

Programming with C++

COMP2011: Examples on C++ Basics and Controls

Cecia Chan
Brian Mak
Dimitris Papadopoulos
Pedro Sander
Charles Zhang

Department of Computer Science & Engineering
The Hong Kong University of Science and Technology
Hong Kong SAR, China



Part I

Guess The Number



- The game program picks a random number in the range of 1 to 100.
- Two players take turns to guess the number.
- After each guess, the program should tell the player if the number is correct, larger than or smaller than their guessed number.
- Whoever first guesses correctly wins the game.

Typical Output

Player 1, please enter your guess:

15

Sorry, the number is smaller than 15

Player 2, please enter your guess:

9

Sorry, the number is bigger than 9

Player 1, please enter your guess:

10

Player 1, you win!!!

Program Requirement

- Validate that a guessed number is in the range set by the program.
 - request a player to enter again until the input is valid.
- Determine if a guess is correct.
- Give suitable feedback to the players.
- Keep running until a guess is correct.

First Attempt: 1 Player and 1 Round

```
1  #include <iostream>      /* File: guess1.cpp */
2  using namespace std;
3
4  int main()                // 1st attempt: 1 player, 1 round
5  {
6      int number = 10;      // The answer is fixed beforehand
7      int guess;
8
9      cout << "Player 1, please enter your guess:" << endl;
10     cin >> guess;
11
12     if (guess == number)
13         cout << "Player 1, you win!!!" << endl;
14
15     else if (guess < number)
16         cout << "Sorry, the number is bigger than " << guess << endl;
17
18     else
19         cout << "Sorry, the number is smaller than " << guess << endl;
20
21     return 0;
22 }
```

Second Attempt: 1 Player and Multiple Rounds

```
1  #include <iostream>      /* File: guess2.cpp */
2  using namespace std;
3
4  int main()               // 2nd attempt: 1 player, multiple rounds
5  {
6      int number = 10;     // The answer is fixed beforehand
7      int guess;
8
9      do                   // Add a loop to implement multiple rounds
10     {
11         cout << "Player 1, please enter your guess:" << endl;
12         cin >> guess;
13
14         if (guess == number)
15             cout << "Player 1, you win!!!" << endl;
16
17         else if (guess < number)
18             cout << "Sorry, the number is bigger than "
19                 << guess << endl;
20         else
21             cout << "Sorry, the number is smaller than "
22                 << guess << endl;
23     } while (guess != number);
24
25     return 0;
26 }
```

Third Attempt: 1 Player, Multiple Rounds, Fixed Range

```
1  #include <iostream>      /* File: guess3.cpp */
2  using namespace std;
3
4  int main()      // 3rd attempt: 1 player, multiple rounds, fixed range
5  {
6      int number = 10;      // The answer is fixed beforehand
7      int guess;
8      int low = 1, high = 100; // Add 2 variables to record the range
9
10     do // Add a loop to implement multiple rounds
11     {
12         cout << "Player 1, please enter your guess:" << endl;
13         cin >> guess;
14
15         while (guess < low || guess > high) // Input validation loop
16         {
17             cout << "Invalid input, please enter a number between "
18                 << low << " and " << high << endl;
19             cin >> guess;
20         } // Can this loop be replaced with do-while?
21
22         if (guess == number)
23             cout << "Player 1, you win!!!" << endl;
24
25         else if (guess < number)
26         {
```


Third Attempt: 1 Player, Multiple Rounds, Fixed Range ..

```
27         cout << "Sorry, the number is bigger than "
28             << guess << endl;
29         low = guess + 1; // Update the lower bound of the range
30     }
31     else
32     {
33         cout << "Sorry, the number is smaller than "
34             << guess << endl;
35         high = guess - 1; // Update the upper bound of the range
36     }
37 } while (guess != number);
38
39 return 0;
40 }
```

Final Code with a Randomly Generated Guess Number

```
1  #include <iostream>      /* File: guess_number.cpp */
2  #include <cstdlib>       // Needed for calling the rand() function
3  #include <time.h>        // May need for calling the time() function
4  using namespace std;
5
6  int main()    // 2 players, multiple rounds, fixed range, random number
7  {
8      /* Random number generation RNG */
9      // time(0) number of seconds since 1970/01/01 00:00:00
10     // rand() returns an int 0 to 2^31-1
11     srand(time(0));      // (1) Seed the RNG
12     int number = rand() % 100 + 1; // (2) a random no. in [1..100]
13
14     int guess;
15     int low = 1, high = 100;
16     int player = 1;      // Set Player 1 as the current player
17
18     cout << "The generated number is: " << number << endl;
19     do
20     {
21         cout << "Player " << player
22             << ", please enter your guess: " << endl;
23         cin >> guess;
24
25         while (guess < low || guess > high) // Input validation loop
26         {
```

