0]);

Ekspresi terindeks dievaluasi untuk menentukan dengan tepat variabel terindeks mana yang diberikan sebagai argumen.

1. Keseluruhan Array sebagai Argumen Fungsi

Suatu fungsi dapat memiliki parameter formal untuk seluruh array sehingga ketika fungsi tersebut dipanggil, argumen yang dipasang untuk parameter formal ini adalah seluruh array. Namun, parameter formal untuk seluruh array bukanlah parameter call-by-value atau all-by-reference, tetapi jenis parameter formal baru yang disebut sebagai parameter array.

Contoh:

Deklarasi Fungsi:

void fill\_up (int a[], int size);

Prasyarat: size adalah ukuran yang dideklarasikan dari array a.

Pengguna akan mengetikkan ukuran bilangan bulat.

Kondisi akhir: Array a diisi dengan ukuran bilangan bulat dari keyboard.

Mendefinisikan Fungsi:

Menggunakan iostream

void fill\_up (int a[], int size)

{

using namespace std;

cout<<"Enter "<<size<<" numbers:\n";

for(int i = 0; i < size; i++)

{

cin>>a[i];

}

size--;

cout<<"The last array index is "<<size<<endl;

}

Parameter formal int a[] adalah parameter array. Tanda kurung siku, tanpa ekspresi indeks di dalamnya, adalah yang digunakan C++ untuk menunjukkan parameter array. Parameter array bukanlah parameter call-by-reference, tetapi untuk tujuan praktis dan berperilaku sangat mirip dengan parameter call-by-reference.

Ketika fungsi fill\_up() dipanggil, harus memiliki dua argumen:

1. Memberikan array bilangan bulat
2. Memberikan ukuran yang dideklarasikan dari array.

Misalnya, berikut ini adalah panggilan fungsi yang dapat diterima:

int score[5], number\_of\_scores = 5;

fill\_up (score, number\_of\_scores);

Panggilan ke fill\_up() ini akan mengisi skor array dengan lima bilangan bulat yang diketik di keyboard. Perhatikan bahwa parameter formal a[] (yang digunakan dalam deklarasi fungsi dan judul definisi fungsi) diberikan dengan tanda kurung siku, tetapi tidak ada ekspresi indeks.

Dilihat dengan cara ini, sebuah array memiliki tiga bagian: alamat dari variabel pertama yang di indeks, jenis dasar dari array, dan ukuran dari array. Ketika sebuah array digunakan sebagai argumen array untuk suatu fungsi, hanya bagian pertama dari ketiga bagian ini yang diberikan ke fungsi tersebut.

Keuntungan paling baik diilustrasikan dengan melihat kembali contoh fungsi fill\_up() yang diberikan Fungsi yang sama dapat digunakan untuk mengisi array dengan ukuran berapapun, selama tipe dasar array adalah int. Misalnya, Anda memiliki deklarasi array berikut:

int score[5], time[10];

Panggilan pertama ke fill\_up() berikut mengisi skor array dengan nilai dan yang kedua mengisi waktu array dengan nilai sepuluh:

fill\_up (score, 5);

fill\_up (time, 10);

Anda dapat menggunakan fungsi yang sama untuk argumen array dengan ukuran berbeda karena ukurannya adalah argumen terpisah.

1. Pengubah parameter konstanta

Saat Anda menggunakan argumen array dalam panggilan fungsi, fungsi tersebut dapat mengubah nilai yang disimpan dalam array. Namun, dalam definisi fungsi yang rumit, Anda mungkin menulis kode yang secara tidak sengaja mengubah satu atau lebih nilai yang disimpan dalam array, meskipun array tersebut tidak boleh diubah sama sekali. Sebagai tindakan pencegahan, Anda dapat memberi tahu compiler bahwa Anda tidak bermaksud mengubah argumen array, dan komputer kemudian akan memeriksa untuk memastikan kode yang Anda tidak sengaja mengubah nilai dalam array. Untuk memberi tahu compiler bahwa argumen array tidak boleh diubah oleh fungsi Anda, Anda memasukkan pengubah konstanta sebelum parameter array untuk posisi argumen itu. Parameter array yang dimodifikasi dengan const disebut konstanta parameter array.

1. Parameter dan Argumen Formal pada Array

Argumen ke suatu fungsi dapat berupa seluruh array, tetapi argumen untuk seluruh array bukanlah argumen call-by-value atau argumen call-by-reference. Ini adalah jenis argumen baru yang dikenal sebagai argumen array. Ketika argumen array dicolokkan untuk parameter array, semua yang diberikan ke fungsi tersebut adalah alamat dalam memori variabel terindeks pertama dari argumen array.

Argumen array tidak memberi tahu fungsi berapa ukuran array. Oleh karena itu, ketika Anda memiliki parameter array untuk suatu fungsi, biasanya Anda juga harus memiliki parameter formal lain bertipe data int yang memberikan ukuran array. Argumen array seperti argumen call-by-reference dengan cara berikut: Jika badan fungsi mengubah parameter array, maka ketika fungsi dipanggil, perubahan itu sebenarnya dilakukan pada argumen array. Dengan demikian, suatu fungsi dapat mengubah nilai argumen array yaitu, dapat mengubah nilai variabel yang di indeks.

