Example: The Trapezium Rule for Integration

Use classes to create a generic integrate function to solve

\[ I = \int_a^b f(x)dx \]

where \( f(x) \) is supplied as an argument to the function. Now use it to solve:

\[ I = \int_0^\infty \frac{1}{\sqrt{2\pi(1+x)}}e^{-\frac{x^2}{4\kappa}}dx \]

with \( \kappa = 1, 2, \ldots, 10 \).

Solution:

Stage I:
First we must create an integrand class containing the function that we wish to integrate. It must contain a function to return the value of \( f(x) \) and also the parameter \( \kappa \). There are some choices to make:

- Do we make the parameter \( \kappa \) a public or private member? If it is private then we will need to write a ‘set’ function;

- Do we wish to have a default value for \( \kappa \)? This can stop unwanted errors, for instance if we divide by \( \kappa \) somewhere and call the function before the value of \( \kappa \) is set we may get errors appearing. To set a default value we must write a constructor function.

Stage II:
Once your integrand class has been written and fully tested (you can compare results with the previous codes you have written) we can then move onto writing a function that integrates the integrand class. Inputs and outputs are:

- takes the range \( a, b \), the integrand function \( f \) and the number of steps \( n \) as an input to the function;
• outputs the value of the integral.

Fully test your function against previous codes.
See website for codes.

Building classes step by step

4.1 Change the integrand function to solve

\[ I = \int_{a}^{b} \frac{1}{\alpha(1 + x)} e^{-\frac{x^2}{s}} dx \]

with \( a = 0, \ b = 1, \ \alpha = 0.02, \) and \( s = 0.1. \) What do you have to change now in your code? Compare and contrast this to procedural approach in week 3.

4.2 Now make a new class \texttt{Integrate}.

(i) Add the trapezium function as a public member;

(ii) Add \( a, \ b \) and \( n \) as members to the class, we can then remove them from the argument list in the trapezium function;

(iii) Write a function to set the value of \( a, \ b \) and \( n; \)

(iv) Use the new class in your main code... it should look something like...

```cpp
// declare an instance of the Integrate class
Integrate I;
// set parameters in the class
I.set_parameters(0.0, 1.0, 100);
// solve integral;
cout << "I = " << I.trapezium(f) << "\n";
```

4.3 Now add another member function that solves the integral using the simpsons rule. Use it to compare the value of the integral using the two different methods.

4.4 What happens if you try to integrate two different functions with \texttt{Integrate}? We will learn more about this in week 5.

Point Class

4.5 Write a program with the Point class from the lecture. Also write in the function operator to add Points. Change the data type from an int to a double.

4.6 Add a new member function that swaps the x and y values in a Point.

4.7 Create a new function that calculates the distance between two Points.
4.8 Now overload the operator so that you can multiply a double by a Point. Evaluate:

\[ z = a + 15b \]

with \( a = (0.13, 2.45) \) and \( b = (3.78, 6.98) \).

4.9 Overload the `std::ostream <<` operator to output a Point in the form \((x, y)\).

**Classes**

4.10 Read through the notes, then 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:

```cpp
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.

4.11 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:

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

Example use

```cpp
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
```