Final Code with a Randomly Generated Guess Number ..

```
27         cout << "Invalid input, please enter a number between "
28             << low << " and " << high << endl;
29         cin >> guess;
30     }
31
32     if (guess == number)
33         cout << "Player " << player << ", you win!!!" << endl;
34     else if (guess < number)
35     {
36         cout << "Sorry, the number is bigger than "
37             << guess << endl;
38         low = guess + 1; // Update the lower bound of the range
39     }
40     else
41     {
42         cout << "Sorry, the number is smaller than "
43             << guess << endl;
44         high = guess - 1; // Update the upper bound of the range
45     }
46
47     player = (player % 2) + 1; // This makes 1 → 2 and 2 → 1
48 } while (guess != number);
49
50 return 0;
51 }
```

Part II

Draw an Isosceles Right-Angled Triangles (RATs)



Draw Triangles

- Design a program that prints some isosceles right-angled triangles (RAT), and allows users to set their size.

- A RAT that has a size of 4 looks like this:

```
*  
**  
***  
****
```

- Furthermore, try the following variations:

Fat RAT	Hollow RAT	Upside-down RAT
<pre>* *** ***** ***** *****</pre>	<pre>* ** * * * * *****</pre>	<pre>***** ***** *** ** *</pre>

A Single RAT

```
1  #include <iostream>      /* File: one-rat.cpp */
2  using namespace std;
3
4  int main()
5  {
6      cout << "Size of a RAT: " << endl;
7      int size;              // height = width = size
8      cin >> size;
9
10     for (int width = 1; width <= size; width++) // #iters=height
11     {
12         // Draw one row of a RAT
13         for (int j = 0; j < width ; j++)        // width of a row
14             cout << '*';
15
16         cout << endl;
17     }
18
19     return 0;
20 }
```

Various RATs

```
1  #include <iostream>          /* File: various-rats.cpp */
2  using namespace std;
3
4  int main()
5  {
6      cout << "Size of a RAT: " << endl;
7      int size;                // height = width = size
8      cin >> size;
9
10     cout << "A fat RAT" << endl;
11     for (int i = 1; i <= size; i++)    // #iterations = height
12     {
13         for (int j = 0; j < i*2 - 1 ; j++) // width of a row
14             cout << '*';
15         cout << endl;
16     }
17
18     cout << "A hollow RAT" << endl;
19     for (int i = 1; i <= size; i++)
20     {
21         for (int j = 0; j < i ; j++)
22             cout << ((j == 0 || j == i - 1 || i == size) ? '*' : ' ');
23         cout << endl;
24     }
25
26
```

Various RATs ..

```
27     cout << "An upside-down RAT" << endl;
28     for (int i = size; i >= 1; i--)
29     {
30         for (int j = 0; j < i; j++)
31             cout << '*';
32         cout << endl;
33     }
34
35     return 0;
36 }
```


A Bug in a RAT: What's Wrong?

```
1  #include <iostream>      /* File: one-bad-rat.cpp */
2  using namespace std;
3
4  int main()
5  {
6      cout << "Size of a RAT: " << endl;
7      int size;
8      cin >> size;
9
10     cout << "A simple RAT:" << endl;
11     for (int i = 0; i < size; i++)
12     {
13         for (int j = 0; j <= i; j++)
14             cout << '*';
15         cout << endl;
16     }
17
18     cout << "Is this a RAT?" << endl;
19     for (int i = 1; i <= size; i++)
20         for (int j = 0; j < i * 2 - 1 ; j++)
21         {
22             cout << '*';
23             cout << endl;
24         }
25
26     return 0;
27 }
```

A Row of RATs

Now try this:

```
*      *      *      *      *      *      *      *      *      *
**     **     **     **     **     **     **     **     **     **
***    ***    ***    ***    ***    ***    ***    ***    ***    ***
***** ***** ***** ***** ***** ***** ***** ***** *****
*****
```

You'll need to measure the width of your screen first.

