C++ Programming
from Beginner to Expert

Chapter 7: Functions and an Introduction to Recursion

Milad Molaee

AMIRKABIR
University of Technology

July 25, 2022

Outline

Introduction

Experience has shown that the best way to develop and maintain large programs is to construct them from small, simple pieces, or components. This technique is called **divide and conquer**.

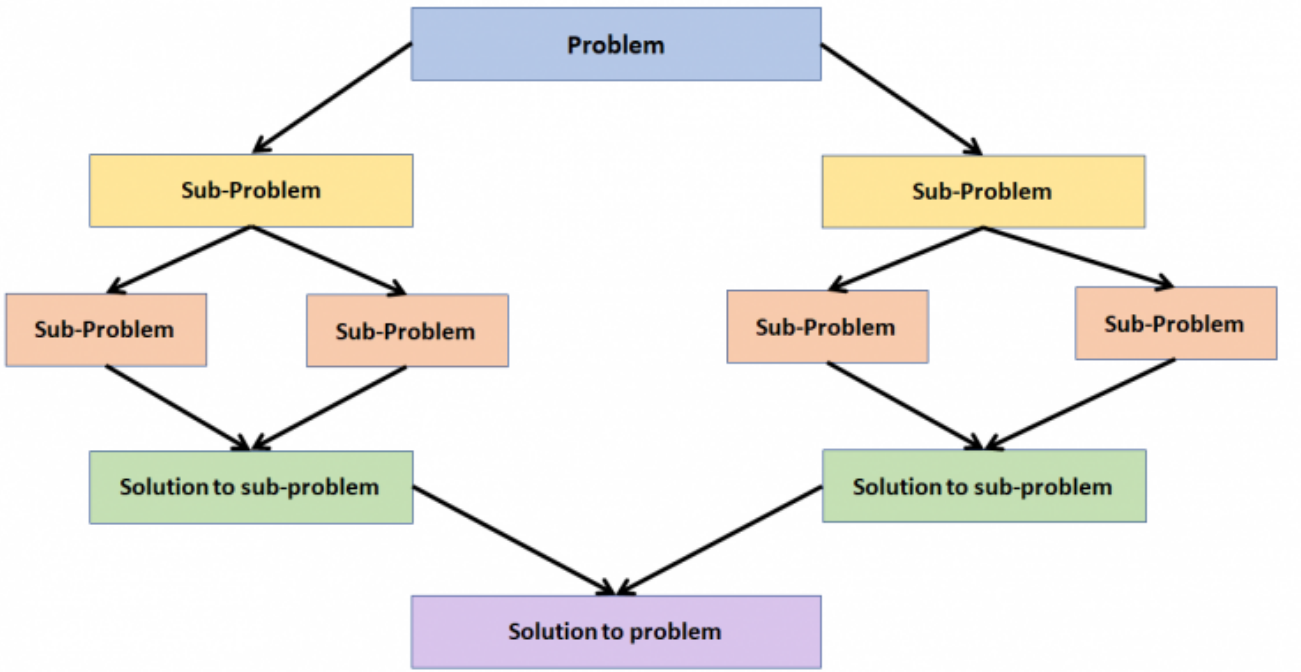


Figure: Divide and Conquer Algorithm

Introduction

AMIRKABIR University of Technology

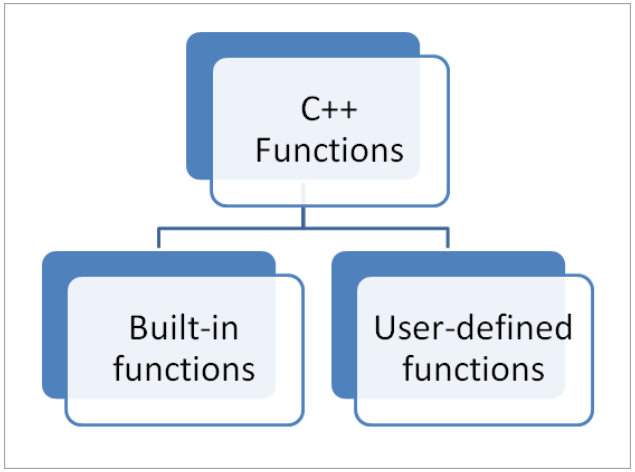
- A function is a block of code which only runs when it is called.
- You can pass data, known as parameters, into a function.
- Functions are used to perform certain actions, and they are important for reusing code: Define the code once, and use it many times.

