PEN203

Structured Program Development in C++

C++ How to Program
Deitel & Deitel

The while repetition statement

- Repetition structures are used whenever a number of actions to be repeated while a condition remains true.
- Example pseudocode:

```
While varible x is less than 10
        Add x to current some
```

- Example C++ code:

```
while(x<10)
        sum=sum+x;
```

Counter-Controlled Repetition

```
// Fig. 2.7: fig02_07.cpp
// Class average program with counter-controlled repetition.
#include <iostream>

using std::cout;
using std::cin;
using std::endl;

// function main begins program execution
int main()
{
   int total;       // sum of grades input by user
   int gradeCounter; // number of grade to be entered next
   int grade;       // grade value
   int average;     // average of grades

   // initialization phase
   total = 0;          // initialize total
   gradeCounter = 1;   // initialize loop counter

```

Counter-Controlled Repetition

```
   // processing phase
   while ( gradeCounter <= 10 ) {      // loop 10 times
      cout << "Enter grade: ";         // prompt for input
      cin >> grade;                    // read grade from user
      total = total + grade;           // add grade to total
      gradeCounter = gradeCounter + 1; // increment counter
   }

   // termination phase
   average = total / 10;               // integer division

   // display result
   cout << "Class average is " << average << endl;

   return 0;   // indicate program ended successfully

} // end function main
```

Counter-Controlled Repetition

```
Enter grade:  98
Enter grade:  76
Enter grade:  71
Enter grade:  87
Enter grade:  83
Enter grade:  90
Enter grade:  57
Enter grade:  79
Enter grade:  82
Enter grade:  94
Class average is 81
```

Algorithms (Top-down, stepwise refinement)

- Programs usually have three phases:
  - Initialization
  - Processing
  - Termination

Algorithms (Top-down, stepwise refinement)

- A class-averaging program (unknown number of students)

```
// Fig. 2.9: fig02_09.cpp
// Class average program with sentinel-controlled repetition.
#include <iostream>

using std::cout;
using std::cin;
using std::endl;
using std::fixed;

#include <iomanip>       // parameterized stream manipulators

using std::setprecision;  // sets numeric output precision

// function main begins program execution
int main()
{
   int total;        // sum of grades
   int gradeCounter;  // number of grades entered
   int grade;        // grade value

   double average;    // number with decimal point for average

   // initialization phase
   total = 0;        // initialize total
   gradeCounter = 0;  // initialize loop counter
```

Algorithms (Top-down, stepwise refinement)

- A class-averaging program (unknown number of students)

```

// processing phase
// get first grade from user
cout << "Enter grade, -1 to end: ";  // prompt for input
cin >> grade;                         // read grade from user

// loop until sentinel value read from user
while ( grade != -1 ) {
   total = total + grade;           // add grade to total
   gradeCounter = gradeCounter + 1;  // increment counter

   cout << "Enter grade, -1 to end: ";  // prompt for input
   cin >> grade;                         // read next grade

} // end while

// termination phase
// if user entered at least one grade ...
if ( gradeCounter != 0 ) {

   // calculate average of all grades entered
   average = static_cast< double >( total ) / gradeCounter;

```

Algorithms (Top-down, stepwise refinement)

- A class-averaging program (unknown number of students)

```
      // display average with two digits of precision
      cout << "Class average is " << setprecision( 2 )
         << fixed << average << endl;

   } // end if part of if/else

   else // if no grades were entered, output appropriate message
      cout << "No grades were entered" << endl;

   return 0;   // indicate program ended successfully

} // end function main
```

```
Enter grade, -1 to end: 75
Enter grade, -1 to end: 94
Enter grade, -1 to end: 97
Enter grade, -1 to end: 88
Enter grade, -1 to end: 70
Enter grade, -1 to end: 64
Enter grade, -1 to end: 83
Enter grade, -1 to end: 89
Enter grade, -1 to end: -1
Class average is 82.50
```

Algorithms (Top-down, stepwise refinement)

Nested Control Structures

```
// Fig. 2.11: fig02_11.cpp
// Analysis of examination results.
#include <iostream>

using std::cout;
using std::cin;
using std::endl;

// function main begins program execution
int main()
{
   // initialize variables in declarations
   int passes = 0;           // number of passes
   int failures = 0;         // number of failures
   int studentCounter = 1;   // student counter
   int result;               // one exam result

   // process 10 students using counter-controlled loop
   while ( studentCounter <= 10 ) {

      // prompt user for input and obtain value from user
      cout << "Enter result (1 = pass, 2 = fail): ";
      cin >> result;

```

Algorithms (Top-down, stepwise refinement)

Nested Control Structures

```
      // if result 1, increment passes; if/else nested in while
      if ( result == 1 )      // if/else nested in while
         passes = passes + 1;

      else  // if result not 1, increment failures
         failures = failures + 1;

      // increment studentCounter so loop eventually terminates
      studentCounter = studentCounter + 1;

   } // end while

   // termination phase; display number of passes and failures
   cout << "Passed " << passes << endl;
   cout << "Failed " << failures << endl;

   // if more than eight students passed, print "raise tuition"
   if ( passes > 8 )
      cout << "Raise tuition " << endl;

   return 0;   // successful termination

} // end function main
```

Algorithms (Top-down, stepwise refinement)

Nested Control Structures

```
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 2
Enter result (1 = pass, 2 = fail): 2
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 2
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 2
Passed 6
Failed 4

Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 2
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Enter result (1 = pass, 2 = fail): 1
Passed 9
Failed 1
Raise tuition
```

Assignment Operators

- Assignment operators can be used instead of assignment expressions:

  a=a+5 can be written as a+=5

  a=a-5 can be written as a-=5

  a=a*5 can be written as a/=5

  a=a/5 can be written as a+=5

  a=a%5 can be written as a%=5

Increment and Decrement Operators

- Increment operator (c++) can be used instead of c=c+1
- Decrement operator (c--) can be used instead of c=c-1
- c++ and c-- postincrement operators
  - Expression executes before the variable is changed
- ++c and --c postincrement operators
  - Variable is changed and then expression executes.

Increment and Decrement Operators

```
// Fig. 2.14: fig02_14.cpp
// Preincrementing and postincrementing.
#include <iostream>

using std::cout;
using std::endl;

// function main begins program execution
int main()
{
  int c;                  // declare variable

  // demonstrate postincrement
  c = 5;                  // assign 5 to c
  cout << c << endl;        // print 5
  cout << c++ << endl;      // print 5 then postincrement
  cout << c << endl << endl; // print 6

  // demonstrate preincrement
  c = 5;                  // assign 5 to c
  cout << c << endl;        // print 5
  cout << ++c << endl;      // preincrement then print 6
  cout << c << endl;        // print 6
```

Increment and Decrement Operators

```

   return 0;   // indicate successful termination

} // end function main
```

```
5
5
6

5
6
6
```