Programming Fundamentals- I

Rao Muhammad Umer
Lecturer,
Web: raoumer.github.io
Department of Computer Science & IT,
The University of Lahore.

Administrative Stuff

- Due Date for submission of Course Survey Feedback form:
  - **30/12/2016**

Problem Solving Strategies

Problem Solving Strategies

- Before writing a program:
  - Have a thorough understanding of the problem
  - Carefully plan an approach for solving it
- While writing a program:
  - Know what "building blocks" are available
  - Use good programming principles

Algorithms

- Computing problems
  - All can be solved by executing a series of actions in a specific order
- Algorithm: procedure in terms of
  - Actions to be executed
  - The order in which these actions are to be executed
- Program control
  - Specify order in which statements are to executed

Structured-Programming Summary

- Structured programming
  - Easier than unstructured programs to understand, test, debug and, modify programs
- Rules for structured programming
  - Rules developed by programming community
  - Only single-entry/single-exit control structures are used
  - Rules:
    1. Begin with the “simplest flowchart”
    2. Any rectangle (action) can be replaced by two rectangles (actions) in sequence
    3. Any rectangle (action) can be replaced by any control structure (sequence, **if, if/else, switch, while, do/while** or **for**)
    4. Rules 2 and 3 can be applied in any order and multiple times

Pseudo code

- Pseudo code
  - Artificial, informal language that helps us develop algorithms
  - Similar to everyday English
  - Not actually executed on computers
  - Helps us “think out” a program before writing it
    - Easy to convert into a corresponding C++ program
    - Consists only of executable statements

Formulating Algorithms

- Counter-controlled repetition
  - Loop repeated until counter reaches a certain value
  - Definite repetition: number of repetitions is known
  - Example: A class of ten students took a quiz. The grades (integers in the range 0 to 100) for this quiz are available to you. Determine the class average on the quiz
  - Pseudocode:

Set total to zero
Set grade counter to one
While grade counter is less than or equal to ten
Input the next grade
Add the grade into the total
Add one to the grade counter
Set the class average to the total divided by ten
Print the class average

/* Class average program with counter controlled repetition*/

```
#include<iostream>
using namespace std;
int main()
{
int counter;
float grade, total, average;
/* initialization phase */
total = 0.0;
counter = 1;
/* processing phase */
while( counter <= 10 )
{
cout << "Enter grade: " ;
cin >> grade ;
total = total + grade;
counter = counter + 1;
}
/* termination phase */
average = total / 10.0;
cout << "Class average is” <<
average;
return 0;  /* indicate program
              ended success */
}
```

Output

- 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

Formulating Algorithms with Top-Down, Stepwise Refinement

- Problem becomes:

  Develop a class-averaging program that will process an arbitrary number of grades each time the program is run.

  - **Unknown number of students**
  - How will the program know to end?
- **Use sentinel value**
  - **Also called signal value, dummy value, or flag value**
  - Indicates "end of data entry."
  - Loop ends when user inputs the sentinel value
  - Sentinel value chosen so it cannot be confused with a regular input (such as `-1` in this case)

Formulating Algorithms with Top-Down, Stepwise Refinement

- Top-down, stepwise refinement
  - Begin with a pseudocode representation of the *top*:

    Determine the class average for the quiz
  - Divide *top* into smaller tasks and list them in order:

    Initialize variables
    Input, sum and count the quiz grades
    Calculate and print the class average
- Many programs have **three phases**:
  - **Initialization:** initializes the program variables
  - **Processing:** inputs data values and adjusts program variables accordingly
  - **Termination:** calculates and prints the final results

Formulating Algorithms with Top-Down, Stepwise Refinement

- Refine the initialization phase from *Initialize variables* to:

  Initialize total to zero
  Initialize counter to zero

- Refine *Input, sum and count the quiz grades* to

  Input the first grade (possibly the sentinel)
  While the user has not as yet entered the sentinel
  Add this grade into the running total
  Add one to the grade counter
  Input the next grade (possibly the sentinel)

Formulating Algorithms with Top-Down, Stepwise Refinement

- Refine *Calculate and print the class average* to

```
If the counter is not equal to zero
    Set the average to the total divided by the counter
    Print the average
else
    Print "No grades were entered"
```

```
/* class average program with sentinel-controlled
repetition */
#include <iostream>
using namespace std;

int main ()
{
float average;
Int counter, grade, total;

/* initialization  phase */
total = 0;
counter = 0;
```

1. Initialize Variables

```
/* processing phase*/
cout << “Enter grade, -1 to end:”;
cin >> grade;

while (grade !=-1){
        total = total + grade;
        counter = counter + 1;
        cout << “ Enter grade, -1 to end:”;
        cin >> grade;

}
```

2. Get user input
3. Perform Loop

3. Calculate Average

4 Print Results

5 Program Output

```
/* termination phase */
if( counter != 0 ) {
        average = ( float) total / counter;
        cout << "Class average is“ << average;
        }
else
        cout << "No grades were entered\n" ;

return 0;  /* indicate program ended successfully */
}
```

- 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

To Find the Maximum Marks

```
#include <iostream>
using namespace std;
int main()
{
int marks[6] =
{36,78,7,99,43,29};
Maximum = marks[0];
int i;
for (i = 0; i <6;i++)
{
   if (Maximum <marks[i])
      {
           Maximum = marks[i];
       }
}
cout << “Maximum Marks
are\n” << Maximum;
getch();
return 0;
}
```

Sorting an Array

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

int main()
{
int s, i, j, temp, a[20];
cout << "Enter total elements: ";
cin >> s;
cout << “Enter “ << s << “elements: “;
for(i=0;i<s;i++)
      cin >> a[i];
for(i=0;i<s;i++)
{
     for(j=i+1; j<s; j++)
        {
            if(a[i]>a[j])
               {
                    temp=a[i];
                    a[i]=a[j];
                    a[j]=temp;
                }
         }
}
cout << "After sorting: \n";
for(i=0; i<s; i++)
       cont << a[i];
getch();
return 0;
}
```