Sintaks untuk deklarasi fungsi dengan parameter array adalah sebagai berikut:

Syntax:

Type\_Returned Function\_Name (..., Base\_Type Array\_Name[], ...);

Contoh:

void show\_the\_world (int a[], int size\_of\_a);

Misalnya, fungsi berikut mengeluarkan nilai dalam array tetapi tidak mengubah nilai dalam array:

Prasyarat: size\_of\_a adalah ukuran yang dideklarasikan dari array a.

Semua variabel yang di indeks dari a telah diberi nilai.

Kondisi akhir: Nilai dalam a telah ditulis ke layar.

void show\_the\_world (int a[], int size\_of\_a)

{

cout<<"The array contains the following values:\n";

for (int i = 0; i < size\_of\_a; i++)

{

cout<<a[i]<<"";

}

cout<<endl;

}

Fungsi ini akan bekerja dengan baik. Namun, sebagai ukuran keamanan tambahan, Anda dapat menambahkan pengubah konstanta ke heading fungsi sebagai berikut:

void show\_the\_world (const int a[], int size\_of\_a);

Dengan tambahan dari modifier const ini, komputer akan mengeluarkan pesan kesalahan jika definisi fungsi Anda mengandung kesalahan yang mengubah salah satu nilai dalam argumen array. Untungnya, versi definisi fungsi ini menyertakan modifier const, sehingga pesan kesalahan akan memberitahu kita bahwa array a diubah. Pesan kesalahan ini akan membantu menjelaskan kesalahan:

Prasyarat: size\_of\_a adalah ukuran yang dideklarasikan dari array a.

Semua variabel yang di indeks dari a telah diberi nilai.

Kondisi akhir: Nilai dalam a telah ditulis ke layar.

void show\_the\_world (const int a[], int size\_of\_a)

{

cout<<"The array contains the following values:\n";

for (int i = 0; i < size\_of\_a; a[i]++)

{

cout<<a[i]<<"";

}

cout<<endl;

}

// [i]++ adalah kesalahan, tetapi compiler tidak akan menganggapnya kecuali kalau Anda menggunakan pengubah const.

Jika kita tidak menggunakan pengubah const dalam definisi fungsi di atas dan jika kita membuat kesalahan seperti yang ditunjukkan, fungsi akan mengkompilasi dan berjalan tanpa pesan kesalahan. Namun, kode tersebut akan berisi loop tak terbatas yang terus menambahkan[0] dan menulis nilai barunya ke layar.

Masalah dengan versi show\_the\_world yang salah ini adalah item yang bertambah salah di loop for. Variabel yang di indeks a[i] bertambah, tetapi harusnya indeks i yang bertambah. Dalam versi yang salah ini, indeks i dimulai dengan nilai 0 dan nilai itu tidak pernah berubah. Tapi sebuah [i], yang sama dengan [0], bertambah. Ketika variabel yang di indeks a[i] bertambah, itu mengubah nilai dalam array, dan karena kita menyertakan pengubah const, komputer akan mengeluarkan pesan peringatan. Pesan kesalahan itu harus menjadi petunjuk tentang apa yang salah.

Anda biasanya memiliki deklarasi fungsi dalam program Anda sebagai tambahan selain definisi fungsi. Saat Anda menggunakan pengubah const dalam definisi fungsi, Anda juga harus menggunakannya dalam deklarasi fungsi agar judul fungsi dan deklarasi fungsi konsisten.

Constmodifier dapat digunakan dengan semua jenis parameter, tetapi biasanya digunakan hanya dengan parameter array dan parameter call-by-references parameter untuk class.

## ARRAY MULTIDIMENSI

C++ memungkinkan Anda untuk mendeklarasikan array dengan lebih dari satu indeks. Pada bagian ini kami menjelaskan array multidimensi. Terkadang berguna untuk memiliki array dengan lebih dari satu indeks dan ini diizinkan di C++. Berikut ini mendeklarasikan array karakter yang disebut halaman. Halaman array memiliki dua indeks:

Indeks pertama berkisar dari 0 hingga 29 dan yang kedua dari 0 hingga 99.

char page[30][100];

Variabel yang di indeks untuk array ini masing-masing memiliki dua indeks. Misalnya, page[0][0], page[15][32], dan page[29][99] adalah tiga variabel yang di indeks untuk array ini. Perhatikan bahwa setiap indeks harus diapit oleh kumpulan tanda kurung siku sendiri. Seperti halnya array satu dimensi yang telah kita lihat, setiap variabel yang di indeks untuk array multidimensi adalah variabel tipe dasar.

