PERTEMUAN 2
ARRAY

A. TUJUAN PEMBELAJARAN

Setelah menyelesaikan pertemuan ini, mahasiswa mampu mempraktekkan:

1. Pengenalan Array
2. Mendeklarasikan Array
3. Array dalam Fungsi
4. Array Multidimensi

B. URAIAN MATERI

1. Pengertian Array

Array dapat diartikan sebagai sesuatu yang berbarisatau berderet-deret. Dalam bahasa pemrograman array adalah variable sejenis 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. Setelah menyelesaikan pertemuan ini, mahasiswa mampu:dikenalkan dasar-dasar untuk mendefinisikan dan menggunakan array dalam C ++ dan menyajikan banyak teknik dasar yang digunakan pada saat mendesain algoritma dan program yang menggunakan array.

Array termasuk dalam struktur data statis, artinya adalah lokasi memoriuntuk suatu array tidak dapat ditambah atau dikurangi selama program dijalankan.

2. Mendeklarasikan Array

Pada program C ++, array yang terdiri dari lima variabel tipe int dapat dideklarasikan sebagai berikut:

```
int score[5];
```

mendeklarasikan lima variabel berikut ke semua yang bertipe int:

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

Variabel individu yang bersama-sama menunjuk array dalam variasi dengan cara yang berbeda. Kita akan menyebutnya variabel yang diindeks, meskipun mereka juga kadang-kadang disebut variabel atau elemen yang diikutsertakan dalam array. Nomor dalam tanda kurung siku disebut indeks atau subskrip. Dalam C ++, indeks diberi nomor dimulai dengan 0, tidak dimulai dengan 1 atau angka lain apa pun kecuali 0. Bilangan variabel yang diindeks dalam array menunjuk ukuran array yang dideklarasikan, atau terkadang hanya ukuran array. Ketika sebuah array dideklarasikan, ukuran dari array tersebut diberikan tanda kurung siku setelah nama array. Variabel yang diindeks kemudian diberi nomor (juga menggunakan tanda kurung siku), dimulai dengan 0 dan diakhiri dengan bilangan bulat yaitu satu

kurang dari ukuran array. Dalam contoh ini, variabel yang diindeks berjenis int, tetapi array dapat memiliki variabel yang indeks dari jenis apa pun. Misalnya, untuk mendeklarasikan array dengan variabel berjenis double, cukup gunakan nama jenis double sebagai ganti int dalam deklarasi array. Semua variabel yang diindeks untuk satu array, bagaimanapun, adalah sama Tipe. Tipe ini disebut tipe dasar dari array. Jadi, dalam contoh array ini adalah skor, tipe dasarnya adalah int. Anda dapat mendeklarasikan array dan variabel biasa bersama-sama. Misalnya, mendeklarasikan dua variabel int next dan max selain skor array:

```
int next, score[5], max;
```

Variabel yang diindeks seperti skor [3] dapat digunakan di mana saja yang mana variabel biasa bertipe int dapat digunakan. Jangan bingung antara dua cara menggunakan tanda kurung siku [] dengan nama array. Saat digunakan dalam deklarasi, seperti int score [5]; angka yang diapit dalam tanda kurung siku menentukan berapa banyak variabel terindeks yang dimiliki array. Jika digunakan di tempat lain, nomor tersebut dimasukkan dalam tanda kurung

siku memberi tahu variabel yang diindeks mana yang dimaksud.

Misalnya, skor [0] sampai skor [4] adalah variabel yang diindeks. 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 diindeks dalam kode di atas. Itu karena n + 1 bernilai 3.

Identitas suatu variabel yang terindeks, seperti skor [i], ditentukan oleh nilai indeksnya, yang dalam hal ini adalah i. Jadi, Anda dapat menulis program yang mengatakan hal-hal seperti "lakukan ini dan itu ke variabel indeks ke-i," di mana nilai i dihitung oleh program.

a. Array dalam Memori

Sebelum membahas bagaimana array direpresentasikan dalam memori komputer, pertama-tama mari kita lihat bagaimana variabel sederhana, seperti variabel bertipe int atau double, direpresentasikan dalam memori komputer. 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.