Bugs in RATs: What's Wrong?

```
1  #include <iostream>          /* File: row-of-bad-rats1.cpp */
2  using namespace std;
3
4  int main()
5  {
6      cout << "Size of a triangle: " << endl;
7      int size;
8      cin >> size;
9
10     // Find out the number of RATs in a row
11     const int TOTAL_NUM_COLUMNS = 105; // Assumed screen width
12     int num_RATs = TOTAL_NUM_COLUMNS / size;
13
14     for (int i = 1; i <= size; i++)
15     {
16         for (int n = 0; n < num_RATs; n++)
17             for (int j = 0; j < i ; j++)
18                 cout << '*';
19
20         cout << endl;
21     }
22
23     return 0;
24 }
```

Bugs in RATs Again: What's Wrong?

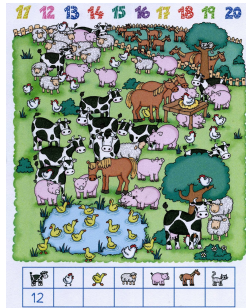
```
1  #include <iostream>          /* File: row-of-bad-rats2.cpp */
2  using namespace std;
3
4  int main()
5  {
6      cout << "Size of a triangle: " << endl;
7      int size;
8      cin >> size;
9
10     const int TOTAL_NUM_COLUMNS = 50; // Assumed screen width
11     int num_RATs = TOTAL_NUM_COLUMNS / size;
12
13     for (int i = 1; i <= size; i++)
14         for (int n = 0; n < num_RATs; n++)
15             {
16                 for (int j = 0; j < i ; j++)
17                     cout << '*';
18                 for (int j = 0; j < (size-i); j++)
19                     cout << ' '; // Print enough spaces after a RAT
20
21                 cout << endl;
22             }
23
24     return 0;
25 }
```

A Row of Good RATs

```
1  #include <iostream>                                /* File: row-of-rats.cpp */
2  using namespace std;
3
4  int main()
5  {
6      cout << "Size of a triangle: " << endl;
7      int size;
8      cin >> size;
9
10     // Find out the number of RATs in a row
11     const int TOTAL_NUM_COLUMNS = 105; // Assumed screen width
12     int num_RATs = TOTAL_NUM_COLUMNS / size;
13
14     for (int i = 1; i <= size; i++)
15     {
16         for (int n = 0; n < num_RATs; n++)
17         {
18             for (int j = 0; j < i ; j++)
19                 cout << '*';
20             for (int j = 0; j < (size-i); j++)
21                 cout << ' '; // Print enough spaces after each RAT
22         }
23         cout << endl;
24     }
25
26     return 0;
27 }
```

Part III

Count Animals



The Count Animals Problem

- There are two types of animals, pigs and sheeps in a farm.
- Each pig weighs 4.5 units and each sheep weighs 3 units.
- The total weight of animals in a barn should be exactly 36 units.
- List out all possible combinations of pigs and sheeps in the farm.

Solution:

$$0 * 4.5 + 12 * 3 = 36$$

$$2 * 4.5 + 9 * 3 = 36$$

$$4 * 4.5 + 6 * 3 = 36$$

$$6 * 4.5 + 3 * 3 = 36$$

$$8 * 4.5 + 0 * 3 = 36$$

First Attempt: What's Wrong?

```
1  #include <iostream>      /* File: two_animals-v1.cpp */
2  using namespace std;
3
4  int main()
5  {
6      float wt_pig = 4.5;
7      float wt_sheep = 3.0;
8      float total_wt = 36;
9
10     for (int num_pigs = 0; num_pigs * wt_pig <= total_wt;
11         ↪   num_pigs++)
12     {
13         float remain_wt = total_wt - num_pigs * wt_pig;
14         int num_sheeps = remain_wt / wt_sheep;
15         remain_wt = remain_wt % wt_sheep;
16
17         if (remain_wt == 0)
18             cout << num_pigs << " * " << wt_pig << " + "
19                 << num_sheeps << " * " << wt_sheep << " = "
20                 << total_wt << endl;
21     }
22     return 0;
23 }
```


Second Attempt: Any Problem?

```
1  #include <iostream>      /* File: two_animals-v2.cpp */
2  using namespace std;
3
4  int main()
5  {
6      float wt_pig = 4.5;
7      float wt_sheep = 3.0;
8      float total_wt = 36;
9
10     for (int num_pigs = 0; num_pigs * wt_pig <= total_wt;
11         ↪ num_pigs++)
12     {
13         float remain_wt = total_wt - num_pigs * wt_pig;
14         int num_sheeps = remain_wt / wt_sheep;
15         remain_wt -= num_sheeps * wt_sheep;
16
17         if (remain_wt == 0)
18             cout << num_pigs << " * " << wt_pig << " + "
19                 << num_sheeps << " * " << wt_sheep << " = "
20                 << total_wt << endl;
21     }
22     return 0;
23 }
```

