Bab 6
Larik

Sub CPMK:

Setelah mempelajari materi dalam bab ini anda akan mampu menggunakan larik untuk memproses sekumpulan data yang semuanya mempunyai tipe yang sama, misalnya sekumpulan data nilai, data nama, data suhu dan sebagainya.

6.1 Pengantar Larik

Misalkan kita ingin memproses lima data nilai. Untuk keperluan ini kita bisa gunakan lima buah variabel. Akan tetapi apabila jumlah data yang akan diproses sudah besar misalnya 100 buah maka penggunaan 100 buah variabel tidaklah praktis. Penggunaan larik merupakan solusi yang sempurna. Larik berperilaku seperti sedaftar variabel dengan mekanisme penamaan uniform yang dapat dideklarasikan dengan satu baris kode sederhana.

Mendeklarasikan dan Mengacu Larik

Dalam C++, sebuah larik yang terdiri dari lima variabel betipe integer dapat dideklarasikan sebagai berikut:

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

Deklarasi ini sama dengan mendeklarasikan lima variabel yang semuanya bertipe integer sebagai berikut:

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

Variabel individual dari sebuah larik dapat diacu dengan beberapa cara. Variabel dari larik ini juga disebut variabel berindeks (*indexed variable*) atau variabel bersubskrip (*subscripted variable*) atau elemen-elemen larik. Bilangan dalam kurung siku disebut indeks atau subskrip. Dalam C++, bilangan indeks dimulai dari 0, tidak dimulai dengan 1 atau bilangan lain selain 0. Jumlah dari variabel berindeks dalam sebuah larik disebut ukuran dari larik. Ketika sebuah larik dideklarasikan, ukuran dari larik dituliskan dalam kurung siku setelah nama larik. Indeks dari variabel individual dari sebuah larik dimulai dari 0 sampai dengan ukuran larik dikurangi 1.

Pada contoh di bagian terdahulu kita mengambil contoh larik dengan elemen-elemen bertipe integer, sebenarnya elemen-elemen larik dapat bertipe sembarang. Sebagai contoh, misalkan kita ingin mendeklarasikan sebuah larik dengan elemen-elemen bertipe double maka kita cukup mengetikkan double sebagai pengganti int.

Kita dapat mendeklarasikan larik bersama dengan pendeklarasian variabel regular. Sebagai contoh, pernyataan berikut ini mendeklarasikan dua variabel regular next dan max bersama dengan pendeklarasian sebuah larik bernama score:

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

Variabel berindeks seperti score[2] dapat digunakan dimanapun di dalam program seperti halnya variabel regular.

Janganlah bingung dengan dua kegunaan kurung siku bersama dengan nama larik. Ketika digunakan dalam pendeklarasian sebuah larik, seperti misalnya

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

bilangan di dalam kurung siku menunjukkan jumlah elemen dalam larik tersebut. Tetapi ketika digunakan pada bagian program lainnya, bilangan dalam kurung siku menunjukkan elemen tertentu dalam larik tersebut. Sebagai contoh, score[0] menunjukkan elemen pertama dalam larik score.

Bilangan indeks yang dituliskan dalam kurung siku tidaklah harus berupa konstanta integer. Kita dapat menuliskan sembarang ekspresi dalam kurung siku yang jika dievaluasi harus menghasilkan nilai bertipe integer dengan nilai terkecil 0 dan terbesar harus lebih kecil satu dari ukuran larik. Sebagai contoh, kode berikut akan menset score[3] sama dengan 99:

```
int n = 2;
score[n + 1] = 99;
```

Pada Contoh Program 6.1 berikut ini digunakan larik, memasukkan data ke dalam elemen-elemen larik dan memproses elemen-elemen larik dengan cara-cara seperti yang telah dibahas pada bagian terdahulu.