Dengan demikian, variabel sederhana dalam memori dijelaskan oleh dua bagian informasi: alamat dalam memori (memberikan lokasi byte pertama untuk variabel itu) dan jenis variabel, yang memberi tahu berapa byte memori yang dibutuhkan variabel. Ketika kita berbicara tentang alamat variabel, alamat inilah yang sedang kita bicarakan. Ketika program Anda menyimpan nilai dalam variabel, yang sebenarnya terjadi adalah bahwa nilai (dikodekan sebagai nol dan satu) ditempatkan dalam byte memori yang

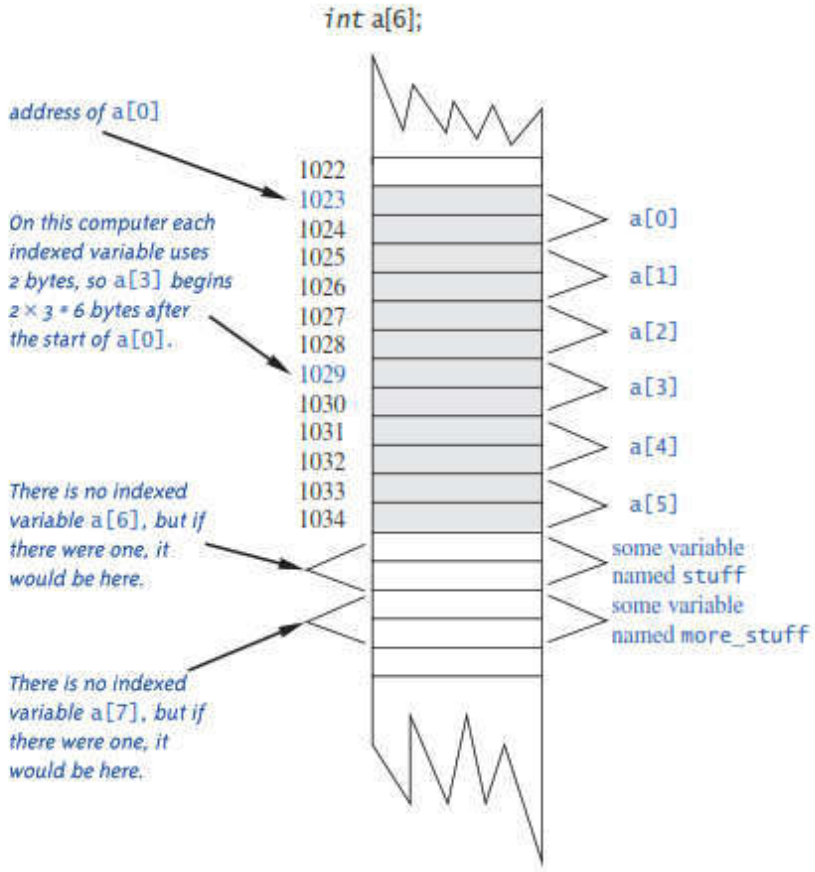
ditetapkan ke variabel itu. Demikian pula, ketika sebuah variabel diberikan sebagai argumen (panggilan-oleh-referensi) ke suatu fungsi, itu adalah alamat variabel yang sebenarnya diberikan ke fungsi pemanggil. Sekarang mari kita lanjutkan untuk membahas bagaimana array disimpan dalam memori.

Variabel yang diindeks array direpresentasikan dalam memori dengan cara yang sama seperti variabel biasa, tetapi dengan array ada sedikit lebih banyak cerita. Lokasi berbagai variabel yang diindeks larik selalu ditempatkan bersebelahan dalam memori. Misalnya, pertimbangkan hal berikut:

```
int a[6];
```

Ketika Anda mendeklarasikan array ini, komputer mencadangkan cukup memori untuk menampung enam variabel bertipe int. Selain itu, komputer selalu menempatkan variabel ini 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 ini dari alamat a [0].

Misalnya, jika Anda mulai dari alamat a [0] dan menghitung cukup memori untuk tiga variabel tipe int, maka Anda akan berada di alamat a [3]. Untuk mendapatkan alamat a [3], komputer dimulai dengan alamat a [0] (yang merupakan angka). Komputer kemudian menambahkan jumlah byte yang dibutuhkan untuk menampung tiga variabel tipe int ke nomor untuk alamat a [0]. Hasilnya adalah alamat dari [3].



Gambar 2.2Array dalam Memori

b. Array dalam Fungsi

1) Variabel Terindeks sebagai Argumen Fungsi

Kita dapat menggunakan variabel indeks array dan seluruh array sebagai argumen untuk fungsi. Kali ini kita membahas variabel yang diindeks array sebagai argumen untuk fungsi.

Variabel yang diindeks dapat menjadi argumen ke suatu fungsi dengan cara yang persis sama 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 dibawah 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.