C++ provides some **pre-defined functions**, such as `main()`, which is used to execute code. But you can also create your own functions to perform certain actions.

```
int main()
{
    cout << "Hello World";

    return 0;
}
```

Functions allow you to **modularize a program by separating its tasks** into self-contained units. You've used a **combination of library functions and your own functions** in almost every program you've written.



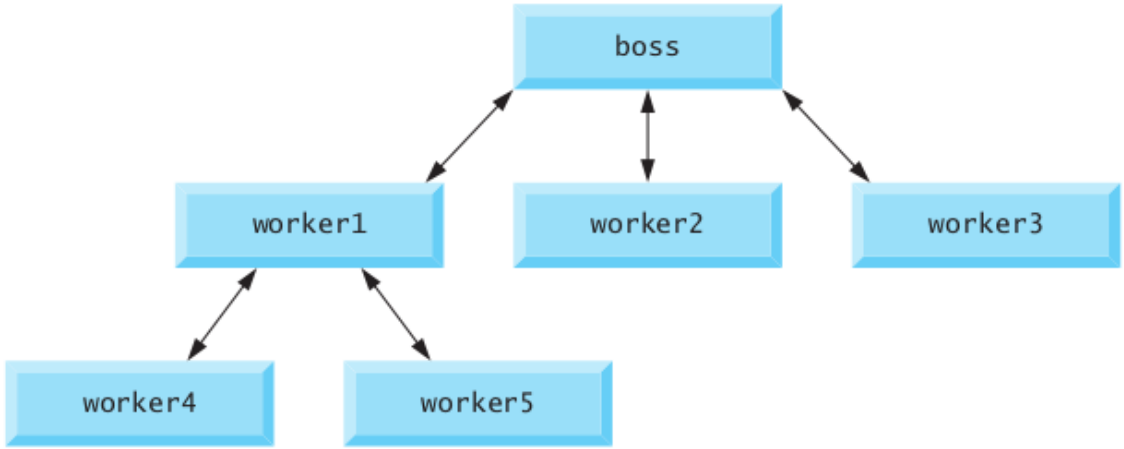
The C++ Standard Library provides a rich collection of functions for **common mathematical calculations, string manipulations, character manipulations, input/output, error checking** and many other useful operations.

Standard Template Library in C++

AMIRKABIR University of Technology

- Container
  - Sequence Container
  - Associative Container
  - Container Adapter
  - Unordered Associative Container
- Iterator
  - `begin()`
  - `next()`
  - `prev()`
  - `advance()`
  - `end()`
- Algorithm
  - Sorting and Searching Algorithm
  - minimum and maximum Operation
  - Numeric Algorithm
  - Modifying and non-modifying Algorithm

Hierarchical boss-function/worker-function relationship



Example

A boss (similar to the calling function) asks a worker (similar to a called function) to perform a task and report back (i.e., return) the results after completing the task. The boss function does not know how the worker function performs its designated tasks. The worker may also call other worker functions, unbeknownst to the boss.

Math Library Functions

Some functions, such as main, are not members of a class. These functions are called **global functions**. For example The `<cmath>` header provides a collection of functions that perform common mathematical calculations. For example, you can calculate the square root of `900.0` with the function call:

```
sqrt(900.0)
```

This expression evaluates to `30.0`. Function `sqrt` takes an argument of type `double` and returns a `double` result.

Function arguments may be **constants**, **variables** or **more complex expressions**.
If `c = 13.0`, `d = 3.0` and `f = 4.0`, then the statement:

```
cout << sqrt(c + d * f) << endl;
```

displays the square root of `13.0 + 3.0 * 4.0 = 25.0`—namely, `5.0`.

