Script for the vhb lecture

Programming in C++
Part 1

Prof. Dr. Herbert Fischer
Deggendorf Institute of Technology

Table of Contents

Primary literature:

Herrmann Dietmar
Grundkurs C++ in Beispielen, vieweg Verlag, 6.Auflage, 2010
ISBN: 3-8266-0910-7

Louis Dirk
C++, Hanser Verlag, 1. Auflage, 2014
ISBN: 3-446-44069-2

Kirch-Prinz Ulla, Kirch Peter
C++ Lernen und professionell anwenden, mitp-Verlag, 5.Auflage, 2010
ISBN: 3-89842-171-6

Arnold Willemer
Einstieg in C++, Galileo Computing, 4.Auflage, 2009
ISBN: 3-8362-1385-0

Secondary literature:

Helmut Balzert
Lehrbuch der Softwaretechnik, Spektrum Akademischer Verlag, 2.Auflage, 2000
ISBN: 3-8274-0480-0

Peter P. Bothner
Ohne C zu C++, Vieweg, 1.Auflage, 2001
ISBN: 3-528-05780-7

Ernst-Erich Doberkat
Das siebte Buch: Objektorientierung mit C++, B.G. Teubner Stuttgart · Leipzig · Wiesbaden, 2000
ISBN: 3-519-02649-X

John R. Hubbard
C++- Programmierung, mitp, 1.Auflage, 2003
ISBN: 3-8266-0910-7

Dietrich May
Grundkurs Softwareentwicklung mit C++, vieweg, 2.Auflage, 2006
ISBN: 3-8348-0125-9

Tools:

Code::Blocks for Windows, Linux, Mac OS (free Software): **http://codeblocks.org/downloads/26**

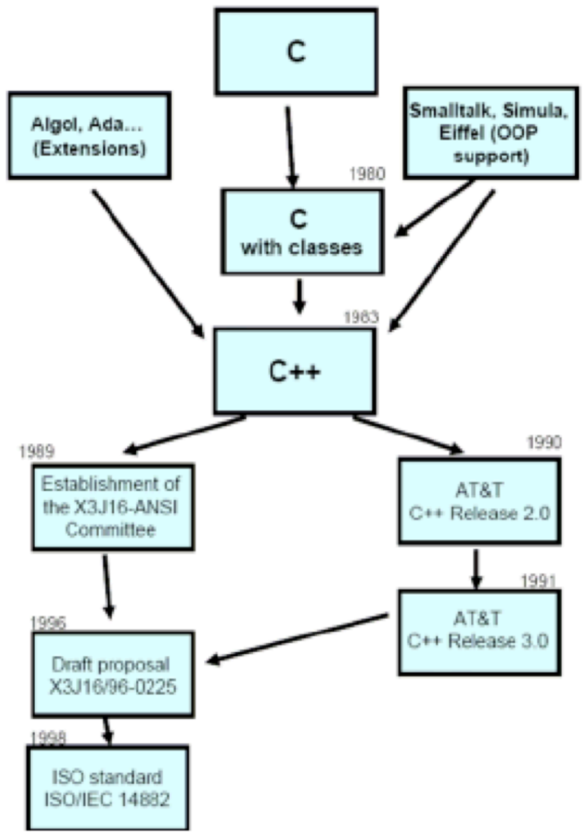
Alternatives:
CodeLite for Windows, Linux, Mac OS: **http://downloads.codelite.org/**
KDevelop for Windows, Linux: **https://www.kdevelop.org/download**
Dev-C++ for Windows: **http://www.bloodshed.net/dev/**
XCode for Mac OS: **https://itunes.apple.com/de/app/xcode/id497799835**

1 Introduction to object-oriented programming: C++

Chapter 1 gives you a short introduction to C++ programming, and you will create your first C++ program.