2) 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, ini adalah jenis parameter formal baru yang disebut sebagai parameter array. Contoh :

```
Deklarasi Fungsi
voidfill_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
//Menggunakaniostream:
voidfill_up(int a[], int size)
```

```
 {
usingnamespacestd;
cout<< "Enter " <<size<< " numbers:\n";
for (int i = 0; i <size; i++)
cin>> a[i];
size--;
cout<< "The lastarrayindexusedis " <<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 ia berperilaku sangat mirip dengan parametercall-by-reference. Mari kita lihat contoh ini secara detail untuk melihat bagaimana argumen array bekerja dalam kasus ini. (Argumen array, tentu saja adalah array yang dipasang untuk parameter array, seperti [].)

Ketika fungsi fill_up dipanggil, ia harus memiliki dua argumen: Yang pertama memberikan array bilangan bulat, dan yang kedua harus 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. (Anda dapat memasukkan angka di dalam tanda kurung siku untuk parameter array, tetapi kompiler akan mengabaikan nomor tersebut, jadi kita tidak akan menggunakan angka seperti itu.) Di sisi lain, argumen yang diberikan dalam pemanggilan fungsi (skor dalam contoh ini) diberikan tanpa tanda kurung siku atau ekspresi indeks apa pun.

Apa yang terjadi pada argumen array ***score*** dalam pemanggilan fungsi ini? Singkatnya, argumen ***score*** dimasukkan untuk parameter array formal a di dalam tubuh fungsi, dan kemudian badan fungsi dijalankan. Jadi, panggilan fungsi

fill_up(score, number_of_scores);

Bisa juga dengan kode berikut :

```
#include<iostream>
Using namespace std;
Int main()
{
size = 5;                          // 5 adalah nilai dari value_of scores
cout<< "Enter " <<size<< " numbers:\n";
for (int i = 0; i <size; i++)
cin>>score[i]; size--;
cout<< "The lastarrayindexusedis " <<size<<endl;
}
```

Parameter formal a adalah jenis parameter yang berbeda dari yang telah kita lihat sebelumnya. Parameter formal a hanyalah placeholder untuk argumen ***score***. Saat fungsi fill_up dipanggil dengan ***score*** sebagai argumen array, komputer berperilaku seolah-olah a diganti dengan argumen ***score*** yang sesuai. Ketika sebuah array digunakan sebagai argumen dalam pemanggilan fungsi, tindakan apa pun yang dilakukan pada parameter array dilakukan pada argumen array, sehingga nilai variabel yang diindeks dari argumen array dapat diubah oleh fungsi tersebut. Jika parameter formal dalam badan fungsi diubah (misalnya dengan pernyataan cin), maka argumen array akan berubah.

Sejauh ini, sepertinya parameter array hanyalah parameter panggilan-demi-referensi untuk sebuah array. Itu hampir benar, tetapi

parameter array sedikit berbeda dari parametercall-by-reference. Untuk membantu menjelaskan perbedaannya, mari kita tinjau beberapa detail tentang array.

Ingatlah bahwa sebuah array disimpan sebagai potongan memori yang berdekatan. Misalnya, pertimbangkan deklarasi berikut untuk score array:

int score[5];

Saat Anda mendeklarasikan array ini, komputer mencadangkan cukup memori untuk menampung lima variabel bertipe int, yang disimpan satu demi satu di memori komputer. Komputer tidak mengingat alamat masing-masing dari lima variabel yang diindeks ini; itu hanya mengingat alamat score variabel yang diindeks [0]. Misalnya, ketika program Anda membutuhkan score [3], komputer menghitung alamat score [3] dari alamat score [0]. Komputer mengetahui bahwa score [3] terletak tiga variabel dalam score sebelumnya [0]. Jadi, untuk mendapatkan alamat score [3], komputer mengambil alamat score [0] dan menambahkan angka yang mewakili jumlah memori yang digunakan oleh tiga variabel int; hasilnya adalah alamat score [3].

Dilihat dengan cara ini, sebuah array memiliki tiga bagian: alamat (lokasi dalam memori) dari variabel pertama yang diindeks, jenis dasar dari array (yang menentukan berapa banyak memori yang digunakan setiap variabel yang diindeks), dan ukuran dari array (yaitu , jumlah variabel yang diindeks). Ketika sebuah array digunakan sebagai argumen array untuk suatu fungsi, hanya bagian pertama dari ketiga bagian ini yang diberikan ke fungsi tersebut. Ketika argumen array dicolokkan untuk parameter formal yang sesuai, semua yang dimasukkan adalah alamat variabel terindeks pertama dari array. Tipe dasar dari argumen array harus sesuai dengan tipe dasar dari parameter formal, sehingga fungsinya juga mengetahui tipe dasar dari array. Namun, argumen array tidak memberi tahu fungsi ukuran array. Ketika kode dalam tubuh fungsi dijalankan, komputer tahu di mana array dimulai di memori dan berapa banyak memori setiap variabel yang indeks gunakan , tetapi (kecuali jika Anda membuat ketentuan khusus) ia tidak tahu berapa banyak variabel terindeks yang dimiliki array. Itulah

mengapa sangat penting bahwa Anda selalu memiliki argumen int lain yang memberi tahu fungsi ukuran array. Itu juga mengapa parameter array tidak sama dengan parameter panggilan-dengan-referensi. Anda dapat menganggap parameter array sebagai bentuk lemah dari parameter call-by-reference di mana segala sesuatu tentang array diceritakan ke fungsi kecuali untuk ukuran array.

Parameter array ini mungkin tampak sedikit aneh, tetapi ada di setidaknya satu properti yang sangat bagus sebagai akibat langsung dari definisi mereka yang tampaknya aneh. Keuntungan ini paling baik diilustrasikan dengan melihat kembali contoh fungsi fill_up yang diberikan

Fungsi yang sama dapat digunakan untuk mengisi array dengan ukuran berapa pun, 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.

3) Pengubah parameter konstanta

Saat Anda menggunakan argumen array dalam panggilan fungsi, fungsi tersebut dapat mengubah nilai yang disimpan dalam array. Ini biasanya baik-baik saja. 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 konst disebut konstanta parameter array.

c. Parameter dan Argumen Formal 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 (yang diindeks oleh 0). Argumen array tidak memberi tahu fungsi ukuran array. Oleh karena itu, ketika Anda memiliki parameter array untuk suatu fungsi, biasanya Anda juga harus memiliki parameter formal lain tipe int yang memberikan ukuran array (seperti pada contoh di bawah nanti).

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 diindeks).

Sintaks untuk deklarasi fungsi dengan parameter array adalah sebagai berikut:

```
Syntax
Type_ReturnedFunction_Name(..., Base_TypeArray_Name[],...);
Contoh
voidsum_array(double& sum, double a[], int size);
```

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

```
voidshow_the_world(int a[], int size_of_a)
//Prasyarat: size_of_a adalah ukuran yang dideklarasikan dari array a.
```

```
 //Semua variabel yang diindeks dari a telah diberi nilai.
//Kondisi akhir: Nilai dalam a telah ditulis
//ke layar.
{
cout<< "The arraycontainsthefollowingvalues:\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:

voidshow_the_world(const int a[], int size_of_a)

Dengan tambahan dari modifierconst ini, komputer akan mengeluarkan pesan kesalahan jika definisi fungsi Anda mengandung kesalahan yang mengubah salah satu nilai dalam argumen array. Misalnya, berikut ini adalah versi fungsi show_the_world yang berisi kesalahan yang secara tidak sengaja mengubah nilai argumen array. Untungnya, versi definisi fungsi ini menyertakan modifierconst, sehingga pesan kesalahan akan memberi tahu kita bahwa array a diubah. Pesan kesalahan ini akan membantu menjelaskan kesalahan:

```
voidshow_the_world(const int a[], int size_of_a)
// Prasyarat: size_of_a adalah ukuran yang dideklarasikan dari array a.
// Semua variabel yang diindeks dari a telah diberi nilai.
// Kondisi akhir: Nilai dalam a telah ditulis
// ke layar
{
```

```
cout<< "The arraycontainsthefollowingvalues:\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 loopfor. Variabel yang diindeks a [i] bertambah, tetapi haruslah 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 diindeks 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-referenceparametersuntuk class.

3. Array Multidimensi

C ++ memungkinkan Anda untuk mendeklarasikan array dengan lebih dari satu indeks. Pada bagian ini kami menjelaskan array multidimensi ini.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.

charpage[30][100];

Variabel yang diindeks untuk array ini masing-masing memiliki dua indeks. Misalnya, page [0] [0], page [15] [32], dan page [29] [99] adalah tiga variabel yang diindeks 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 diindeks 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 diindeks 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

TypeArray_Name[Size_Dim_1][Size_Dim_2]...[Size_Dim_Last];

Contoh

charpage[30][100];

int matrix[2][3];

doublethree_d_picture[10][20][30];

Sebuah deklarasi array, dari bentuk yang ditunjukkan di atas, akan mendefinisikan satu variabel yang diindeks 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]

a. 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.

charpage[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:

```
voiddisplay_page(constchar 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 mengijinkan 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

constchar 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:

voidget_page(char p[][100], int size_dimension_1);

C. SOAL LATIHAN/TUGAS

1. Apa yang dimaksud dengan array.
2. Jelaskan perbedaan arti dari int a [5]; dan arti daria [4]. Apa arti [5] dan [4] dalam setiap kasus?
3. Jelaskan Perbedaan Array dengan variabel biasa.
4. Jelaskan perbedaan Array satu dimensi dengan multidimensi.
5. Buatlah program sederhana menggunakan Array dua dimensi.

D. REFERENSI

Soulie, Juan. 2007. *C++ Language Tutorial, Juni 2007. Diambil dari :*

https://www.cplusplus.com/files/tutorial.pdf (6 November 2020)

Moh.Sjukani . 2014. ALGORITMA (Algoritma & Struktur Data 1) dengan *C, C++ dan Java, Edisi 9, Mitra Wacana Media.*