Function	Description	Example
`ceil(x)`	rounds x to the smallest integer not less than x	`ceil(9.2)` is `10.0` `ceil(-9.8)` is `-9.0`
`cos(x)`	trigonometric cosine of x (x in radians)	`cos(0.0)` is `1.0`
`exp(x)`	exponential function e^x	`exp(1.0)` is `2.718282` `exp(2.0)` is `7.389056`

`fabs(x)`	absolute value of x	`fabs(5.2)` is `5.2`
		`fabs(0.0)` is `0.0`
		`fabs(-1.2)` is `1.2`

`floor(x)`	rounds x to the largest integer not greater than x	`floor(9.2)` is `9.0` `floor(-9.8)` is `-10.0`
`fmod(x)`	remainder of x/y as a floating-point number	`fmod(2.6, 1.2)` is `0.2`
`log(x)`	natural logarithm of x (base e)	`log(2.718282)` is `1.0` `log(7.389056)` is `2.0`
`log10(x)`	logarithm of x (base 10)	`log(10.0)` is `1.0` `log(100.0)` is `2.0`
`pow(x)`	x raised to power y (x^y)	`pow(2, 7)` is `128` `pow(25, 0.5)` is `5`
`sin(x)`	trigonometric sine of x (x in radians)	`sin(0.0)` is `0.0`
`sqrt(x)`	square root of x (where x is a nonnegative value)	`sqrt(64)` is `8`
`tan(x)`	trigonometric tangent of x (x in radians)	`tan(0.0)` is `0.0`

Function-Prototype and Argument-Coercion Notes

AMIRKABIR University of Technology

For a function that's not defined in a class, you must either define the function before using it or you must declare that the function exists.

```
int maximum(int x, int y, int z); // function prototype
```

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

int maximum(int x, int y, int z); // function prototype

int main()
{
    cout << "Enter three integer values: ";
    int int1, int2, int3;
    cin >> int1 >> int2 >> int3;
    // invoke maximum
    cout << "The maximum integer value is: "
         << maximum(int1, int2, int3) << endl;
}

int maximum(int x, int y, int z) // returns the largest of three integers
{
    int maximumValue{x}; // assume x is the largest to start
    // determine whether y is greater than maximumValue
    if (y > maximumValue)
    {
        maximumValue = y; // make y the new maximumValue
    }
    // determine whether z is greater than maximumValue
    if (z > maximumValue)
    {
        maximumValue = z; // make z the new maximumValue
    }
    return maximumValue;
}
```

Output

```
Enter three integer grades: 86 67 75
The maximum integer value is: 86
```

Output

```
Enter three integer grades: 67 86 75
The maximum integer value is: 86
```

Output