Sebuah array mungkin memiliki sejumlah indeks, tetapi mungkin jumlah indeks yang paling umum adalah dua. Array dua dimensi dapat divisualisasikan sebagai tampilan dua dimensi dengan indeks pertama memberikan baris dan indeks kedua memberikan kolom. Misalnya, variabel yang di indeks array dari page array dua dimensi dapat divisualisasikan sebagai berikut:

page[0][0], page[0][1], ..., page[0][99]

page[1][0], page[1][1], ..., page[1][99]

page[2][0], page[2][1], ..., page[2][99]

...

page[29][0], page[29][1], ..., page[29][99]

Anda dapat menggunakan array page untuk menyimpan semua karakter pada halaman teks yang memiliki 30 baris (bernomor 0 hingga 29) dan 100 karakter pada setiap baris (bernomor 0 hingga 99).

Berikut deklarasi array multidimensi:

Syntax:

Type Array\_Name[Size\_Dim\_1][Size\_Dim\_2]...[Size\_Dim\_Last];

Contoh:

char page[30][100];

int matrix[2][3];

double three\_d\_picture[10][20][30];

Sebuah deklarasi array, dari bentuk yang ditunjukkan di atas, akan mendefinisikan satu variabel yang di indeks untuk setiap kombinasi indeks array. Misalnya, deklarasi sampel kedua di atas mendefinisikan enam variabel terindeks berikut untuk matriks array:

matrix[0][0], matrix[0][1], matrix[0][2],

matrix[1][0], matrix[1][1], matrix[1][2];

1. Parameter Array Multidimensi

Deklarasi array dua dimensi berikut sebenarnya mendeklarasikan array satu dimensi berukuran 30, yang jenis dasarnya adalah array satu dimensi dengan karakter berukuran 100.

char page[30][100];

Melihat array dua dimensi sebagai array dari array akan membantu kita memahami bagaimana C++ menangani parameter untuk array multidimensi.

Misalnya, berikut ini adalah fungsi yang mengambil array seperti page dan mencetaknya ke layar:

void display\_page (const char p[][100], int size\_dimension\_1)

{

for (int index1 = 0; index1 < size\_dimension\_1; index1++)

{

// Mencetak satu baris:

for(int index2 = 0; index2 < 100; index2++)

{

cout<<p[index1][index2];

}

cout<<endl;

}

}

Perhatikan bahwa dengan parameter array dua dimensi, ukuran dimensi pertama tidak diberikan, jadi kita harus menyertakan parameter int untuk memberikan ukuran dimensi pertama ini. (Seperti array biasa, compiler akan mengizinkan Anda untuk menentukan dimensi pertama dengan menempatkan angka dalam pasangan pertama dari tanda kurung siku. Namun, angka seperti itu hanyalah sebuah komentar; kompilator mengabaikan nomor tersebut). Ukuran dari dimensi kedua (dan semua dimensi lainnya jika ada lebih dari dua) diberikan setelah parameter array, seperti yang ditunjukkan untuk parameter const char p[][100].

Jika Anda menyadari bahwa array multidimensi adalah array dari array, maka aturan ini mulai masuk akal. Karena parameter array dua dimensi adalah parameter untuk array-array, dimensi pertama sebenarnya adalah indeks array dan diperlakukan seperti indeks array untuk array satu dimensi biasa. Dimensi kedua adalah bagian dari deskripsi tipe dasar, yaitu array karakter berukuran 100.

Ketika parameter array multidimensi diberikan dalam heading fungsi atau deklarasi fungsi, ukuran dimensi pertama tidak diberikan, tetapi ukuran dimensi yang tersisa harus diberikan dalam tanda kurung siku. Karena ukuran dimensi pertama tidak diberikan, Anda biasanya memerlukan parameter tambahan tipe int yang memberikan ukuran dimensi pertama ini. Di bawah ini adalah contoh deklarasi fungsi dengan parameter array dua dimensi p:

void get\_page (char p[][100], int size\_dimension\_1);

# TUGAS PENDAHULUAN ARRAY

1. Apa yang dimaksud dengan array?

Array adalah sebuah variabel yang menyimpan lebih dari 1 data yang memiliki tipe data yang sama yang berderet-deret sedemikian rupa sehingga alamatnya saling bersebelahan / berdampingan (contiguous). Kemudian, dapat diakses berdasarkan indeksnya.

1. Jelaskan perbedaan arti dari int a[5]; dan arti dari a[4]. Apa arti [5] dan [4] dalam setiap kasus!

int a[5] adalah deklarasi dari variabel array a yang berukuran 5 indeks dan a[4] adalah variabel identitas ke 5 dari array a karena indeks array dimulai dari 0.

Dalam kasus ini [5] adalah panjang array yang dimulai dari 0 sampai 4 dan   
[4] adalah Panjang array yang dimulai dari 0 sampai 3.

1. Jelaskan perbedaan array dengan variabel biasa!

Array adalah sebuah variabel yang menyimpan lebih dari 1 data yang memiliki tipe data yang sama. Sedangkan Variabel biasa adalah nama lokasi suatu penyimpanan untuk menampung data yang nilainya selalu berubah.

1. Jelaskan perbedaan array satu dimensi dengan multidimensi!

Array satu dimensi adalah kumpulan elemen yang tersusun dalam suatu baris dan hanya memiliki satu indeks. Sedangkan array multidimensi adalah sebuah perluasan dari array satu dimensi yang memiliki lebih dari satu indeks.

1. Buatlah program sederhana menggunakan array dua dimensi!

