3. Cycles

Ján Dugáček

October 9, 2019

Table of Contents

Motivation

- Write a program that reads 10 numbers and outputs their average

Motivation #2

- If you copy and paste something 10 times, editing it afterwards is annoying
- There might be a number that tells how many numbers will there be, allowing large numbers
- We need a better solution

Motivation #3

```
int i = 0;
float sum = 0;
while (i < 10) {
        int got;
        std::cin >> got;
        sum += got;
        i++;
}
std::cout << (sum / 10) << std::endl;
```

- We repeat the same part of the code over and over *while* the condition is met

While loop

```
unsigned long int y = 1;
while (x > 1) {
        y *= x;
        x--;
}
```

- `while` loop is much like `if`, but at the end of the block, it checks for the condition again and if it's still true, the block is executed again, after it is checked again and so it goes forever while the condition is true
- This particular loop computes the factorial of `x` (the result is in variable `y`)

While loop #2

```
unsigned long int y = 1;
while (x > 1) {
        y *= x;
        if (y > 100000000000000) break;
        x--;
}
```

- This is like the previous one, but we check if `y` is greater than a billiard, if it is, the loop is *broken*, immediately interrupted and the execution jumps out of it
- A lesser variant of `break` is `continue`, which skips the rest of the block but goes to check for the condition again
- If there is nothing to stop the cycle from looping, the program will freeze

```
while (1) {}
```

While loop #3

```
unsigned long int y = 1;
do {
        y *= x;
        x--;
} while (x > 1);
```

- This is a variation of the `while` loop that starts executing the block and checks the condition if it's going to continue after
- The only difference is that the block is executed at least once regardless of the condition
- The code above *does not* correctly compute the factorial of `x`, if `x` is 0, the result is 0, which is incorrect because $0! = 1$

For loop

```
int x = 0;
int i = 0;
while (i < 10) {
        x += i;
        i++;
}
```

- Defining a variable, changing it and checking it in a loop is so common that a new construct was created to shorten it:

```
int x = 0;
for (int i = 0; i < 10; i++) {
        x += i;
}
```

- This does the same as the code above

For loop #2

```
int x = 0;
for (int i = 0; i < 10; i++) {
        if (i % 2 == 1) continue;
        x += i;
}
```

- The first part defines a variable (but may also assign a value to an existing variable or may be left blank), the second part is a condition that must be met if it's to be repeated and the last part is a change that happens after every loop (the condition is checked right afterwards, so in the case above, it never enters the block with `i` equal to 10)
- The % sign is modulo, the result of `i % 2` is the remainder of division of `i` by 2
- `continue` makes the loop restart, but `i` is increased before the check (unlike in `while`)
- The code above sums all even numbers between 0 and 10 (including 0, excluding 10)
- The variable change in each iteration of the loop is called *iterator* and usually named `i` (if there are more of them, they are named `i`, `j`, `k`, ...)

Nested loop

```
int x = 0;
for (int i = 1; i <= 10; i++) {
        for (int j = 1; j <= 10; j++) {
                x += i * j;
        }
}
```

- Many applications require using loops within loops
- In this case, we calculate the sum of all multiplications between numbers in range 1 to 10

Exercise

1. Calculate the harmonic average of 10 numbers supplied by the user
2. Same as previous, but keep asking for numbers until the user inputs zero
3. Calculate the product of odd numbers between 23 and a number supplied by the user
4. Calculate $\int_{-2}^{4} sin(x^2)dx$
5. Calculate π by computing the ratio between the integral of a constant and the integral of the circle function $y = \pm\sqrt{1 - x^2}$

- Hint:

```
#include <cmath>
int main(int argc, char** argv) {
        float x = 1;
        float y = sin(x);
        float z = sqrt(x);
```

Advanced exercise

- Find primes lesser than 10000 using Eratostenes' sieve
- Is it faster than trying to divide each number by all primes lesser than its square root?

Homework

- Write a program that approximately computes x that best satisfies the equation $x + 1 = 1/x$
- You have two weeks to do it

Advanced Homework

- Find the values of a, b and c where $f(x) = a \cdot \ln(x \cdot b) - b \cdot \sqrt{200 - x \cdot c} + c$ has the smallest sum of second powers of values of $f(1)$, $f(2)$, ... $f(100)$
- You have two weeks to do it