```
Enter three integer grades: 67 75 86
The maximum integer value is: 86
```

When compiling the program, the compiler uses the prototype to:

- Ensure that maximum's first line (**function header**) (line 17) matches its **prototype** (line 5).
- Check that the call to maximum (**function call**) (line 14) contains the correct number and types of arguments, and that the types of the arguments are in the correct order (in this case, all the arguments are of the same type).
- Ensure that the value returned by the function can be used correctly in the expression that called the function, for example, for a function that returns void you cannot call the function on right side of an assignment.
- Ensure that each argument is consistent with the type of the corresponding parameter, for example, a parameter of type double can receive values like 7.35, 22 or –0.03456, but not a string like "hello". If the arguments passed to a function do not match the types specified in the function's prototype, the compiler attempts to convert the arguments to those types.

Data types		Size (byte)
`long double`		12
`double`		8
`float`		4
`unsigned long long int`	synonymous with `unsigned long long`	8
`long long int`	synonymous with `long long`	8
`unsigned long int`	synonymous with `unsigned long`	4
`long int`	synonymous with `long`	4
`unsigned int`	synonymous with `unsigned`	4
`int`		4
`unsigned short int`	synonymous with `unsigned short`	2
`short int`	synonymous with `short`	2
`unsigned char`		1
`char`		1
`bool`		1

Data Type	Range	Size (byte)
`long double`	$-1.18973 \times 10^{4932}$ to 1.18973×10^{4932}	12
`double`	-1.79769×10^{308} to 1.79769×10^{308}	8
`float`	$-3.40282e \times 10^{38}$ to 3.40282×10^{38}	4
`unsigned long long int`	0 to $2^{64} - 1$	8
`long long int`	$-(2^{63})$ to $2^{63} - 1$	8
`unsigned long int`	0 to $4,294,967,295$	4
`long int`	$-2,147,483,648$ to $2,147,483,647$	4
`unsigned int`	0 to $4,294,967,295$	4
`int`	$-2,147,483,648$ to $2,147,483,647$	4
`unsigned short int`	0 to $65,535$	2
`short int`	$-32,768$ to $32,767$	2
`unsigned char`	0 to 255	1
`char`	-128 to 127	1
`bool`	`true / false`	1

C++ Standard Library Headers

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Standard Template Library - Part 1

AMIRKABIR University of Technology

Standard Library header	Explanation
`<iostream>`	Contains function prototypes for the C++ standard input and output functions.
`<iomanip>`	Contains function prototypes for stream manipulators that format streams of data.
`<cmath>`	Contains function prototypes for math library functions.
`<cstdlib>`	Contains function prototypes for conversions of numbers to text, text to numbers, memory allocation, random numbers and various other utility functions.
`<ctime>`	Contains function prototypes and types for manipulating the time and date.
`<array>`, `<vector>`, `<list>`, `<forward_list>`, `<deque>`, `<queue>`, `<stack>`, `<map>`, `<unordered_map>`, `<unordered_set>`, `<set>`, `<bit_set>`	These headers contain classes that implement the C++ Standard Library containers. Containers store data during a program's execution. A container is a holder object that stores a collection of other objects (its elements). They are implemented as class templates, which allows great flexibility in the types supported as elements. The container manages the storage space for its elements and provides member functions to access them, either directly or through iterators (reference objects with similar properties to pointers).

Standard Template Library - Part 2

AMIRKABIR University of Technology

Standard Library header	Explanation
`<cctype>`	Contains function prototypes for functions that test characters for certain properties (such as whether the character is a digit or a punctuation), and function prototypes for functions that can be used to convert lowercase letters to uppercase letters and vice versa.
`<cstring>`	Contains function prototypes for C-style string-processing functions.
`<typeinfo>`	Contains classes for runtime type identification (determining data types at execution time).
`<exception>`, `<stdexcept>`	These headers contain classes that are used for exception handling.
`<memory>`	Contains classes and functions used by the C++ Standard Library to allocate memory to the C++ Standard Library containers.
`<fstream>`	Contains function prototypes for functions that perform input from and output to files on disk.
`<string>`	Contains the definition of class string from the C++ Standard Library.
`<sstream>`	Contains function prototypes for functions that perform input from strings in memory and output to strings in memory.

Standard Template Library - Part 3

AMIRKABIR University of Technology

Standard Library header	Explanation
`<functional>`	Contains classes and functions used by C++ Standard Library algorithms.
`<iterator>`	Contains classes for accessing data in the C++ Standard Library containers.
`<algorithm>`	Contains functions for manipulating data in C++ Standard Library containers.
`<cassert>`	Contains macros for adding diagnostics that aid program debugging.
`<cfloat>`	Contains the floating-point size limits of the system.
`<climits>`	Contains the integral size limits of the system.
`<cstdio>`	Contains function prototypes for the C-style standard input/output library functions.
`<local>`	Contains classes and functions normally used by stream processing to process data in the natural form for different languages (e.g., monetary formats, sorting strings, character presentation, etc.).
`<limits>`	Contains classes for defining the numerical data type limits on each computer platform—this is C++'s version of <climits> and <cfloat>.
`<utility>`	Contains classes and functions that are used by many C++ Standard Library headers.

Random-Number Generation

AMIRKABIR University of Technology

The element of chance can be introduced into computer applications by using the C++ Standard Library function rand. Consider the following statement:

```
i = rand();
```

The function rand generates an unsigned integer between 0 and `RAND_MAX` (a symbolic constant defined in the `<cstdlib>` header). You can determine the value of `RAND_MAX` for your system simply by displaying the constant. If rand truly produces integers at random, every number between 0 and `RAND_MAX` has an equal chance (or probability) of being chosen each time rand is called.

To produce integers in the range 0 to 5, we use the remainder operator (%) with rand as follows:

```
i = rand() % 6;
```

This is called scaling. The number 6 is called the scaling factor.

