

## Evaluating a Formula

- **Problem (The Registrar's Headache):**

Student's are to be allowed to register for a certain course offered by an outstanding Professor based on their grade point average (gpa) and a complicated formula given below.

Write a C++ program to produce a table for the value produced by the formula for gpa=0.00 up to gpa=4.00, increasing gpa by 0.50 each time.

If the value of the formula is greater than or equal to zero the student is to be admitted into the class. Otherwise, the student can not register for the class.

- **Formula:**

$$result = \frac{gpa^3 + 7gpa - 1}{gpa^2 - \frac{gpa + 5}{3}}$$

- **Program:**

```
/* program prob2
 * create a table to evaluate a formula result = f(gpa)
 */
#include <iostream>
using namespace std;
int main()
{
    declare gpa and result as real variables

    action part of program

    return 0;
}
```

## Data Types for Real Numbers

- Data Type **float**:  
"float" creates a single precision real variable.

```
float gpa;
```

- Data type **double**:  
"double" creates a double precision real variable. Double precision numbers can represent larger and smaller numbers than single precision numbers and with more precision (more bits are used to store the number).

```
double gpa;
```

- **Return to Our Problem:**

```
/* program prob2
 * create a table to evaluate a formula result = f(gpa)
 */
#include <iostream>
using namespace std;
int main()
{
    double gpa,result;

    action part of program

    return 0;
}
```

## Setting Up the for loop

- **Another Revision:**

```
/* program prob2
 * create a table to evaluate a formula result = f(gpa)
 */
#include <iostream>
using namespace std;
int main()
{
    double gpa,result;

    for (gpa = 0.00; gpa <= 4.00; gpa = gpa + 0.50)
    {
        compute result = the formula applied to gpa
        print gpa and result
        if result is >= 0 then print "Admit"
    }
    return 0;
}
```

## Writing the Formula in C + +

- **Formula:**

$$result = \frac{gpa^3 + 7gpa - 1}{gpa^2 - \frac{gpa + 5}{3}}$$

- **C + + Statement:**

result = gpa \* gpa \* gpa + 7 \* gpa - 1 / gpa \* gpa - gpa + 5 / 3; {wrong}

- **Order (Precedence) of Operations:**

a) Unary Minus, Unary Plus, + +, --

b) \*, /, %

c) +, -

- **Associativity of Operations:**

Associativity determines which of two operators has priority given that they have the same precedence.

a) For binary operators, associativity is left to right.

b) for unary operators, associativity is right to left.

e.g., y = -x + +;

- **Example:**

Let a = 10; b = 2

$$y = \frac{a}{b + 3}$$

y = a / b + 3; {y = 8 wrong}

y = a / (b + 3); {y = 2 correct}

- **Our Formula:**

result = (gpa \* gpa \* gpa + 7 \* gpa - 1) / (gpa \* gpa - (gpa + 5) / 3);

- **Note:**

When in doubt, use parentheses!

## Printing Real Numbers

- **Another Revision:**

```
/* program prob2
 * create a table to evaluate a formula result = f(gpa)
 */
#include <iostream>
using namespace std;
int main()
{
    double gpa,result;

    for (gpa = 0.00; gpa <= 4.00; gpa = gpa + 0.50)
    {
        result = (gpa*gpa*gpa + 7*gpa-1)/(gpa*gpa-(gpa+5)/3);
        cout << "gpa = " << gpa << " result = " <<
            result << endl;
        if result is >= 0 then print "Admit"
    }
    return 0;
}
```

- **Note:**

C++ will print only the decimal places necessary to represent the value of the number accurately.

e.g.,

2.0 will print as 2

12.600 will print as 12.6

By default, C++ prints real numbers with up to six significant digits. The number of digits to the right of the decimal point depends on how many digits are left over after printing the integer portion of the number.

e.g.,

2143.67645 by default will print as 2143.68

12.3456789 by default will print as 12.3457

## The if (Conditional) Statement

- **The if (Conditional) Statement:**

```
if (condition)  
    simple_statement
```

