EXERCISES: WEEK 1

Game programming I - 5SD022

Questions

Q: What fundamental data types are there in C++?

Q: What differentiates these fundamental data types?

Q: What is a variable?

Q: What arithmetic operators does C++ have?

Q: ... comparison operators?

Q: ... logical operators?

Q: ... compound operators?

Q: How does an assignment operator work?

Q: What does operator precedence mean?

Q: What is an boolean expression?

Q: What types of control statements are there in C++?

Q: Give an example of an control statement

Q: What type of loops do we have?

Q: In what way do they differ?

Exercise programs

Program 1
- Write a program that declares a variable of `int`eger type
- Assign the value 5 to the variable
- Output the value with `std::cout`
- Compile and run the program

Program 2
- Extend **Program 1** with another variable of `int`eger type
- Assign the value 3 to the variable
- Output both values on one line
- Compile and run the program

Program 3
- Extend **Program 2**
- Declare a third variable of type `int`eger
- Add the variable from **Program 1** and **Program 2**, and assign it to new variable
- Output the result of the addition
- Compile and run the program

Program 4
- Write a program that declares a variable of `float`ing point type
- Assign a value to the variable
- Output the value to the console
- Compile and run the program

Program 5
- Extend **Program 4**
- Declare a variable of `int`eger type
- Assign the value from the variable in **Program 4** to the new `int`eger variable
- Output the value of the `int`eger
- Compile and run the program

Note: What happens with the `float`ing point variable and it's value when it get's assigned to a `int`eger?

Program 6

- Write a program with two floating point variables
- Assign `0.1f` to the first one
- Assign `0.2f` to the second one
- Output the addition of the two variables
- Compile and run

- Above the output line, write: `std::cout.precision(8);`
- Compile and run
- What happens?
- Why do we get the result?

Program 7

- Write a program with three variables of `float`ing point type
- Assign a `float`ing point value to the two first variables when declaring them
- Assign the arithmetic result of a subtraction to the third variable
- Square the third variable and assign the result back to it
- Output the result
- Compile and run the program

Program 8

- Write a program that outputs the result of: (1 / 2)
- Compile and run
- What happens?

- Change the division to: (`1.0f` / 2)
- Compile and run
- What happens?

- Change the division to: (1 / `2.0f`)
- Compile and run
- What happens?

Program 9

- Write a program that calculates a `float`ing point value from Fahrenheit to Celsius
- Compile and run

Program 10

- Write a program that lets the user input an `integer` value
- Output the value of the `integer` variable squared
- Compile and run

- Run the program again, now enter a `floating` point value
- What happens?

- Run the program yet again, now place your right index finger on a random key
- Press that key and move the finger over the keyboard in random directions
- Press enter
- What happens?

Program 11

- Extend **Program 10**
- Let the user input two `integer` values
- Square the first variable and divide by the second
- Assign the result to a `floating` point variable
- Output the result
- Compile and run

Program 12

- Write a program that converts user input from Celsius to Fahrenheit
- Compile and run

Program 13

- Write the program below, compile and run:

```
#include <iostream>

int main(int argc, char* argv[]) {
      std::cout << "sizeof(long long)  : " << sizeof(long long) << std::endl;
      std::cout << "sizeof(int)        : " << sizeof(int) << std::endl;
      std::cout << "sizeof(short)      : " << sizeof(short) << std::endl;
      std::cout << "sizeof(char)       : " << sizeof(char) << std::endl;
      std::cout << "sizeof(float)      : " << sizeof(float) << std::endl;
      std::cout << "sizeof(double)     : " << sizeof(double) << std::endl;

      return 0;
}
```

- What does it say, and why does it output the values for the different types?

Program 14
- Write the following program:

```
#include <iostream>

int main(int argc, char* argv[]) {
      int X = 0, Y = 3;
      X = ++Y + 1;
      std::cout << “X: “ << X << std::endl;

      return 0;
}
```

- What value does X have and why?

Program 15
- Write a program that has two variables
- Ask the user for values and store these in the variables
- Output the largest value of the two

Program 16
- Extend **Program 15**
- Output the smallest value of the two

Program 17
- Extend **Program 15**
- Output something telling the user that they are equal

Program 18
- Write a program that gives the following output
- Let the user decide the number of bottles

```
“... ”
“4 bottles of juice on the wall, one fell down”
“3 bottles of juice on the wall, one fell down”
“2 bottles of juice on the wall, one fell down”
“1 bottles of juice on the wall, one fell down”
“0 bottles of juice on the wall, one fell down”
“No bottles left on the wall!”
```

Program 19

- Write the following program

```
#include <iostream>

int main(int argc, char* argv[]) {
	int value = 0;
	std::cout << “Give me a number: ”;
	std::cin >> value;

	if(value > 0) {
		int value = 10;
		value *= 2;
	}
	else if(value == 0) {
		std::cout << “Not a valid number!” << std::endl;
	}
	else {
		std::cout << “Value: “ << value << std::endl;
	}
	return 0;
}
```

- What does the program output?
- Describe what happens

Program 20

- Write a for loop that goes from 0 - 10 and output the number at every iteration

Program 21

- Write a reversed version of **Program 20**

Program 22

- Write a loop that goes from 0 to 100
- Output all numbers divisible by 3

Program 23

- Write a program that loops from 0 to 7 and the back to 0
- When it reaches the largest number output “Going down.”
- Output the value every iteration

Program 24
- Write a program that calculates the N!