Problems of Comparing Floating-point Numbers

```
1  #include <iostream>      /* File: float-comparison-v1.cpp */
2  using namespace std;
3
4  int main()
5  {
6      float x = 0.1;
7      float product = 10.0 * x;
8
9      float sum = 0.0;
10     for (int i = 0; i < 10; ++i)
11         sum += x;
12
13     cout << "sum = " << sum << endl;
14     cout << "product = " << product << endl;
15     cout << "10.0 * x = " << 10.0 * x << endl;
16     cout << (sum == product) << endl;
17
18     return 0;
19 }
```

Problems of Comparing Floating-point Numbers ..

```
1  #include <iostream>      /* File: float-comparison.cpp */
2  using namespace std;
3
4  int main()
5  {
6      float x = 0.1;
7      float product = 10.0 * x;
8
9      float sum = 0.0;
10     for (int i = 0; i < 10; ++i)
11         sum += x;
12
13     // Set output precision to 10 significant figures
14     cout.precision(10);
15     // Print boolean outputs as true or false instead of 1 and 0
16     cout << boolalpha;    // boolean + alpha
17
18     cout << "sum = " << sum << endl;
19     cout << "product = " << product << endl;
20     cout << "10.0 * x = " << 10.0 * x << endl;
21     cout << (sum == product) << endl;
22     return 0;
23 }
```

Count Animals: Further Work

- Further check ways to compare floating point numbers [here](#).
- What if the total number of animals is not more than 10.
- What if we have three types of animal instead of two? (You may ignore the constraint on the total number of animals.)
- For those who know recursion, can you work out a recursive solution?

Part IV

GPA Calculator



- Assume the following letter grade to grade point conversion:

Letter Grade	Grade Point
A	4.0
B	3.0
C	2.0
D	1.0
F	0.0

- Design a program that calculates a student's GPA (grade point average).

Typical Output

```
No.  of credits of your course (0 to stop): 3
Your letter grade (A, B, C, D or F): A
No.  of credits of your course (0 to stop): 4
Your letter grade (A, B, C, D or F): B
No.  of credits of your course (0 to stop): 2
Your letter grade (A, B, C, D or F): E
Invalid input, please enter your grade again!
No.  of credits of your course (0 to stop): 2
Your letter grade (A, B, C, D or F): D
No.  of credits of your course (0 to stop): 0
You have taken a total of 9 credits ...
and your GPA is 2.88889
```

Program Requirements

- A student first enters the number of credits of his/her course.
- The program stops if the number of credits is ≤ 0 .
- The student then enters the letter grade A, B, C, D or F.
- Invalid letter grades are ignored and the student is prompted to re-enter the grade.
- The program shall calculate the total number of credits earned by the student and his/her GPA according to the following formula:

$$\frac{\sum_{i=1}^n (grade_i * credit_i)}{\sum_{i=1}^n credit_i}$$

Major components of the program:

- ❶ A loop for each taken course to
 - (a) ask for the number of credits
 - (b) ask for the letter grade
 - (c) convert the letter grade to grade points
 - (d) accumulate the total credits and grade points
- ❷ Calculate and output the GPA.

You will need some variables to hold:

- the number of credits
- the letter grade
- the converted grade points
- the sum of credits
- the sum of grade points

Variables

```
1 char grade;
2 /* Don't forget to initialize some of these variables */
3 int num_credits, total_num_credits = 0;
4 int total_grade_points = 0;
5     ↪ // Is it a good idea to use integer type here?
6
7 // Think about the output
8 cout << "You have taken a total of " << total_num_credits
9     << " credits ..." << endl << "and your GPA is "
10    << total_grade_points / total_num_credits << endl;
```

Recall the **usual arithmetic conversion** rules for binary operations:

- If all operands are int
 - compute using integer arithmetic
- If one operand is double/float
 - convert the other operand to double/float
 - compute using floating-point arithmetic
 - return the result in double/float