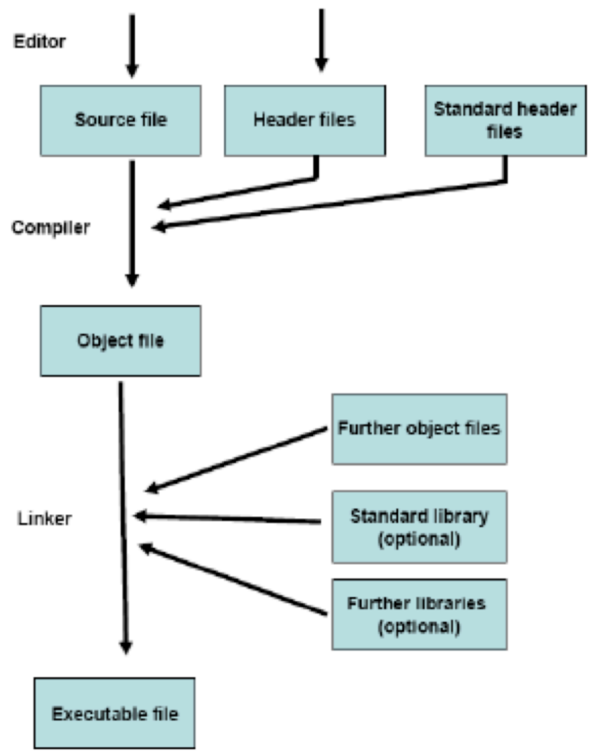
C++ evolved from another programming language called C, a programming language often referred to as “C with classes”. In today's C++ programs the relationship to C can still be recognized clearly. C++ was not developed to replace C, but to improve it.

1.1 Development of C++


C
Algol, Ada...
(Extensions)
Smalltalk, Simula, Eiffel (OOP support)
1980
C with classes
1983
C++
1989
Establishment of the X3J16-ANSI Committee
1990
AT&T C++ Release 2.0
1991
AT&T C++ Release 3.0
1996
Draft proposal X3J16/96-0225
1998
ISO standard ISO/IEC 14882

C++ was developed at Bell Laboratories (Murray Hill, USA) by Bjarne Stroustrup to implement simulation projects with minimal memory and time requirements. Early versions of the programming language, initially called "C with classes", have been in use since 1980. The name C++ is credited to Rick Mascitti (1983). He points out that the name C++ signifies the evolutionary nature of the changes from C: "++" is the C increment operator.
C was chosen as the basis for C++ because of its efficiency and portability. During the further development of C++, the compatibility to C was always taken into account. This means that comprehensive software written in C remains usable in C++ programs. These include, for example, tools and libraries for graphics systems or database applications.
The SIMULA67 programming language had a decisive influence on the implementation of object-oriented concepts, in particular in the creation of classes, inheritance and the design of virtual functions. Operator overloading and the possibility of freely placing declarations in the program code was borrowed from the ALGOL68 programming language. Two programming languages, Ada and Clu, have influenced the implementation of templates and exception handling.
After all, many developments dating back to the years 1987 to 1991 can be traced back to the experiences and problems of C++ programmers. This includes, for example, multiple inheritance, the concept of pure virtual functions and the use of shared memory for objects.

1.1.1 Making a C++ program executable



To create and execute a C++ program, basically the same steps are necessary as in C:

- The program is created using an editor.
- The program is compiled, which means it is translated into the computer's machine language.
- The linker finally creates the executable file.

1.1.2 Editor

An editor is used to create the text files that contain the C++ code. There are two different types of files:

- Source files

Source files contain definitions of global variables and functions. Each C++ program consists of at least one source file.

- Header files

Header files, also called include files, provide the information needed in various source files. This includes:

  - Type definitions, e.g. class definitions
  - Declarations of global variables and functions
  - Definitions of macros and inline functions

The correct extension must be used when naming the files. However, these vary from compiler to compiler: The most common extensions for source files are .cpp and .cc. The names of header files either end - as in C - on .h or they do not have an extension at all. However, extensions like .hpp may also occur. Header files of the C standard library can of course still be used.