Program 25
- Write a program that converts from m/s to km/h and mi/h
- Output both results

Program 26
- Write a program that simulates the guessing game
- The magic number to be guessed should be random at every run of the program
- Let the player input an `integer` value between 1 and 100
- If the number is lower than the magic number, output that to the player
- If the number is higher than the magic number, output that to the player
- If the number is correct, end the game and congratulate the player
- Let the player guess until they guess the correct magic number
- When the player guesses right, the number of tries should be outputted as well

Numeral Systems

Base-2 Numeral System

Because of the components computers are built out of (transistors), they internally use a base-2 numeral system or *binary number system* where every position is represented by either a 0 (zero) or a 1 (one). These ones and zeroes are called *bits*.

Every natural number in the more human readable format, base-10, can be represented as a sequence of bits, e.g 9 in base-10 can be written as 1001 as base-2.

If we have 1101_2 (subscript 2), to convert it to base-10 we take every position of the binary number and take two to the power of N, where N represents the position, right to left, starting at 0.

Looking at 1101_2, converting it to base-10, we take:

$$
\begin{aligned}
&1101_2 \Rightarrow \\
&1 * 2^3 + 1 * 2^2 + 0 * 2^1 + 1 * 2^0 \Rightarrow \\
&1 * 8 \ + 1 * 4 \ \ + 0 \ * 2 \ + 1 * 1 \Rightarrow \\
&8 + 4 + 1 = 13_{10}
\end{aligned}
$$

A more lengthy example $1010\ \ 0010_2$:

$$
\begin{aligned}
&1 * 2^7 + 0 * 2^6 + 1 * 2^5 + 0 * 2^4 + 0 * 2^3 + 0 * 2^2 + 1 * 2^1 + 0 * 2^0 \ \Rightarrow \\
&128 + 32 + 2 \ = \ 162_{10}
\end{aligned}
$$

Base-16 Numeral System

Computers nowadays use modern hardware that can handle 32-bit and 64-bit natural numbers with ease, using base-2 when communicating instructions and data to it's hardware becomes tedious. Instead we use base-16 numeral systems or simply *hexadecimal*.

Using hexadecimal works well with binary systems when grouping four binary digits together, starting from the lowest digit; four binary digits represents values from 0_{16} to F_{16}.

Having a binary number with 5 digits: $1\ \ 0010_2$, padding three 0:s can make grouping easier: $\underline{000}1\ \ 0010_2$, converting it to hexadecimal gives: 12_{16}.

Convert Exercises

E: Convert 1011_2 to decimal.

E: Convert 110_2 to decimal.

E: Convert 1100_2 to decimal.

E: Convert $10\ 0110_2$ to decimal.

E: Convert $1110\ 0111_2$ to decimal.

E: Convert $101\ 1111\ 0110_2$ to decimal.

E: Convert $111\ 0110\ 1100\ 1001_2$ to decimal.

E: Convert 5_{10} to binary.

E: Convert 15_{10} to binary.

E: Convert 28_{10} to binary.

E: Convert 255_{10} to binary.

E: Convert 305_{10} to binary.

E: Convert 1023_{10} to binary.

E: Convert 2_{16} to binary.

E: Convert 9_{16} to binary.

E: Convert 12_{16} to binary.

E: Convert 17_{16} to binary.

E: Convert $A4_{16}$ to binary.

E: Convert 112_{16} to binary.

E: Convert $7FF_{16}$ to binary.

E: Convert $DEAD_{16}$ to binary.

E: Convert 1001_2 to hexadecimal.

E: Convert 1011_2 to hexadecimal.

E: Convert $0101\ \ 1100_2$ to hexadecimal.

E: Convert $0111\ \ 0101_2$ to hexadecimal.

E: Convert $0111\ \ 0101_2$ to hexadecimal.

E: Convert 3_{10} to hexadecimal.

E: Convert 10_{10} to hexadecimal.

E: Convert 16_{10} to hexadecimal.

E: Convert 21_{10} to hexadecimal.

E: Convert 30_{10} to hexadecimal.

E: Convert 33_{10} to hexadecimal.

Pseudo Random Number Generator

```
#include <iostream>
#include <time.h>
#include <stdlib.h>

int random(int min, int max)
{
	return min + (rand() % (max - min + 1));
}

int main(int argc, char* argv[])
{
	srand((unsigned int)time(0));

	std::cout << "random(): " << random(0, 10) << std::endl;

	return 0;
}
```