```cpp
#include <iostream>
#include <iomanip>
#include <cstdlib> // contains function prototype for rand
using namespace std;

int main()
{
    // loop 20 times
    for (unsigned int counter{1}; counter <= 20; ++counter)
    {
        // pick random number from 1 to 6 and output it
        cout << setw(10) << (1 + rand() % 6);

        // if counter is divisible by 5, start a new line of output
        if (counter % 5 == 0)
        {
            cout << endl;
        }
    }
}
```

```cpp
#include <iostream>
#include <iomanip>
#include <cstdlib> // contains function prototype for rand
using namespace std;

int main()
{
    unsigned int frequency1{0}; // count of 1s rolled
    unsigned int frequency2{0}; // count of 2s rolled
    unsigned int frequency3{0}; // count of 3s rolled
    unsigned int frequency4{0}; // count of 4s rolled
    unsigned int frequency5{0}; // count of 5s rolled
    unsigned int frequency6{0}; // count of 6s rolled
    int face;
                // stores each roll of the die
    // summarize results of 60,000,000 rolls of a die
    for (unsigned int roll{1}; roll <= 60000000; ++roll)
    {
        face = 1 + rand() % 6; // random number from 1 to 6

        // determine roll value 1-6 and increment appropriate counter
        switch (face)
        {
        case 1:
            ++frequency1; // increment the 1s counter
            break;
        case 2:
            ++frequency2; // increment the 2s counter
            break;
```

```
30          case 3:
31              ++frequency3; // increment the 3s counter
32              break;
33          case 4:
34              ++frequency4; // increment the 4s counter
35              break;
36          case 5:
37              ++frequency5; // increment the 5s counter
38              break;
39          case 6:
40              ++frequency6; // increment the 6s counter
41              break;
42          default: // invalid value
43              cout << "Program should never get here!";
44          }
45      }
46
47      cout << "Face" << setw(13) << "Frequency" << endl; // output headers
48      cout << "1" << setw(13) << frequency1
49           << "\n2" << setw(13) << frequency2
50           << "\n3" << setw(13) << frequency3
51           << "\n4" << setw(13) << frequency4
52           << "\n5" << setw(13) << frequency5
53           << "\n6" << setw(13) << frequency6 << endl;
54  }
```

```cpp
#include <iostream>
#include <iomanip>
#include <cstdlib> // contains prototypes for functions srand and rand
using namespace std;

int main()
{
    unsigned int seed{0}; // stores the seed entered by the user

    cout << "Enter seed: ";
    cin >> seed;

    srand(seed); // seed random number generator

    // loop 10 times
    for (unsigned int counter{1}; counter <= 10; ++counter)
    {
        // pick random number from 1 to 6 and output it
        cout << setw(10) << (1 + rand() % 6);

        // if counter is divisible by 5, start a new line of output
        if (counter % 5 == 0)
        {
            cout << endl;
        }
    }
}
```

Seeding the Random-Number Generator with the Current Time

AMIRKABIR University of Technology

To randomize without having to enter a seed each time, we can use a statement like

```
srand(static_cast<unsigned int>(time(0)));
```

This causes the computer to read its clock to obtain the value for the seed. Function time (with the argument 0 as written in the preceding statement) typically returns the current time as the number of seconds since January 1, 1970, at midnight Greenwich Mean Time (GMT). This value (which is of type time_t) is converted to an unsigned int and used as the seed to the random-number generator—the static_cast in the preceding statement eliminates a compiler warning that's issued if you pass a time_t value to a function that expects an unsigned int. The function prototype for time is in <ctime>.

Previously, we simulated the rolling of a six-sided die with the statement

```
unsigned int face{1 + rand() % 6};
```

which always assigns an integer (at random) to variable face in the range 1 face 6. The width of this range (i.e., the number of consecutive integers in the range) is 6 and the starting number in the range is 1. Referring to the preceding statement, we see that the width of the range is determined by the number used to scale rand with the remainder operator (i.e., 6), and the starting number of the range is equal to the number (i.e., 1) that is added to the expression rand % 6. We can generalize this result as

```
type variableName{shiftingValue + rand() % scalingFactor};
```

where the shiftingValue is equal to the first number in the desired range of consecutive integers and the scalingFactor is equal to the width of the desired range of consecutive integers.

Case Study: Game of Chance; Introducing Scoped enums

AMIRKABIR University of Technology

Question

A player rolls two dice. Each die has six faces. These faces contain 1, 2, 3, 4, 5 and 6 spots. After the dice have come to rest, the sum of the spots on the two upward faces is calculated. If the sum is 7 or 11 on the first roll, the player wins. If the sum is 2, 3 or 12 on the first roll (called "craps"), the player loses (i.e., the "house" wins). If the sum is 4, 5, 6, 8, 9 or 10 on the first roll, then that sum becomes the player's "point." To win, you must continue rolling the dice until you "make your point." The player loses by rolling a 7 before making the point.

Answer - Part 1

