Introduction to Modern C++

Lab Session 3

Sorting, Numerical integration, and Angry Birds

This lab session is dedicated to a first introduction to algorithms, through the branching and looping statements. Last week's lab challenged you to translate simple computations into C++ code; today's lab is about translating simple algorithms into C++ code.

1 Sorting

Let (u_k) be a sequence of numbers (e.g. 3, 1, 4, 1, ...). A typical exercise for Computer Science students is to write an algorithm that *sorts* (u_k), i.e. that outputs these numbers in increasing sequence: 1, 1, 3, 4, ...

Let's start with a very naive algorithm. It works as follows:

1. Input a list L of integer numbers
2. Find the smallest number x in L
3. Output x
4. Remove x from L
5. Return to step 2 until L is empty

Here is an implementation of this algorithm:

```
#include <iostream>
#include <algorithm>
#include <vector>

int main() {
  std::vector<int> L {9, 8, 7, 6, 5, 4, 3, 2, 1, 0};

  while (L.size() > 0) {
    auto smallest_element = std::min_element(L.begin(), L.end());
    std::cout << L[smallest_element - L.begin()] << ", ";
    L.erase(smallest_element);
  }
}
```

Test this algorithm and make sure that it works. What is `L.size()`? What is `auto`? What is `min_element`? What are `L.begin(), L.end()` ? What is `L.erase`?

Task 2. In fact, this algorithm is not very fast. Indeed, assume the list has N elements. Finding the smaller number takes in average $N/2$ steps. Removing a number takes in average $N/2$ steps. And we repeat exactly N times. In total, we perform in average N^2 operations. What's the worst-case scenario? Assume $N = 10^6$, how many operations are performed?

Task 3. Use the standard library function `std::sort(L.begin(), L.end())` to sort L quickly.
Hint: You can print the contents of L with the following code:

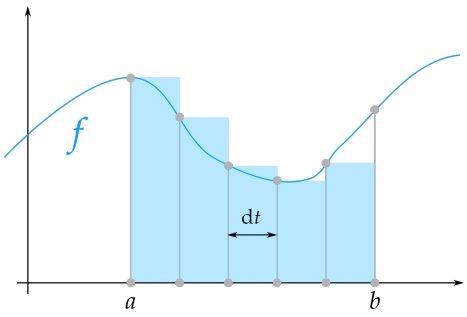
```
for(int x : L) {
  std::cout << x << ", ";
}
```

This algorithm is much faster, it performs on average around $N \log(N)$ operations. Assume $N = 10^6$, how many operations are performed?

• **Task 4.** Try to come up with your own sorting algorithm. Test it, and evaluate its complexity!

2 Numerical integration

In this section we will implement a numerical integrator, i.e. a program that takes some continuous function f as input, and outputs (an approximate value of) $\int_a^b f(t)\,\mathrm{d}t$.



Remark: For this section you may need to use the `--std=c++14` option during compilation!

Reminder: This integral can be interpreted as the "area under the curve" of f. We can approximate this area by dividing the interval $[a, b]$ into segments, and assuming that f is constant on this segment. This gives the following:

$$\int_a^b f(t)\,\mathrm{d}t \approx \frac{1}{N}\sum_{i=1}^{N} f\left(a + \frac{i(b-a)}{N}\right) \tag{1}$$

This is illustrated in the figure on the right, where $\mathrm{d}t = (a-b)/N$. The expression is exact in the limit $N \to \infty$. You can use $N = 100$ to test your code.

Task 4. Write the following function:

```
double f(double t) {
  return t;
}
```

What does this function do? What is the value of $\int_a^b f(t)\,\mathrm{d}t$?

Task 5. Implement Equation (1) with a `for` loop. Test that you find the correct value of $\int_a^b f(t)\,\mathrm{d}t$.

Task 6. Make a function:

```
double Integrate(double a, double b, auto f) {
  // Your code here
}
```

This function takes as input a, b and a function f, and returns $\int_a^b f(t)\,\mathrm{d}t$.

Task 7. Compute the following integrals:

$$\int_0^1 t^2\,\mathrm{d}t = \frac{1}{3}, \qquad \int_{-\pi}^{+\pi} t^2\cos(t)\,\mathrm{d}t = -4\pi, \qquad \int_{-\infty}^{+\infty} \frac{\mathrm{d}t}{\cosh t} = \pi, \qquad \int_{-\infty}^{+\infty} \frac{\mathrm{d}t}{\cosh^2 t} = 2$$

Hint: Replace ∞ by 100. Why can we do this?

Challenge: How to improve the accuracy of numerical integration?

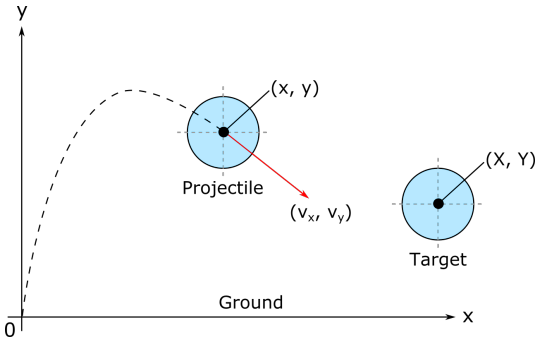
Remark: This approach is simple, but it doesn't work very well for multiple integrals. We will see in a homework a very powerful alternative, known as Monte-Carlo integration.

3 • Angry Birds

The Angry Birds is a 2009 game whose main goal is to throw (angry) birds at stuff. Disregarding the graphics (for now), the aim of this section is to implement the main algorithm :

1. Input initial angle α and velocity $v \in [v_{\text{min}}, v_{\text{max}}]$. Let $t = 0$.
2. While there is no collision do:
   (a) Move the projectile according to Physics
   (b) Detect potential collisions
   (c) Increment time $t \mapsto t + \mathrm{d}t$
3. Depending on the collision, output success or failure.

The situation is depicted on the right figure.



3.1 Collision detection

For the sake of simplicity, there are only two kinds collisions to consider:

1. with the ground;
2. with the target.

We will assume that the projectile, and the target, are circles of radius R (Optional: you may have two different radii R_p and R_t if you prefer).

Task 8. Assuming that the projectile has position (x, y) and that the ground is the half-plane $y < 0$, write the boolean expression `groundCollision` which is true if and only if there is collision between the projectile and the ground.

Task 9. Assume that the target has position (X, Y). Write the boolean expression `targetCollision` which is true if and only if there is a collision between the projectile and the target.

3.2 Physics simulation

We assume that gravity is downward vertical and constant. According to Physics, this means that during time $\mathrm{d}t$, the projectile's velocity undergoes $v_y \mapsto v_y - g \times \mathrm{d}t$.

Task 10. Express v_x and v_y at $t = 0$, as a function of α and v.

Task 11. During time $\mathrm{d}t$, the projectile moves as follows: $x \mapsto x + v_x \times \mathrm{d}t$, and $y \mapsto y + v_y \times \mathrm{d}t$. Implement these expressions.
Hint: You need to declare the variables `x, y, X, Y, vx, vy, g, t` and `dt`.

Task 12. Assemble the whole algorithm together.
Hint: You can escape a `while` loop using `break`.
Let $\mathrm{d}t = 0.01, g = 9.81, X = 10$, and $Y = 2$. Play!

Remark: You just solved a differential equation (here: $\ddot{y} + g = 0$), using an approach known as Euler's explicit method. More advanced methods exist and appear in simulations of financial and economic models.