- **Compound Statement Form:**

```
if (condition)  
{  
    compound_statement  
}
```

- **Yet Another Revision:**

```
/* program prob2  
 * create a table to evaluate a formula result = f(gpa)  
 */  
#include <iostream>  
using namespace std;  
int main()  
{  
    double gpa,result;  
  
    for (gpa = 0.00; gpa <= 4.00; gpa = gpa + 0.50)  
    {  
        result = (gpa*gpa*gpa + 7*gpa-1)/(gpa*gpa-(gpa+5)/3);  
        cout << "gpa = " << gpa << " result = " <<  
            result << endl;  
        if (result >= 0)  
            cout << " Admit" << endl;  
    }  
    cout << "The table is finished" << endl;  
    return 0;  
}
```

## The Program So Far

- **Program:**

```
/* program prob2
 * create a table to evaluate a formula result = f(gpa)
 */
#include <iostream>
using namespace std;
int main()
{
    double gpa,result;

    for (gpa = 0.00; gpa <= 4.00; gpa = gpa + 0.50)
    {
        result = (gpa*gpa*gpa + 7*gpa-1)/(gpa*gpa-(gpa+5)/3);
        cout << "gpa = " << gpa << " result = " <<
            result << endl;
        if (result >= 0)
            cout << " Admit" << endl;
    }
    cout << "The table is finished" << endl;
    return 0;
}
```

- **Output:**

```
gpa = 0 result = 0.6
Admit
gpa = 0.5 result = -1.65789
gpa = 1 result = -7
gpa = 1.5 result = 154.5
Admit
gpa = 2 result = 12.6
Admit
gpa = 2.5 result = 8.56667
Admit
gpa = 3 result = 7.42105
Admit
gpa = 3.5 result = 7.04867
Admit
gpa = 4 result = 7
Admit
```

## Creating a Readable Table

- **Table Header:**  
`cout << "Table of Function Values" << endl;`
- **Double Spacing:**  
`cout << "Table of Function Values" << endl << endl;`
- **Staying on the Same Line:**  
`cout << "gpa = " << gpa << " result = " << result;`
- **Starting a New Line:**  
`cout << endl;`
- **Printing a Table Trailer:**  
`cout << endl << "The table is finished" << endl;`
- **Column Headings:**  
`cout << "Grade Point Average    Value of Formula    Status"  
    << endl;`

## The Complete Program:

```
/* program prob2
 * create a table to evaluate a formula result = f(gpa)
 */
#include <iostream>
using namespace std;
int main()
{
    double gpa,result;

    cout << "Table of Function Values " << endl << endl;
    cout << "Grade Point Average   Value of Formula   Status"
         << endl;
    for (gpa = 0.00; gpa <= 4.00; gpa = gpa + 0.50)
    {
        result = (gpa*gpa*gpa + 7*gpa-1)/(gpa*gpa-(gpa+5)/3);
        cout << "      " << gpa << "      " << result;
        if (result >= 0)
            cout << "                Admit";
        cout << endl;
    }
    cout << endl << "The table is finished" << endl;
    return 0;
}
```

### ● Output:

Table of Function Values

Grade Point Average	Value of Formula	Status
0	0.6	Admit
0.5	-1.65789	
1	-7	
1.5	154.5	Admit
2	12.6	Admit
2.5	8.56667	Admit
3	7.42105	Admit
3.5	7.04867	Admit
4	7	Admit

The table is finished!

## Aligning Columns

- **The TAB Character "\t":**

Inserting the tab character "\t" into the format string causes the cursor to tab to the next tab position before printing.