```cpp
#include <iostream>
#include <cstdlib> // contains prototypes for functions srand and rand
#include <ctime>   // contains prototype for function time
using namespace std;

unsigned int rollDice(); // rolls dice, calculates and displays sum

int main()
{
    // scoped enumeration with constants that represent the game status
    enum class Status
    {
        CONTINUE,
        WON,
        LOST
    }; // all caps in constants

    // randomize random number generator using current time
    srand(static_cast<unsigned int>(time(0)));
    unsigned int myPoint{0};             // point if no win or loss on first roll
    Status gameStatus;                   // can be CONTINUE, WON or LOST
    unsigned int sumOfDice{rollDice()}; // first roll of the dice

    // determine game status and point (if needed) based on first roll
    switch (sumOfDice)
    {
    case 7:  // win with 7 on first roll
    case 11: // win with 11 on first roll
        gameStatus = Status::WON;
        break;
```

Answer - Part 2

```
31      case 2:  // lose with 2 on first roll
32      case 3:  // lose with 3 on first roll
33      case 12: // lose with 12 on first roll
34          gameStatus = Status::LOST;
35          break;
36      default:                            // did not win or lose, so remember point
37          gameStatus = Status::CONTINUE; // game is not over
38          myPoint = sumOfDice;           // remember the point
39          cout << "Point is " << myPoint << endl;
40          break; // optional at end of switch
41      }
42
43      // while game is not complete
44      while (Status::CONTINUE == gameStatus) // not WON or LOST
45      {
46          sumOfDice = rollDice(); // roll dice again
47
48          // determine game status
49          if (sumOfDice == myPoint)
50          { // win by making point
51              gameStatus = Status::WON;
52          }
53          else
54          {
55              if (sumOfDice == 7)
56              { // lose by rolling 7 before point
57                  gameStatus = Status::LOST;
58              }
59          }
60      }
```

Answer - Part 3

```
61
62      // display won or lost message
63      if (Status::WON == gameStatus)
64      {
65          cout << "Player wins" << endl;
66      }
67      else
68      {
69          cout << "Player loses" << endl;
70      }
71  }
72
73  // roll dice, calculate sum and display results
74  unsigned int rollDice()
75  {
76      int die1{1 + rand() % 6}; // first die roll
77      int die2{1 + rand() % 6}; // second die roll
78
79      int sum{die1 + die2};     // compute sum of die values
80
81      // display results of this roll
82      cout << "Player rolled " << die1 << " + " << die2
83           << " = " << sum << endl;
84      return sum;
85  }
```

Output of run 1

```
Player rolled 2 + 5 = 7
Player wins
```

Output of run 2

```
Player rolled 3 + 3 = 6
Point is 6
Player rolled 5 + 3 = 8
Player rolled 4 + 5 = 9
Player rolled 2 + 1 = 3
Player rolled 1 + 5 = 6
Player wins
```

Output of run 3

```
Player rolled 6 + 6 = 12
Player loses
```

Output of run 4

```
Player rolled 1 + 3 = 4
Point is 4
Player rolled 4 + 6 = 10
Player rolled 2 + 4 = 6
Player rolled 6 + 4 = 10
Player rolled 2 + 3 = 5
Player rolled 2 + 4 = 6
Player rolled 1 + 1 = 2
Player rolled 4 + 4 = 8
Player rolled 4 + 3 = 7
Player loses
```

C++11 Random Numbers

AMIRKABIR University of Technology

```cpp
#include <iostream>
#include <iomanip>
#include <random> // contains C++11 random number generation features
#include <ctime>
using namespace std;

int main()
{
    // use the default random-number generation engine to
    // produce uniformly distributed pseudorandom int values from 1 to 6
    default_random_engine engine{static_cast<unsigned int>(time(0))};
    uniform_int_distribution<unsigned int> randomInt{1, 6};