1.1.3 Compilers

A translation unit consists of a source file and the included header files. The compiler generates an object file (also called module) from each translation unit; the object file contains the machine code. In addition to the compilers that generate the machine code directly, there are also C++ into C translation programs, so-called »C-Front-Compilers«. They translate a C++ program into a C program. Only then is the object file created with a standard C compiler.

1.1.4 Linker

The linker combines object files into an executable file. Besides the self-generated object files, it also contains the startup-code and the modules with the used functions and classes of the standard library.

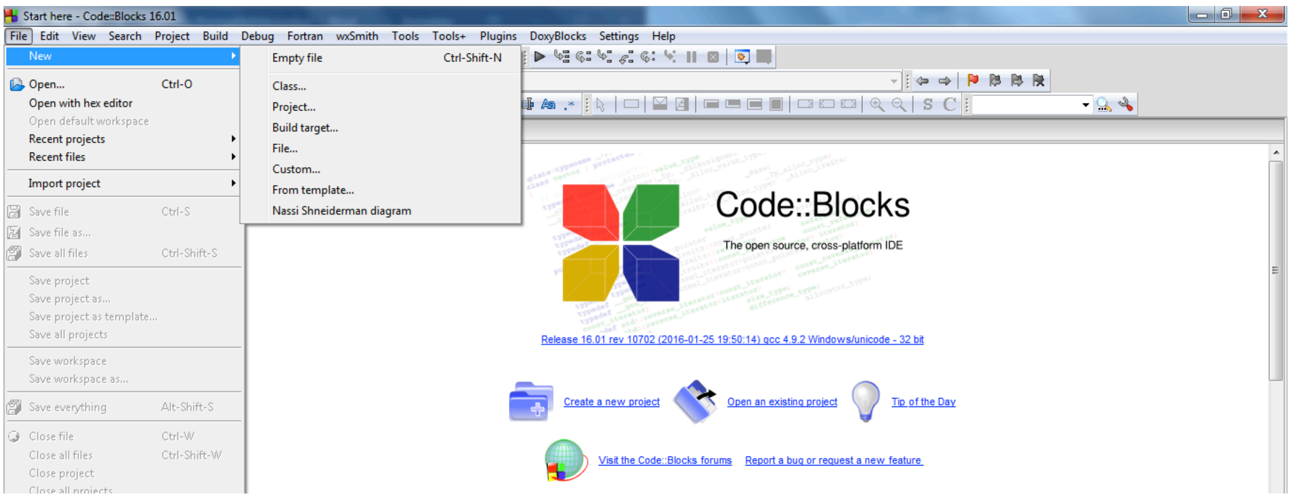
1.2 Introduction to the programming environment: C++

After this short introduction, we now want to create our first C++ program. We will create a Win32 console application in this course. These are programs which run in a DOS window. We use Code::Blocks as development environment. However, you can also use any other C++ development tool (e.g. Dev-C++, CodeLite, XCodes, MS Visual Studio, etc.).
You can download Code::Blocks from **http://codeblocks.org/downloads/26**.
A video tutorial on Code::Blocks can be found on the homepage of this course.

Creating a console application with Code::Blocks

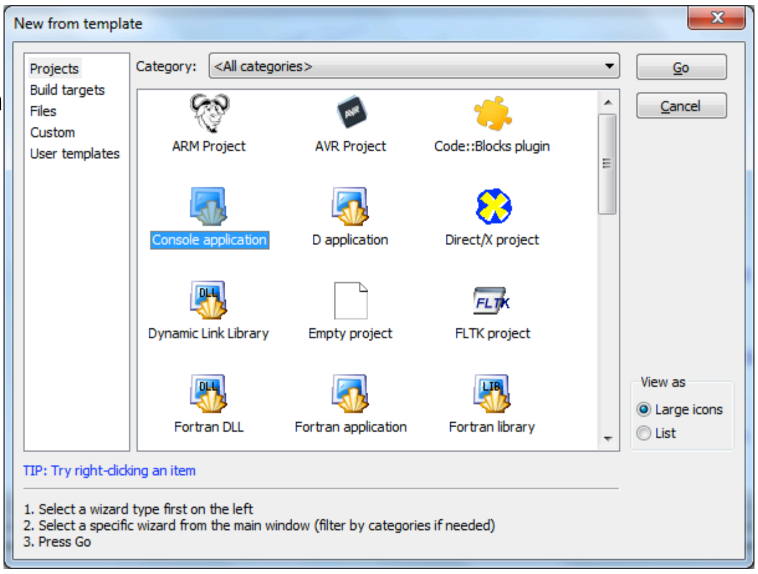
Step 1

Start the Code::Blocks development environment. Select *New* from the *File* menu and then select *Project* or click on *Create a new project* in the info area.