- **Final Version:**

```
/* program prob2
 * create a table to evaluate a formula result = f(gpa)
 */
#include <iostream>
using namespace std;
int main()
{
    double gpa,result;

    cout << "\t\t\tTable of Function Values" << endl << endl;
    cout << "Grade Point Average\tValue of Formula\tStatus"
         << endl;
    for (gpa = 0.00; gpa <= 4.00; gpa = gpa + 0.50)
    {
        result = (gpa*gpa*gpa + 7*gpa-1)/(gpa*gpa-(gpa + 5)/3);
        cout << gpa << "\t\t" << result;
        if (result >= 0)
            cout << "\t\tAdmit";
        cout << endl;
    }
    cout << endl << "The table is finished" << endl;
    return 0;
}
```

- **Output:**

Table of Function Values

Grade Point Average	Value of Formula	Status
0	0.6	Admit
0.5	-1.65789	
1	-7	
1.5	154.5	Admit
2	12.6	Admit
2.5	8.56667	Admit
3	7.42105	Admit
3.5	7.04867	Admit
4	7	Admit

The table is finished!

## Escape Sequences

- In C + + , the computer uses the \ (backslash) symbol as an **escape character**. It tells the computer to treat the next character as special.
- The combination of the \ (backslash) followed by a character is known as an **escape sequence**.
- **Common Escape Sequences in C + + :**
  - \n - the newline character (linefeed)
  - \t - the tab character
  - \b - the backspace character
  - \\" - the double quote character
  - \\ - the backslash character
  - \% - the percent sign character (some compilers use %%)
  - \0 - the null character
  - \a - BEL (makes an audible signal)
  - \f - formfeed (new page on printer)
  - \r - carriage return (return to beginning of line)
- **Example:**

```
cout << "\% \\ \"\n";  
  
% \ "
```

## Compound Assignment Operators

- `+=`, `-=`, `*=`, `/=`, `%=`

`gpa = gpa + 0.50;` can be written as `gpa += 0.50;`

`gpa = gpa - 0.50;` can be written as `gpa -= 0.50;`

`gpa = gpa * 0.50;` can be written as `gpa *= 0.50;`

`gpa = gpa / 0.50;` can be written as `gpa /= 0.50;`

`num = num % 7;` can be written as `num %= 7;`

- Note:

`number = number + 1;` is equivalent to

`number += 1;` is equivalent to

`number ++;` is equivalent to

`++number;`

## Precedence of Arithmetic, Assignment, & Relational Operators

<u>Precedence</u>	<u>Associativity</u>
a) Unary Minus, Unary Plus, ++, --	right to left
b) *, /, %	left to right
c) +, -	left to right
d) <, <=, >, >=	left to right
e) ==, !=	left to right
f) =, +=, -=, *=, /=, %=	right to left

### ● Examples:

```
w = x = y = z = 25;
```

```
b = 10;
```

```
a = b + = 5;
```

```
if (x * y + z == x / z - y)
```

```
    cout << "They are equal" << endl;
```

## Mixed Mode Arithmetic

- An arithmetic expression that applies an operator to two operands of the same type produces a result of that type.
- In C++, it is possible to perform arithmetic operations on operands of different types. This is called **mixed mode arithmetic**.
- When an operator is applied to two operands of different types, C++ uses its rule of automatic type conversion to convert the value of the more restrictive type to that of the less restrictive type (e.g., "int" is more restrictive than "double" or "float".)

- Example

```
double real_num, real_sum;  
int int_num, int_sum;
```

```
int_num = 5;  
real_num = 5.6;  
real_sum = int_num + real_num;  
int_sum = int_num + real_num;  
cout << "int_sum = " << int_sum << " and real_sum = "  
    << real_sum << endl;
```

- Output:

```
int_sum = 10 and real_sum = 10.6
```

- Notes:

- When a real number is assigned to an integer, the fractional part is truncated.