Contoh Program 6.1 Program yang Menggunakan Larik

```
//Reads in five scores and shows how much each
//score differs from the highest score.
```

```
#include <iostream>

int main( )
{
  int i, score[5], max;
  cout << "Enter 5 scores:\n";
  cin >> score[0];
  max = score[0];
  for (i = 1; i < 5; i++)
  {
    cin >> score[i];
    if (score[i] > max)
      max = score[i];
 //max is the largest of the values score[0],..., score[i].
  }
  cout << "The highest score is " << max << endl
         << "The scores and their\n"
         << "differences from the highest are:\n";
  for (i = 0; i < 5; i++)
     cout << score[i] << " off by "
           << (max - score[i]) << endl;
  return 0;
}
```

Contoh Output Program:

```
Enter 5 scores:
5 9 2 10 6
The highest score is 10
The scores and their
differences from the highest are:
5 off by 5
9 off by 1
2 off by 8
10 off by 0
6 off by 4
```

Inisialisasi Larik

Larik bisa diinisialisasi ketika dideklarasikan. Pada inisialisasi larik, nilai-nilai inisial untuk elemen-elemen larik ditempatkan pada kurung kurawal dan dipisahkan dengan koma. Sebagai contoh:

```
int value[3] = {4, 5, 6};
```

Deklarasi tersebut ekivalen dengan kode berikut:

```
int value[3];
```

```
value[0] = 4;
value[1] = 5;
value[2] = 6;
```

Apabila nilai inisial yang diberikan lebih sedikit jumlahnya dari elemen-elemen larik yang ada maka elemen-elemen larik sisanya akan diinisialisasi dengan 0. Tetapi, variabel regular dan larik yang tidak diinisialisasi yang dideklarasikan dalam definisi fungsi termasuk fungsi main tidak akan diinisialisasi oleh compiler.

Apabila kita menginisialisasi larik ketika larik tersebut dideklarasikan, kita bisa saja tidak menuliskan ukuran larik, dan secara otomatis larik akan dideklarasikan dengan ukuran minimum sesuai dengan jumlah nilai-nilai inisial yang ada. Sebagai contoh, deklarasi berikut:

```
int b[] = {5, 12, 11};
```

ekivalen dengan deklrasi

```
int b[3] = {5, 12, 11};
```

6.2 Larik pada Fungsi

Kita dapat menggunakan elemen dari sebuah larik atau sebuah larik secara keseluruhan sebagai argumen dari sebuah fungsi.

Elemen Larik sebagai Argumen Fungsi

Kita dapat menggunakan elemen dari sebuah larik sebagai argumen dari sebuah fungsi. Sebagai contoh, misalkan kita mempunyai program yang memuat deklarasi berikut:

```
double i, n, a[10];
```

Misalkan dalam program tersebut kita juga mendeklarasikan sebuah fungsi:

```
double function_1(double x);
```

maka pemanggilan fungsi:

```
function_1(a[3]);
```

adalah legal, karena argumen dari function_1 bertipe double dan a[3] juga bertipe double juga.

Larik sebagai Argumen Fungsi

Kita dapat menggunakan sebuah larik secara keseluruhan sebagai argumen dari sebuah fungsi. Sebagai contoh, perhatikan deklarasi dan definisi fungsi pada contoh 6.2 berikut:

Deklarasi fungsi:

```
void fillUp(int a[], int size);
//Precondition: size is the declared size of the array a.
//The user will type in size integers.
//Postcondition: The array a is filled with size integers
//from the keyboard.
```

Definisi fungsi:

```
void fillUp(int a[], int size)
{
  cout << "Enter " << size << " numbers:\n";
  for (int i = 0; i < size; i++)
  cin >> a[i];
  cout << "The last array index used is " << (size - 1) << endl;
}
```

Fungsi fillUp mempunyai sebuah parameter yang berupa larik dan sebuah parameter yang berupa variabel biasa yang bertipe integer.

Ketika fungsi fillUp dipanggil, maka pemanggilan ini harus melibatkan dua argumen yaitu sebuah argumen berupa larik dan sebuah argumen berupa variabel tunggal bertipe integer. Sebagai contoh, pemanggilan berikut merupakan pemanggilan yang legal terhadap fungsi fillUp:

```
int score[5], numberOfScores = 5;
fillUp(score, numberOfScores);
```

Perhatikan bahwa pada pemanggilan fungsi fillUp, argumen pertama yang berupa larik dituliskan tanpa kurung siku.