Start here - Code::Blocks 16.01
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
New
Open... Ctrl-O
Open with hex editor
Open default workspace
Recent projects
Recent files
Import project
Save file Ctrl-S
Save file as...
Save all files Ctrl-Shift-S
Save project
Save project as...
Save project as template...
Save all projects
Save workspace
Save workspace as...
Save everything Alt-Shift-S
Close file Ctrl-W
Close all files Ctrl-Shift-W
Close project
Empty file Ctrl-Shift-N
Class...
Project...
Build target...
File...
Custom...
From template...
Nassi Shneiderman diagram
Code::Blocks
The open source, cross-platform IDE
Release 16.01 rev 10702 (2016-01-25 19:50:14) gcc 4.9.2 Windows/unicode - 32 bit
Create a new project
Open an existing project
Tip of the Day
Visit the Code::Blocks forums
Report a bug or request a new feature

Step 2

In the following dialog box, choose the *Console application* icon. Then click on *Go*. Select C++ as language, then assign an arbitrary project title to your file and choose a location where you want to save your project. Finally check if the compiler selected is the *Gnu GCC Compiler*, and whether the two check-boxes for *Debug* und *Release* have been selected. At last, click on *Finish*.



1. Select a wizard type first on the left
2. Select a specific wizard from the main window (filter by categories if needed)
3. Press Go

Step 3

Click on *Sources* in the navigation bar (on the left), then double-click *main.cpp*. This is our main program. Code::Blocks has already generated a code skeleton that can be used further on:

```
#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[ ])
{
        cout << “Hello world!” << endl;
        system(“pause”);
        return 0;
}
```

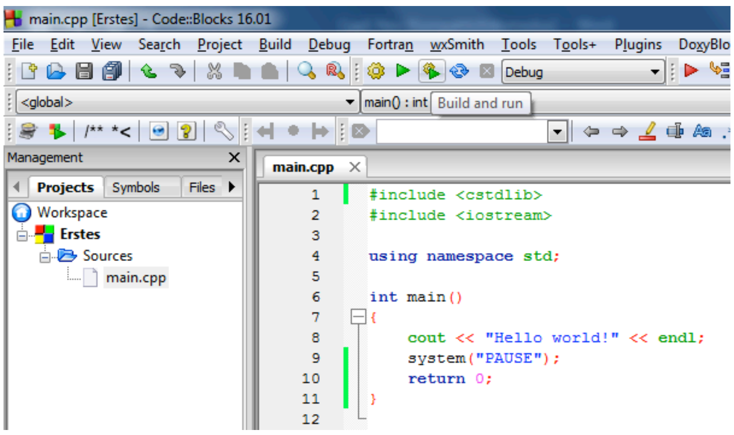
By clicking on the appropriate icon, the compiler creates an executable program. That means that in this case a DOS window will open, which generates the output "Hello world!

You can choose one of the following symbols:

Build:	Compiles the program and generates an executable application.
Run:	Executes the program compiled last.
Build & Run:	Compiles the program and then executes it.

