C++ from beginner and beyond

# Section 4: getting started

Compile errors

Compiler errors: Errors that are found/happen at the compile stage

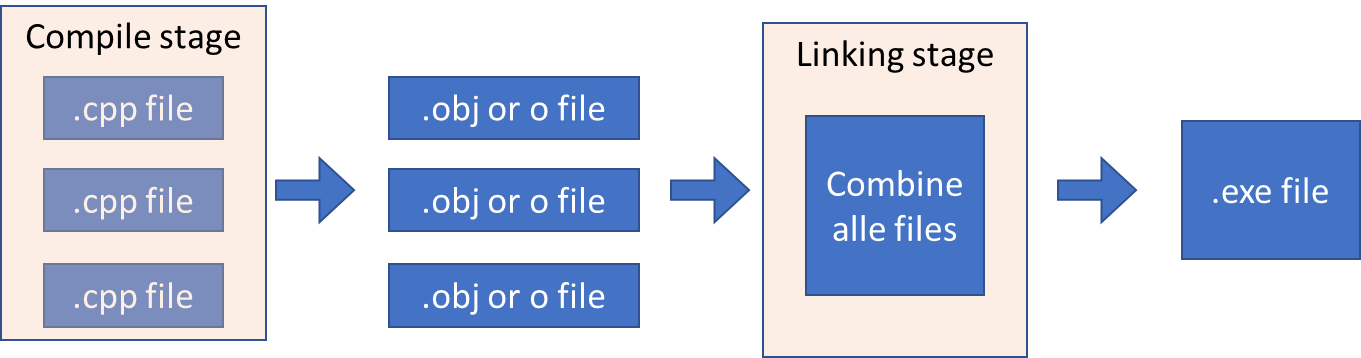
* Syntax errors -> errors related to the structure of the code (no semi-colon, or parentheses)
* Semantic errors -> something wrong with the meaning of your code (division by 0, adding an int and string)

When debugging in C++ start at the first error and work your way down. Often, the first error is the route cause of some of the other errors. Also, do not ignore compiler warnings. Warnings help you prevent unexpected behavior.

Link errors

Linker errors are error that occur in the linking stage are caused by the ‘builder’ not knowing where some part of code comes from... This happens when are files are missing or in the wrong places.

From file to executable



Run time errors

Errors that occur when the program is executing.

For example: divide by zero, file not found or out of memory errors. Also, the errors often make your program crash and we can deal with these times of errors by using exception handling.

Logic errors

Logic errors (often referred to as ‘Bugs’) cause your program to run incorrectly and are mistakes made by the programmer.

Sections challenge

|  |
| --- |
| int main()  {  std::cout << "Enter your favourite number between 1 and 100: ";  int favnr = { 0 };  std::cin >> favnr;  std::cout << "Amazing!! That's my favourite number too! \n";  std::cout << "No really, " << favnr << " is my favourite number! \n";  return 0;  } |

# Section 5: Structure of a C++ program

C++ key words: <https://en.cppreference.com/w/cpp/keyword>

**An overview of the C++ structure**

Identifiers: something that the programmer names

Operators:

=,/,\*

>> 🡪 the insertion operator

<< 🡪 the extraction operator

:: 🡪 scope resolution operator

**C++ preprocessor**

The preprocessor processes your code before the compiler. It removes commend and performs certain statements such as the ‘include’ statement. Finally, the c++ preprocessor does not understand C++, it simply follows the preprocessor directives and prepares the code for the compiler.

**Comments**

Work the same as any other programming language. There are however two types:

* Single line commends //
* Multi line commend /\* \*/

Don’t state the obvious in your commends.

**Main function**

Every C++ program has to have a main function. The main function forms the starting point of your program

**Namespaces**

Name spaces help programmers to scope their code. This is particularly useful when you have different objects that have similar function names (as this avoids naming conflicts).

We call a name space by using the :: operator.

Basic input and output using cin and cout

* Cout outputs to the console or user interface
* Cin takes outputs from the console or user interface
* Cerr
* Clog

New lines are not automatically added so we need to use endl or \n

When using Cin in put gets pulled from a so-called buffer. This means that if you enter multiple input items as input and the de compiler expects multiple items, these items are automatically read from the buffer.

# Section 6: Variables and Constants

A variable is an abstraction for a memory location and allows you to use meaningful names instead of memory addresses.

Variables must be declared before they are used, however its value may change trough out the program

Variables can be initialized in the following ways:

* Int age {0};
* Int age = 0;
* Int age (0);

**Global variables**

Global variables are accessible throughout the code. They allow the programmer to initiate variables one and access them in different functions.

It is important to note that global variables are overwritten/ shadowed by local variables when the variable names are the same.

**C++ built-in primitive types**

* Char 8 bits
  + Char16\_t char32\_t at least 16 or 32 bits respectively
  + Initialize with singe quotes ‘
* Int 16 bits
  + Short 16 bits
  + Long 32 bits
  + Long long 64 bits
  + We can use the unsigned keyword to create positive or negative
* Float 7 decimal
  + Double 15 decimal
  + Long double 19 decimal
* Bool 8 bits
  + Only true or false (1/0)

**Size of variable**

You can return the size of a variable or item using the sizeof() function.

You can also use the following parameters to get the max or min value of an object type:

* CHAR\_ MIN MAX
* INT\_ MIN MAX
* SHRT\_ MIN MAX
* LONG\_ MIN MAX
* LLONG\_ MIN MAX

**What is a constant?**

The value of a constant cannot be changed once declared.