Apa yang terjadi terhadap argumen score apabila fungsi fillUp dipanggil?. Apabila fungsi fillUp dipanggil maka argumen score semua parameter a yang ada pada tubuh fungsi fillUp akan digantikan oleh score, kemudian kode yang ada pada tubuh fungsi ini akan dieksekusi. Oleh karena itu pemanggilan fungsi

```
fillUp(score, numberOfScores);
```

ekivalen dengan kode berikut:

```
{
size = 5;
cout << "Enter " << size << " numbers:\n";
for (int i = 0; i < size; i++)
cin >> score[i];
cout << "The last array index used is " << (size - 1) << endl;
}
```

Modifier Const

Ketika kita menggunakan argumen larik pada pemanggilan sebuah fungsi, maka fungsi tersebut akan merubah nilai-nilai yang tersimpan dalam larik tersebut. Apabila kita ingin pemanggilan fungsi tersebut tidak merubah nilai-nilai yang tersimpan dalam larik maka kita harus gunakan modifier const yang ditempatkan di depan nama larik. Parameter yang dimodifikasi dengan const disebut parameter larik konstan (*constant array parameter*). Sebagai contoh, fungsi showTheWorld berikut ini menggunakan modifier const pada argumen lariknya:

```
void showTheWorld(const int a[], int sizeOfa);
//Precondition: sizeOfa is the declared size of the array a.
//All indexed variables of a have been given values.
//Postcondition: The values in a have been written to the screen.
{
  cout << "The array contains the following values:\n";
  for (int i = 0; i < sizeOfa; i++)
  cout << a[i] << " ";
  cout << endl;
}
```

6.3 Pemrograman dengan Larik

Seringkali ukuran pasti dari larik yang dibutuhkan ketika kita menulis program tidak diketahui, atau ukuran larik yang dibutuhkan bervariasi dari satu eksekusi ke eksekusi yang lain. Satu cara yang umum dan mudah untuk menangani masalah ini adalah mendeklarasikan larik dengan ukuran terbesar yang mungkin dibutuhkan. Program bisa menggunakan seluruh atau sebagian dari elemen-elemen larik yang ada. Ada beberapa hal yang harus diperhatikan jika kita menggunakan larik secara sebagian, yaitu harus diingat elemen-elemen yang digunakan dan adanya kemungkinan pengacuan terhadap elemen larik yang belum terisi data atau kosong. Contoh Program 6.2 berikut ini diimplementasikan dengan memperhatikan hal-hal tersebut.

Contoh Program 6.2 Larik Terisi Sebagian (*Partially Filled Array*)

```
//Shows the difference between each of a list of golf scores and their average.
#include <iostream>

const int MAX_NUMBER_SCORES = 10;
void fillArray(int a[], int size, int& numberUsed);
//Precondition: size is the declared size of the array a.
//Postcondition: numberUsed is the number of values stored in a.
//a[0] through a[numberUsed-1] have been filled with
//nonnegative integers read from the keyboard.
```

```
double computeAverage(const int a[], int numberUsed);
//Precondition: a[0] through a[numberUsed-1] have values; numberUsed > 0.
//Returns the average of numbers a[0] through a[numberUsed-1].
void showDifference(const int a[], int numberUsed);
//Precondition: The first numberUsed indexed variables of a have values.
//Postcondition: Gives screen output showing how much each of the first
//numberUsed elements of the array a differs from their average.
int main( )
{
  int score[MAX_NUMBER_SCORES], numberUsed;
  cout << "This program reads golf scores and shows\n"
       << "how much each differs from the average.\n";
  cout << "Enter golf scores:\n";

fillArray(score, MAX_NUMBER_SCORES, numberUsed);
showDifference(score, numberUsed);
return 0;
}

void fillArray(int a[], int size, int& numberUsed)
{
  cout << "Enter up to " << size << " nonnegative whole numbers.\n"
       << "Mark the end of the list with a negative number.\n";
  int next, index = 0;
  cin >> next;
  while ((next >= 0) && (index < size))
  {
    a[index] = next;
    index++;
    cin >> next;
  }
  numberUsed = index;
}

double computeAverage(const int a[], int numberUsed)
{
  double total = 0;
  for (int index = 0; index < numberUsed; index++)
  total = total + a[index];
  if (numberUsed > 0)
  {
    return (total/numberUsed);
  }
  else
  {
     cout << "ERROR: number of elements is 0 in computeAverage.\n"
          << "computeAverage returns 0.\n";
     return 0;
  }
}
```

```
void showDifference(const int a[], int numberUsed)
{
  double average = computeAverage(a, numberUsed);
  cout << "Average of the " << numberUsed
         << " scores = " << average << endl
         << "The scores are:\n";

  for (int index = 0; index < numberUsed; index++)
   cout << a[index] << " differs from average by "
          << (a[index] - average) << endl;
}
```