Step 4

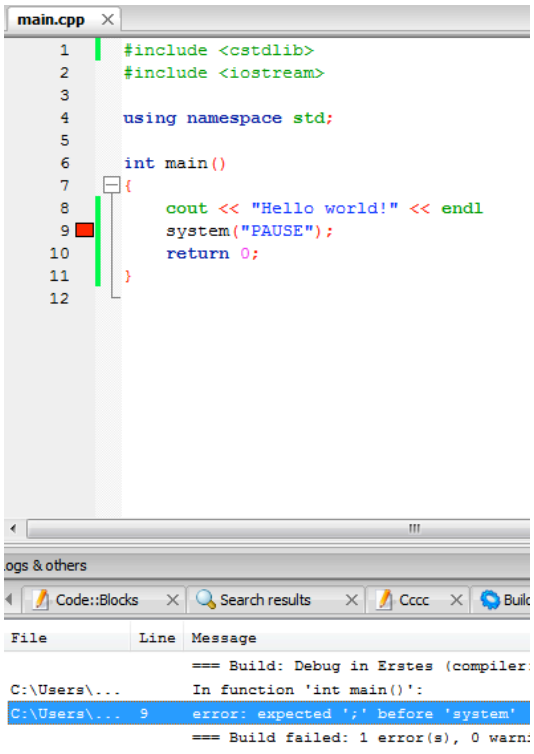
Click on “Build & Run“ to compile and execute the program.


main.cpp [Erstes] - Code::Blocks 16.01
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlo
Debug
<global>
main() : int
Build and run
Management
Projects Symbols Files
Workspace
Erstes
Sources
main.cpp

```
#include <cstdlib>
#include <iostream>

using namespace std;

int main()
{
    cout << "Hello world!" << endl;
    system("PAUSE");
    return 0;
}
```

Step 5

If the program contains errors, the line that contains the error will be marked with a red rectangle (on the left). In addition, an error message will be displayed below. Correct the error(s) - in this case the missing semicolon- and compile again. Then save the file.

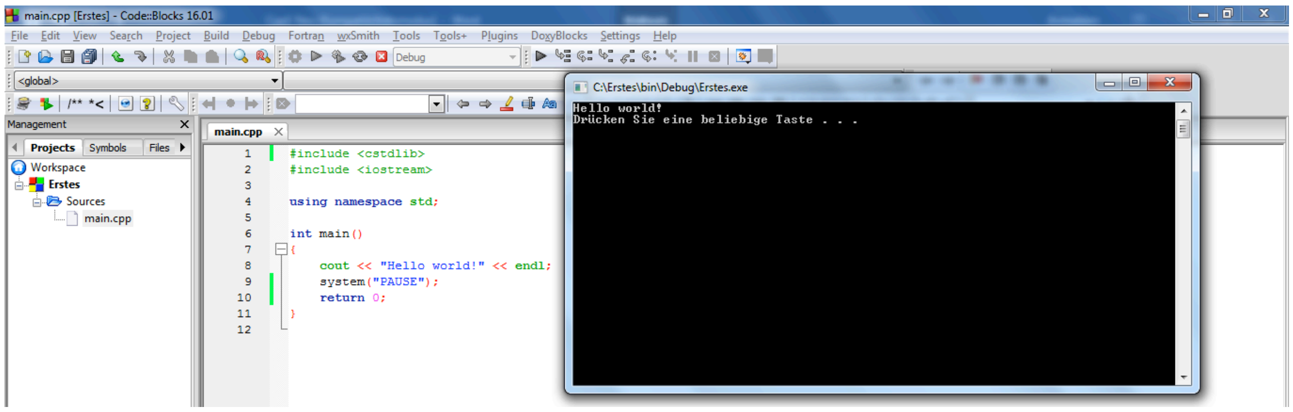


```
#include <cstdlib>
#include <iostream>

using namespace std;

int main()
{
    cout << "Hello world!" << endl
    system("PAUSE");
    return 0;
}
```

```
File            Line  Message
                      === Build: Debug in Erstes (compiler:
C:\Users\...          In function 'int main()':
C:\Users\...    9     error: expected ';' before 'system'
                      === Build failed: 1 error(s), 0 warn:
```

Step 6

Now click on “Run“ and the program will be executed in a DOS-window.