- For real division, one or both of the operands must be real:

```
real_num = int_num/2;           // wrong  
real_num = (double)int_num/2;  // correct (type casting)
```

## Standard (Predeclared) Library Functions:

- `sqrt(x)`  
`real_num = sqrt(64); // real_num = 8.0`
- `abs(integer)`  
`int_num = abs(25); // int_num = 25`  
`int_num = abs(-25); // int_num = 25`
- `fabs(realnum)`  
`real_num = fabs(-2.72); // real_num = 2.72`
- `ceil(realnum)`  
`int_num = ceil(3.54); // int_num = 4`
- `floor(realnum)`  
`int_num = floor(3.54); // int_num = 3`  
(Note: same as `int_num = 3.54`)
- Note: to round a real number you can write  
`int_num = real_num + 0.5;`
- Note: to use the above library functions, you must include the following directive:  
  
`#include <cmath>`

## Data Type char

- A variable of data type **char** stores a single character.

- Example  
char letter;

```
letter = 'a';           // note the single quotes
```

- Example:

```
if (letter != 'b')  
    cout << "letter has the value " << letter << endl;
```

- Example

```
letter = 'a';           // note the single quotes  
letter ++;             // letter == 'b'  
cout << "letter has the value " << letter << endl;  
letter = letter + 3;    // letter == 'e'  
cout << "letter has the value " << letter << endl;
```

- Example

```
letter = 67;           // letter == 'C'  
cout << "letter has the value " << letter << endl;
```

- Example

```
letter = '7';          // letter equals the character 7  
                        // not the number 7  
cout << "letter has the value " << letter << endl;
```

## Debugging

- **Types of Errors:**

- **Compilation Errors (Syntax Errors)**

```
coutit << "hello"; // misspelled keyword
x := x + 3;        // wrong assignment symbol
a = (b + 1/3;      // missing right parenthesis
x + 3 = 5;        // expression on left side of assignment
cout << "error"   // missing semicolon
```

- **Execution Errors:**

An error detected once the program is running

a) uninitialized variables used in assignment statements

```
x = x + 5; // where x was never initialized
```

b) divide by zero

```
y = (x + 25)/0;
```

- **Logical Errors:**

Normally, the hardest type of error to catch.

The program may run without any execution errors, but the result is wrong!

- **Find the Compilation Errors:**

```
#include <iostuff>
use namespace std;
integer main()
{
    doubles x,y;
    cout<<"Grade Point Average\tValue of Formula\tStatus\n";
    four (x = 0.00; x <= 4.00; x = x + 0.50)
    {
        y = x * x * x + 7 * x - 1) / (x * x - (x + 5) / 3)
        cout << "\t" << X << "\t" << Y;
        if (y >= 0) cout << "\t\tAdmit";
        cout << endl
    }
    returnit 0;
}
```

## Standard Input/Output Streams

- C + + has three standard I/O streams:

- **cin**

The standard input stream (cin) is associated with the **keyboard**. Unless specified otherwise, all input to your program comes from the keyboard. cin can be redirected to get input for another source.

- **cout**

The standard output stream (cout) is associated with the **monitor**. Unless specified otherwise, all output from your program goes to the monitor. cout can be redirected to send output to another destination.

- **cerr**

The standard error stream (cerr) is associated with the **monitor**.

## Files

- A **file** is a collection of data (usually stored on a disk.)
- An **input file** contains items for input to the program (i.e., items you might otherwise type in from the keyboard during interactive data entry).
- An **output file** contains items output from your program (i.e., items you might otherwise have sent to the screen).
- In C++ we access a file by declaring a stream and attaching it to a particular file.
- You must include the fstream library  
`#include <fstream>`

- **Examples: (simple) declaring and opening a file for output**

```
ofstream outfile("c:\\myoutput.txt");  
  
ofstream myoutput("c:\\chapter2\\myoutput.txt");  
  
ofstream cout("c:\\mypgms\\hw2.txt"); //this redirects cout  
  
ofstream myout("con"); //output to console
```

- **Examples: (simple) declaring and opening a file for input**

```
ifstream infile("c:\\myinput.txt");  
  
ifstream cin("c:\\mydata\\hw2input.txt");
```

- **Closing files:**

If you open a file, you should close it:

```
outfile.close();  
infile.close();
```