GPA Calculator: Using `if`

```
1  char grade;
2  int num_credits, total_num_credits = 0;
3  double total_grade_points = 0;
4
5  cout << "No. of credits of your course (0 to stop): ";
6  cin >> num_credits;
7  cout << "Your letter grade (A, B, C, D or F): ";
8  cin >> grade;
9  total_num_credits += num_credits; // Update total no. of credits
10
11 if (grade == 'A')           // Convert letter grade to grade point
12     total_grade_points += num_credits * 4;
13 else if (grade == 'B')
14     total_grade_points += num_credits * 3;
15 else if (grade == 'C')
16     total_grade_points += num_credits * 2;
17 else if (grade == 'D')
18     total_grade_points += num_credits * 1;
19 else if (grade == 'F')
20     total_grade_points += num_credits * 0;
21 else
22     total_num_credits -= num_credits;
```

GPA Calculator: Using **switch**

```
1  /*
2   * Codes for variables definition and initialization, and inputs
3   */
4
5  switch (grade)           // Convert letter grade to grade point
6  {
7      case 'A':
8          total_grade_points += num_credits * 4; break;
9      case 'B':
10         total_grade_points += num_credits * 3; break;
11     case 'C':
12         total_grade_points += num_credits * 2; break;
13     case 'D':
14         total_grade_points += num_credits * 1; break;
15     case 'F':
16         total_grade_points += num_credits * 0; break;
17     default:
18         total_num_credits -= num_credits;
19 }
```

GPA Calculator: Using `if` .. Allowing Small Case

```
1  /*
2   * Codes for variables definition and initialization, and inputs
3   */
4
5  if ((grade == 'A') || (grade == 'a'))
6      total_grade_points += num_credits * 4;
7
8  else if ((grade == 'B') || (grade == 'b'))
9      total_grade_points += num_credits * 3;
10
11 else if ((grade == 'C') || (grade == 'c'))
12     total_grade_points += num_credits * 2;
13
14 else if ((grade == 'D') || (grade == 'd'))
15     total_grade_points += num_credits * 1;
16
17 else if ((grade == 'F') || (grade == 'f'))
18     total_grade_points += num_credits * 0;
19
20 else
21     total_num_credits -= num_credits;
```

GPA Calculator: Using `switch` .. Allowing Small Case

```
1  switch (grade)                // Convert letter grade to grade point
2  {
3      case 'A':
4      case 'a':
5          total_grade_points += num_credits * 4; break;
6      case 'B':
7      case 'b':
8          total_grade_points += num_credits * 3; break;
9      case 'C':
10     case 'c':
11         total_grade_points += num_credits * 2; break;
12     case 'D':
13     case 'd':
14         total_grade_points += num_credits * 1; break;
15     case 'F':
16     case 'f':
17         total_grade_points += num_credits * 0; break;
18     default:
19         total_num_credits -= num_credits;
20 }
```

GPA Calculator: Complete Program .. Using constants

```
1  #include <iostream>      /* File: gpa.cpp */
2  using namespace std;
3
4  const double A = 4.0;    // Definition of Constants
5  const double B = 3.0;
6  const double C = 2.0;
7  const double D = 1.0;
8  const double F = 0.0;
9
10 int main()
11 {    // Variables definition and initialization
12     char grade;
13     int num_credits, total_num_credits = 0;
14     double total_grade_points = 0;
15
16     do
17     {
18         cout << "No. of credits of your course (0 to stop): ";
19         cin >> num_credits;
20
21         if (num_credits <= 0) // What does this do?
22             break;
23
24         cout << "Your letter grade (A, B, C, D or F): ";
25         cin >> grade;
26         total_num_credits += num_credits; // Update total no. of credits
```

GPA Calculator: Complete Program .. Using constants ..

```
27
28     switch (grade) // Convert letter grade to grade point
29     {
30         case 'A':    // No break here; execute code in case 'a'
31         case 'a':
32             total_grade_points += num_credits * A; break;
33         case 'B':
34         case 'b':
35             total_grade_points += num_credits * B; break;
36         case 'C':
37         case 'c':
38             total_grade_points += num_credits * C; break;
39         case 'D':
40         case 'd':
41             total_grade_points += num_credits * D; break;
42         case 'F':
43         case 'f':
44             total_grade_points += num_credits * F; break;
45         default :
46             cout << "Invalid input, please enter your grade again!\n"
47                  << " ";
48             total_num_credits -= num_credits;
49     }
50 } while (true);    // Why is this not an infinite loop?
51
```


GPA Calculator: Complete Program .. Using constants ..

```
52     cout << "You have taken a total of " << total_num_credits
53         << " credits ..." << endl << "and your GPA is "
54         << total_grade_points / total_num_credits << endl;
55
56     return 0;
57 }
```