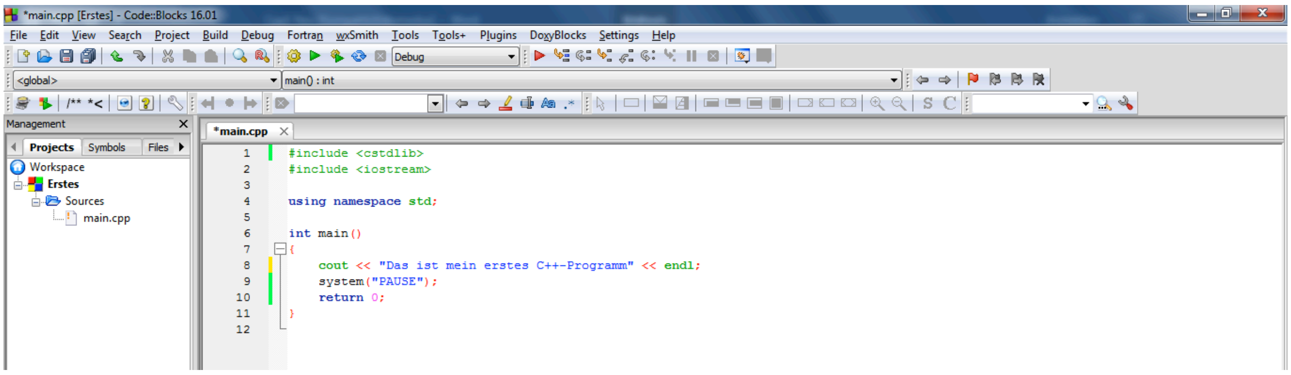

main.cpp [Erstes] - Code::Blocks 16.01
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Debug
<global>
Management
Projects Symbols Files
Workspace
Erstes
Sources
main.cpp
C:\Erstes\bin\Debug\Erstes.exe

```
#include <cstdlib>
#include <iostream>

using namespace std;

int main()
{
    cout << "Hello world!" << endl;
    system("PAUSE");
    return 0;
}
```

```
Hello world!
Drücken Sie eine beliebige Taste . . .
```

Step 7

Save the file (*Save All files*) and close the program.


*main.cpp [Erstes] - Code::Blocks 16.01
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Debug
<global>
main() : int
Management
Projects Symbols Files
Workspace
Erstes
Sources
main.cpp
*main.cpp

```
#include <cstdlib>
#include <iostream>

using namespace std;

int main()
{
    cout << "Das ist mein erstes C++-Programm" << endl;
    system("PAUSE");
    return 0;
}
```

Now you have probably implemented your first C++ program successfully.

Congratulations!

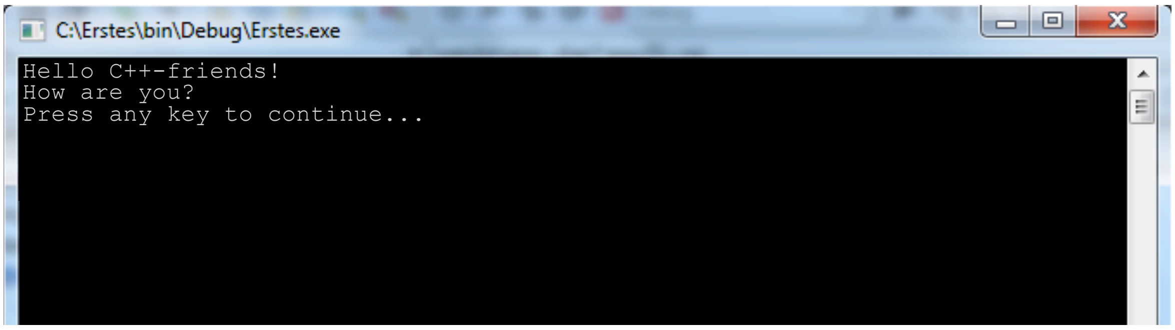
1.2.1 Simple output program

Let's have a look at out program again. Now insert the following lines:

```
cout << "Hello C++-friends!";
cout << endl;
cout << "How are you?" << endl;
```

