Programming Paradigms, C^{++} Exercise 1

Deadline: 27/29.08.2018

1. Type `which g++` to find out in which directory the C^{++} compiler is situated.

   Change the current directory to this directory. Show that you are able to type `ls -t`, `ls -l`, `ls -lh`, `ls -l | more`, `ls -lt | less` and understand what they do.

2. In your home directory, create a directory `hello`, and show that you are able to find it. In this directory, create a file `hello.cpp` with the following contents:

   ```
   #include <iostream>

   int main( int argc, char* argv [ ] )
   {
      std::cout << "hello world!\n";
      return 0;
   }
   ```

3. Create a `Makefile` with the following contents:

   ```
   hello: hello.cpp
         g++ -o hello hello.cpp -std=c++11
   ```

   Use the `make` command to compile the **hello world** program. It is essential that the before `g++`, there is a TAB (not spaces), because `make` distinguishes between TAB and spaces. In order to test the Makefile, type

   ```
   make              // Program is compiled
   make              // Program is not compiled a second time.
   touch hello.cpp
   make              // Program is compiled again.
   ./hello           // Program is executed.
   ```

4. Make a copy of the previous program, which must be called `hello2.cpp`. Modify it into

```
#include <iostream>
#include <string>

int main( int argc, char* argv [ ] )
{
   std::cout << "what is your name? \n";
   std::string name;
   std::cin >> name;
   std::cout << "hello, " << name << "!\n";
   return 0;
}
```

Add corresponding lines to the **Makefile**. In order to compile it, type `make hello2`.

5. Now we can also compile the program that demonstrates that C^{++} has *value semantics*. (Assignment/parameter passing is *copying*, values with distinct names are *distinct*.)

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

int main( int argc, char *argv[] )
{
   std::vector<int> x(10);
      // Creates vector<int> (dynamic array) with 10 zeroes.

   // size_t is an integer >= 0 that is guaranteed to be big
   // enough for every index. Now, and always in the future.

   for( size_t i = 0; i < x. size( ); ++ i )
       x[i] = i;

   std::vector<int> y = x;
   y[5] = 12;

   for( size_t i = 0; i < x. size( ); ++ i )
      std::cout << i << " : " << x[i] << " " << y[i] << "\n";
   return 0;
}
```

Add commands to the **Makefile**, compile it, and observe that `std::vector` has value semantics.