C++ Exercises

Basic Syntax

Make sure that you include, at least, some comments in your code.

1. Write the classic hello world program — a program that prints the text string "hello world" (or any other string you like).

2. Write a program to calculate the area and circumference of a circle of radius 5.0. Change the code so that the radius may be specified by the user.

3. Write a program to compute the value of $N!$ for a given integer N. Include error checking code to handle the case when N is negative. Optional: Write versions of the code using `for`, `while` and `do-while` loops. Which do you prefer?

Functions

1. Write a function to compute the value of $\sin x$ using the first N terms of the power series expansion (N should be an argument to the function):

$$\sin x \approx \sum_{k=0}^{N} (-1)^k \frac{x^{2k+1}}{(2k+1)!}$$

   How many terms of the series are required to agree with the built-in sin function. Produce output of the two functions side by side for the values $0, \pi/6, \pi/4, \pi/2, 2\pi/3$ and π.

2. Write a function that takes the arguments a, b and c, computes and then displays the roots of the quadratic equation

$$ax^2 + bx + c = 0.$$

   You will need to identify whether the roots are real or complex. If the roots are complex, display the results in the form $A + Bi$

3. Write a recursive function to calculate the nth Fibonacci number. Recall that the Fibonacci numbers, f_n, are defined by

$$f_0 = 0, \quad f_1 = 1, \quad f_n = f_{n-1} + f_{n-2}$$

   Use your function to calculate and display the Fibonacci quotient, $q_n = f_n/f_{n-1}$, for a number of different values of n. You should find that as n increases q_n converges to the golden mean, $(1+\sqrt{5})/2 \approx 1.618$.

Pointers

1. Create an integer array of dimension 10. Print out the memory addresses of each element in the array. What do you notice?

2. Create two versions of a function that returns the cube of a floating point number. Pass the variable itself in one version and the pointer to the variable in the other. If you overwrite the variable in the function does it matter? If so, why?

3. Write a function that calculates the maximum value of an array of numbers.

File I/O

1. Write a program to output a file containing three columns of data: x, $\sin(x)$ and $\cos(x)$ for $x \in [0, 4\pi]$. Use at least 50 points in the range. If you like, check the data using gnuplot.

2. Write a program to read in five numbers from an ASCII file. What do you need to know about the format of the numbers?

Classes

1. Write a bank account class. You should be able to make deposits and withdrawals and display the balance, which must be a private integer variable. Member functions should be:

```
void account::withdraw(int amount); //Take from account
void account::deposit(int amount); // Put into account
int account::balance(void); //Read out balance
```

   Make sure that the account constructor function initialises the balance to zero. If you like, add an overloaded constructor function to set the initial balance.

2. Write a class to implement a simple queue of up to 10 integers. A queue is a first-in, first-out (FIFO) data structure. Member functions should be:

```
void queue::put(int item); //Add item to queue
int queue::get(void); //Get next item from queue
```

   Example use

```
queue Q;

Q.put(1);    //Queue contains 1
Q.put(2);    //Queue contains 1 2
Q.put(3);    //Queue contains 1 2 3

cout << Q.get(); //Prints 1, queue contains 2 3
cout << Q.get(); //Prints 2, queue contains 3
```

Overloading

1. Define three classes: circle, square and (equilateral) triangle that each contain a (public) variable representing the characteristic dimension of each shape: i. e., radius or length of one side. Now write three different functions that calculate the area of each shape. The functions should all be called `area()` and use overloading to distinguish between the three shapes.

2. Write a class to handle fractions. Define addition, subtraction, multiplication and division operators for these fractions. The results need not be simplified to take account of common factors. Make the numerator and denominator private members of the class and write an output routine to display the fractions as e. g. 1/3. Note that you will need to define the overloaded operators as friend functions.

Inheritance

1. Create a base class containing public, private and protected variables. e. g.

```
class base
 {
  private:   double a;
  protected: double b;
  public:    double c;
 };
```

   Now create a number of different derived classes using public, protected and private keywords before the base class name and containing additional variables. e. g.

```
class derived: public base //Try replacing public with private or protected
 {
  private:   double A;
  protected: double B;
  public:    double C;
 };
```

   Create some objects and try and use each of the base and derived variables from `main()`. e. g.

```
main()
 {
  base B; derived D;

  cout << D.A << D.a;
 }
```

   What do you notice when you try to compile? Make sure that you understand how the encapsulation works!

2. Combine the account class and the queue class from the class exercises to create a bank account class that can print out the last 10 transactions.

A mathematical example: sparse arrays

1. Create a class array that defines integer arrays of a size specified via constructor function. Make sure that you define a destructor function to clean up any memory allocated by the constructor. Define a copy constructor to allocate new memory every time a local copy of a variable is made. Overload `=`, `+` and `-` to operate on these arrays, do you need to be careful? How will you set and read the elements of the array? You can overload the `[]` operator so that your array acts as a normal array. e. g.

```
int &operator[](int i) {return ....;}
```

   `i` is the index between the square brackets and this function must be a member function of the class, it cannot be a friend. Modify the `[]` operator so that your array is indexed from 1 to N. Also, try to include bounds checking code: if the index is not between 1 and N produce and error message and quit. Member functions:

```
 array(int arg); //Constructor function
~array(void); //Destructor function
 array(const array &old_array); //Copy constructor
 array operator=(const array &old_array); //Overloaded =
 int &operator[](int index); //Overloaded []
 void print(void); //Print contents of array
 friend array operator+(const array &a, const array &b); //Overloaded +
 friend array operator-(const array &a, const array &b); //Overloaded -
```

2. Create a class `spase_array` derived from array. This class should only store elements of the array that are non-zero. For simplicity assume that the array is indexed from 1 to 100. How could you generalise this to any range? Think carefully about how you want to access members of the array. Overload addition and subtraction to handle the cases when a normal array is combined with a sparse array.

3. Implement a sparse array class using a linked list data structure. For each list element, a linked list stores the data itself and a link to the next piece of data. The advantage of this structure is that it can easily allocate new memory as required. Make sure that new data is inserted in numerical order of the indices. What sorts of safety checks do you need to include? Overload addition, subtraction to handle any combination of the existing array classes.