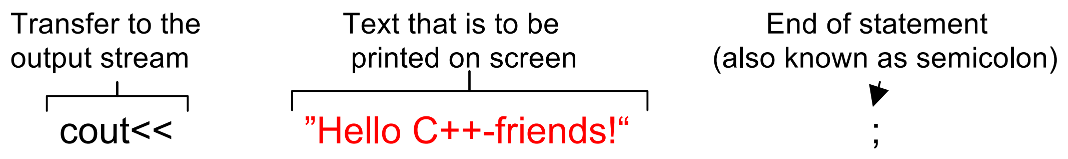
```
#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[ ])
{
        cout << "Hello C++-friends!";
        cout << endl;
        cout << "How are you?" << endl;

        system("pause");
        return 0;
}
```

If you now click on “Build & Run“, the following output will appear in the console window:



```
Hello C++-friends!
How are you?
Press any key to continue...
```

To print text on the screen, we need a C++ class called *iostream*. A brief description of this class would therefore be appropriate here. You probably do not know what classes are, but there is no need to worry about that. The *iostream* class uses *streams* to perform basic input and output operations – for example, printing text on the screen or reading user input operations.


Transfer to the output stream
Text that is to be printed on screen
End of statement (also known as semicolon)
cout<<
"Hello C++-friends!"
;

To access the standard C++ output stream, use the keyword *cout*. For console applications, the standard output by default is the console or the screen. The *iostream* class uses special operators to insert the data into the stream. The << transfer operator inserts the data that follows into the stream. In order to print text on the console, for example, you would enter the following: cout << “Do something!”;
With this, you tell your program to insert the text “Do something” into the standard output stream. Please make sure that the text is enclosed in quotation marks and that the code line ends with a semicolon. When the program line is executed, the text will appear on your screen.

Please note: cout is only used in console applications.

1.2.3 Header files

Before you can use cout, you must first tell the compiler where to find the description (called *declaration*) of the iostream class in which cout is located. The *iostream* class is declared in the IOSTREAM file. This file is also called *header file*.
Use the *#include* directive to tell the compiler that it needs to search for the iostream class declaration in IOSTREAM: The *#include* directive includes frequently used functions into the source code and makes them usable within the program.

#include <iostream>

If you forgot to include the corresponding header file of a class or function into your program, you will be notified of a compiler error. The compiler error message could, for example, be as follows: *Undefined symbol 'cout'.*
If you get this message, you should immediately check whether you have included all the headers required for your program.

using namespace std;

Specifies the namespace in which all library elements are declared in C++.

Note:
The first line *#include <cstdlib>* is only used to include system statements like system("pause"); and system("cls“); and can be omitted if these statements are not used.

1.2.4 Libraries

In C/C++ there are libraries that contain different classes and functions. The function libraries partly originate from the programming language C. The big advantage of C and C++ is the standardization of the libraries. The C++ standard library provides the following components as an extensible framework: strings, containers, algorithms, complex numbers, input/output and much more. The namespace *std* contains all data types, classes and functions.

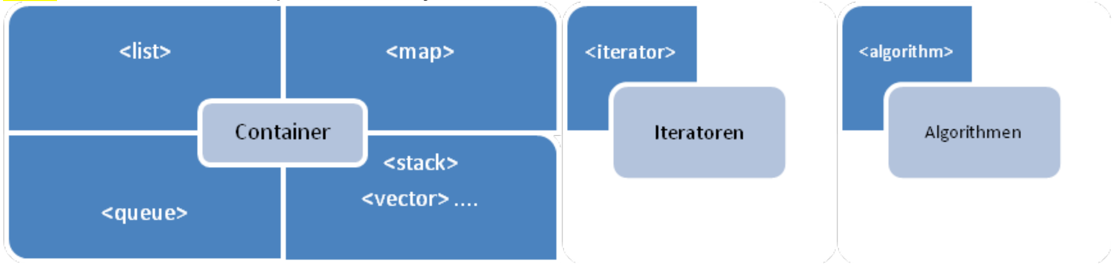
The functions from the library files required for the program are already bound to it during the linking phase. The use of libraries is done in two steps. First, the header file must be included into the source code file using the *#include* pre-processor command. In the second step, the linker gets to know the actual library files, for example <*string*>.