Output Program:

```
This program reads golf scores and shows
how much each differs from the average.
Enter golf scores:
Enter up to 10 nonnegative whole numbers.
Mark the end of the list with a negative number.
69 74 68 -1
Average of the 3 scores = 70.3333
The scores are:
69 differs from average by -1.33333
74 differs from average by 3.66667
68 differs from average by -2.33333
```

6.4 Larik Multidimensi

Ada kalanya kita perlu menggunakan larik dengan lebih dari satu indeks, dan hal ini dimungkinkan dalam C++. Berikut ini kita mendeklarasikan larik dari karakter-karakter yang diberi nama page. Larik page mempunyai dua indeks: indeks pertama mempunyai jangkauan dari 0 sampai 29 dan indeks yang kedua mempunyai jangkauan dari 0 sampai 99.

```
char page[30][100];
```

Variabel berindeks/elemen dari larik ini masing-masing mempunyai dua indeks. Sebagai contoh, page[0][0], page[15][32], dan page[29][99] merupakan tiga elemen dari larik page. Perlu diperhatikan bahwa setiap indeks harus dituliskan dalam kurung siku. Setiap elemen dari larik multidimensi mempunyai tipe yang sama dengan tipe dasar dari larik tersebut yaitu char.

Sebuah larik bisa saja mempunyai lebih dari satu indeks, tetapi pada umumnya jumlah indeks larik yang sering digunakan adalah dua. Larik dua dimensi dapat direpresentasikan dalam bentuk tampilan dua dimensi dengan indeks pertama sebagai baris dan indeks kedua sebagai

kolom. Sebagai contoh, elemen-elemen dari larik dua dimensi page bisa disajikan dalam bentuk tampilan dua dimensi 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]
```

Kita dapat menggunakan larik page untuk menyimpan karakter-karakter pada sebuah halaman teks yang mempunyai tiga puluh baris (0 sampai dengan 29) dan seratus karakter pada setiap baris (0 sampai dengan 99).

Pada C++, larik dua dimensi seperti page, secara aktual merupakan larik dari larik. Larik page di atas secara aktual merupakan larik satu dimensi berukuran 30, dengan tipe dasar berupa larik satu dimensi dengan ukuran 100 yang bertipe character.

6.5 Parameter Larik Dua Dimensi

Deklarasi larik dua dimensi berikut ini secara aktual mendeklarasikan sebuah larik satu dimensi berukuran 30 dengan tipe dasar larik satu dimensi dari karakter dengan ukuran 100.

```
char page[30][100];
```

Menggambarkan sebuah larik dua dimensi sebagai sebuah larik dari larik akan membantu kita untuk memahami bagaimana C++ menangani parameter-parameter untuk larik mmultidimensi. Sebagai contoh, berikut ini adalah sebuah fungsi yang menggunakan larik yang serupa dengan page, dan menampilkannya ke layar monitor:

```
void displayPage(const char p[][100], int sizeDimension1)
{
  for (int index1 = 0; index1 < sizeDimension1; index1++)
  {//Printing one line:
    for (int index2 = 0; index2 < 100; index2++)
    cout << p[index1][index2];
    cout << endl;
  }
}
```

Ukuran dari dimensi pertama pada parameter larik dua dimensi tidak dicantumkan, sehingga kita harus menyetakan parameter bertipe integer untuk menentukan ukuran dari dimensi pertama tersebut. Ukuran dari dimensi kedua (ketiga dan seterusnya jika ada) bisa kita tuliskan setelah nama parameter larik.

Ketika kita memandang parameter larik dua dimensi sebagai larik dari larik, maka untuk sebuah parameter larik dua dimensi

const char p[][100]

ukuran dimensi pertama benar-benar merupakan indeks larik dan diperlakukan seperti indeks pada larik satu dimensi. Sementara itu, dimensi kedua merupakan bagian dari deskripsi tipe dasar yaitu larik dari karakter dengan ukuran 100.

Contoh Program 6.3 merupakan sebuah program yang menggunakan larik dua dimensi yang diberi nama grade untuk menyimpan data yang berupa nilai quiz dan mengolah serta kemudian menampilkannya. Indeks pertama dari larik digunakan untuk menunjukkan mahasiswa yang dimakud, dan indeks yang kedua menunjukkan quiz.

```
#include <iostream>
#include <iomanip>

const int NUMBER_STUDENTS = 4, NUMBER_QUIZZES = 3;
void computeStAve(const int grade[][NUMBER_QUIZZES], double stAve[]);
void computeQuizAve(const int grade[][NUMBER_QUIZZES], double quizAve[]);
void display(const int grade[][NUMBER_QUIZZES], const double stAve[], const double quizAve[]);

int main( )
{
  int grade[NUMBER_STUDENTS][NUMBER_QUIZZES];
  double stAve[NUMBER_STUDENTS];
  double quizAve[NUMBER_QUIZZES];

  <The code for filling the array grade goes here, but is not shown.>

  computeStAve(grade, stAve);
  computeQuizAve(grade, quizAve);
  display(grade, stAve, quizAve);
  return 0;
}

void computeStAve(const int grade[][NUMBER_QUIZZES], double stAve[])
{
  for (int stNum = 1; stNum <= NUMBER_STUDENTS; stNum++)
  {//Process one stNum:
    double sum = 0;
```

```
    for (int quizNum = 1; quizNum <= NUMBER_QUIZZES; quizNum++)
       sum = sum + grade[stNum-1][quizNum-1];
    //sum contains the sum of the quiz scores for student number stNum.
   stAve[stNum-1] = sum/NUMBER_QUIZZES;
   //Average for student stNum is the value of stAve[stNum-1]
  }
}

void computeQuizAve(const int grade[][NUMBER_QUIZZES], double quizAve[])
{
  for (int quizNum = 1; quizNum <= NUMBER_QUIZZES; quizNum++)
  {//Process one quiz (for all students):
    double sum = 0;
    for (int stNum = 1; stNum <= NUMBER_STUDENTS; stNum++)
       sum = sum + grade[stNum-1][quizNum-1];
    //sum contains the sum of all student scores on quiz number quizNum.
    quizAve[quizNum-1] = sum/NUMBER_STUDENTS;
    //Average for quiz quizNum is the value of quizAve[quizNum-1]
  }
}

void display(const int grade[][NUMBER_QUIZZES], const double stAve[], const double quizAve[])
{
  cout.setf(ios::fixed);
  cout.setf(ios::showpoint);
  cout.precision(1);
  cout << setw(10) << "Student"
         << setw(5) << "Ave"
         << setw(15) << "Quizzes\n";
  for (int stNum = 1; stNum <= NUMBER_STUDENTS; stNum++)
  {//Display for one stNum:
    cout << setw(10) << stNum
           << setw(5) << stAve[stNum-1] << " ";
    for (int quizNum = 1; quizNum <= NUMBER_QUIZZES; quizNum++)
    cout << setw(5) << grade[stNum-1][quizNum-1];
    cout << endl;
  }
  cout << "Quiz averages = ";
  for (int quizNum = 1; quizNum <= NUMBER_QUIZZES; quizNum++)
     cout << setw(5) << quizAve[quizNum-1];
  cout << endl;
}
```

Output Program:

```
Student     Ave          Quizzes
        1    10.0      10    10    10
        2     1.0       2     0     1
        3     7.7       8     6     9
        4     7.3       8     4    10
Quiz Average =     7.0   5.0   7.5
```