    // loop 10 times
    for (unsigned int counter{1}; counter <= 10; ++counter)
    {
        // pick random number from 1 to 6 and output it
        cout << setw(10) << randomInt(engine);

        // if counter is divisible by 5, start a new line of output
        if (counter % 5 == 0)
        {
            cout << endl;
        }
    }
}
```

Scope Rules

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Function-Call Stack and Activation Records

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Inline Functions

Implementing a program as a set of functions is good from a software engineering stand-point, but function calls involve execution-time overhead. C++ provides inline functions to help reduce function-call overhead. Placing the qualifier inline before a function's return type in the function definition advises the compiler to generate a copy of the function's body code in every place where the function is called (when appropriate) to avoid a function call. This often makes the program larger. The compiler can ignore the inline qualifier and generally does so for all but the smallest functions. Reusable inline functions are typically placed in headers, so that their definitions can be included in each source file that uses them.

References and Reference Parameters

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Default Arguments

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Unary Scope Resolution Operator

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Function Overloading

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Function Templates

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Recursion

AMIRKABIR University of Technology

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Example Using Recursion: Fibonacci Series

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Recursion vs. Iteration

AMIRKABIR University of Technology

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Summery and Conclusion

AMIRKABIR University of Technology

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Exercises

AMIRKABIR University of Technology

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

First Program in C++: Printing a Line of Text

AMIRKABIR University of Technology

Consider a simple program that prints a line of text (Fig. 2.1). This program illustrates several important features of the C++ language. The text in lines 1–10 is the program's source code. The line numbers are not part of the source code.

```
1  // Fig. 2.1: fig02_01.cpp
2  // Text-printing program.
3  #include <iostream> // enables program to output data to the screen
4
5  // function main begins program execution
6  int main() {
7  std::cout << "Welcome to C++!\n"; // display message
8
9  return 0; // indicate that program ended successfully
10 } // end function main
```

output

```
Welcome to C++!
```

The Stream Insertion Operator and Escape Sequences

AMIRKABIR University of Technology

Escape sequence	Description
\n	Newline. Position the screen cursor to the beginning of the next line.
\t	Horizontal tab. Move the screen cursor to the next tab stop.
\r	Carriage return. Position the screen cursor to the beginning of the current line; do not advance to the next line.
\a	Alert. Sound the system bell.
\\	Backslash. Used to print a backslash character.
\'	Single quote. Used to print a single-quote character.
\"	Double quote. Used to print a double-quote character.

Arithmetic Operators

AMIRKABIR University of Technology

Operation	Arithmetic operator	Algebraic expression	C++ expression
Addition	+	$f + 7$	`f + 7`
Subtraction	-	$p - c$	`p - c`
Multiplication	*	bm or $b \cdot m$	`b * m`
Division	/	$\frac{x}{y}$ or x/y or $x \div y$	`x / y`
Remainder	%	$r \bmod s$	`r % s`

Precedence of Arithmetic Operators

AMIRKABIR University of Technology

Operator(s)	Operation(s)	Order of evaluation (precedence)
`( )`	Parentheses	Evaluated first. For nested parentheses, such as in the expression `a * (b + c / (d + e))`, the expression in the innermost pair evalu ates first. [Caution: If you have an expression such as `(a + b) * (c - d)` in which two sets of parentheses are not nested, but appear "on the same level," the C++ Standard does not specify the order in which these parenthesized subexpressions will evaluate.]

`*`
`/`
`%`
	Multiplication
	Division
	Remainder
		Evaluated second. If there are several, they're evaluated left to right.

`+` `-`	Addition Subtraction	Evaluated last. If there are several, they're evaluated left to right.

Decision Making: Equality and Relational Operators

AMIRKABIR University of Technology

Algebraic relational or equality operator	C++ relational or equality operator	Sample C++ condition	Meaning of C++ condition
Relational operators			
$>$	`>`	`x > y`	x is greater than y
$<$	`<`	`x < y`	x is less than y
$\geq$	`>=`	`x >= y` or	x is greater than or equal to y
$\leq$	`<=`	`x <= y`	x is less than or equal to y
Equality operators			
$=$	`==`	`x == y`	x is equal to y
$\neq$	`!=`	`x != y`	x is not equal to y