Examples of the most important libraries in C++:

<iostream>	Input/Output
<string>	Strings of characters
<cstdlib>	Help functions
<cmath>	Mathematical functions
<ctime>	Date und time
<random>	Random numbers

A list of other libraries can be found on the following website:
http://www.cplusplus.com/reference/

STL – Standard Template Library



STL stands for **S**tandard **T**emplate **L**ibrary. STL is a collection of template classes and provides a container. A container combines data of the same data type in a particular structure. The simplest form of a container is an array (see **Section 4.1**). The STL provides functionalities for these containers, such as search and sort functions or insert and delete operations.

1.2.5 endl;

The iostream class contains special manipulators that can be used to control the streams. The only manipulator we want to work with at the moment is *endl;* (end line). It is used to insert a new line into the output stream. We us *endl* to insert a new line after we have printed text on the screen. Please note that the last character of *endl* is an l and not a 1.
endl can be appended to the end of a cout statement, or, with cout, in a separate line. You can also write several *endl* one after the other in order to output several blank lines. As an alternative, you can also write \n into the string to create a new line.

```
        cout << ”Hello C++-friends!“ << endl;

or:     cout << ”Hello C++- friends!“;
        cout << endl;
or:     cout<< ”Hello C++- friends!\n“;
```

1.2.6 main function

The *main* function (main program) is where a program starts its execution. After processing all statements within the curly brackets, the program is terminated.

The skeleton program in Code::Blocks is as follows:

```
int main(int argc, char *argv[ ])
```

However, it is sufficient to write:

```
int main()
```

1.2.7 Brackets und whitespace characters

The curly brackets in the program are noticeable. In C++, a code block begins with an opening curly bracket { and ends with a closing curly bracket }. The brackets are used to mark the beginnings and the ends of code blocks of loops, functions, if statements, and so on. Our program contains one set of brackets only, as it is a very simple program.

In C++, whitespace characters are simply ignored. In most cases it is completely irrelevant where you insert blanks or new lines. You can of course not use whitespaces within keywords or variable names, but otherwise you are completely free to use them.

For example, the following source codes are fully equivalent:

```
int main()
{
        cout << "Hello World!";
}
```

equals

```
int main(){cout<<"Hello World!";}
```

1.2.8 Comments

```
#include <iostream>
#include <cstdlib>

using namespace std;       // This is a comment

int main()
{
        cout << ”Hallo C++ friends!“;   // This is another comment
        system(“pause”);
}
```

After the characters //, you can enter single-line comments into your source code. Comment lines are used to document your program.

Multi-line comments can also be written as follows:

```
/*   Comment
          .....
*/
```

1.2.9 system("pause");

The C library provides a system function called *system("pause");*. It is used to make the screen or console wait for a key press. After the keyboard input, the console window closes. This function varies depending on the compiler. The Visual C++ compiler allows to omit *system("pause");*. Return EXIT_SUCCESS; can also be omitted or replaced by return 0;.

Task: Write a program that prints your name and address on the screen as follows:

Moritz Mustermann
Am Stadtplatz 1

94469 Deggendorf

The window is supposed to close after pressing a key. Comment on each line.

Solution:

```
#include <iostream>                  // Includes iostream header file
#include <cstdlib>                   // Includes cstdlib header file
using namespace std;                 // Namespace

int main()                           // Namespace
{                                    // Beginning of the program
   cout<<"Moritz Mustermann"<<endl;        // Prints out text on screen with a line break
   cout<<"Am Stadtplatz 1"<<endl;          // Prints out text on screen with a line break
   cout<<endl;                             // Blank line
   cout<<"94469 Deggendorf"<<endl;         // Prints out text on screen with a line break

system("pause");                     // Closes window after keyboard input
}                                    // End of the program
```

1.2.10 system("cls");

The *system("cls");* command fully deletes the screen output. The cursor is then located at the top left at the beginning of the command line.