● **Example: redirecting cout to a file**

```
/* program prob2
 * create a table to evaluate a formula result = f(gpa)
 */
#include <iostream>
#include <fstream>
using namespace std;

int main()
{
    double gpa,result;

    //un-comment to redirect cout to a file
// ofstream cout("c:\\mypgms\\prob2.txt");

    cout << "\t\t\tTable of Function Values" << endl << endl;
    cout << "Grade Point Average\tValue of Formula\tStatus"
        << endl;
    for (gpa = 0.00; gpa <= 4.00; gpa = gpa + 0.50)
    {
        result = (gpa*gpa*gpa + 7*gpa-1)/(gpa*gpa-(gpa + 5)/3);
        cout << gpa << "\t\t\t" << result;
        if (result >= 0)
            cout << "\t\t\tAdmit";
        cout << endl;
    }
    cout << endl << "The table is finished" << endl;

    //un-comment when redirecting cout to a file
// cout.close(); //close output file
    return 0;
}
```

● **Example: sending program output to a file**

```
/* program prob2
 * create a table to evaluate a formula result = f(gpa)
 */
#include <iostream>
#include <fstream>
using namespace std;

int main()
{
    double gpa,result;

    // declare and open output file
    ofstream outfile("c:\\mypgms\\prob2.txt");
//    ofstream outfile("con");    //un-comment for debugging

    outfile << "\t\t\tTable of Function Values" << endl << endl;
    outfile << "Grade Point Average\tValue of Formula\tStatus"
        << endl;
    for (gpa = 0.00; gpa <= 4.00; gpa = gpa + 0.50)
    {
        result = (gpa * gpa * gpa + 7 * gpa - 1) / (gpa * gpa - (gpa + 5) / 3);
        outfile << gpa << "\t\t\t" << result;
        if (result >= 0)
            outfile << "\t\t\tAdmit";
        outfile << endl;
    }
    outfile << endl << "The table is finished" << endl;

    outfile.close();    //close output file
    return 0;
}
```

● **Example: sending program output to a file - separate .open()**

```
/* program prob2
 * create a table to evaluate a formula result = f(gpa)
 */
#include <iostream>
#include <fstream>
using namespace std;

int main()
{
    double gpa,result;
    ofstream outfile;          //declare an output file object

    // open output file
    outfile.open("c:\\mypgms\\prob2.txt");
// outfile.open("con");      //un-comment for debugging

    outfile << "\t\t\tTable of Function Values" << endl << endl;
    outfile << "Grade Point Average\tValue of Formula\tStatus"
        << endl;
    for (gpa = 0.00; gpa <= 4.00; gpa = gpa + 0.50)
    {
        result = (gpa * gpa * gpa + 7 * gpa - 1) / (gpa * gpa - (gpa + 5) / 3);
        outfile << gpa << "\t\t\t" << result;
        if (result >= 0)
            outfile << "\t\t\tAdmit";
        outfile << endl;
    }
    outfile << endl << "The table is finished" << endl;

    outfile.close();          //close output file
    return 0;
}
```

● **Example: redirecting cout to a file - separate .open()**

```
/* program prob2
 * create a table to evaluate a formula result = f(gpa)
 */
#include <iostream>
#include <fstream>
using namespace std;

int main()
{
    double gpa,result;
    ofstream cout;          //declare cout as an ofstream object

    // open output file
    cout.open("c:\\mypgms\\prob2.txt");
// cout.open("con");      //un-comment for debugging

    cout << "\t\t\tTable of Function Values" << endl << endl;
    cout << "Grade Point Average\tValue of Formula\tStatus"
        << endl;
    for (gpa = 0.00; gpa <= 4.00; gpa = gpa + 0.50)
    {
        result = (gpa * gpa * gpa + 7 * gpa - 1) / (gpa * gpa - (gpa + 5) / 3);
        cout << gpa << "\t\t\t" << result;
        if (result >= 0)
            cout << "\t\t\tAdmit";
        cout << endl;
    }
    cout << endl << "The table is finished" << endl;

    cout.close();          //close output file
    return 0;
}
```