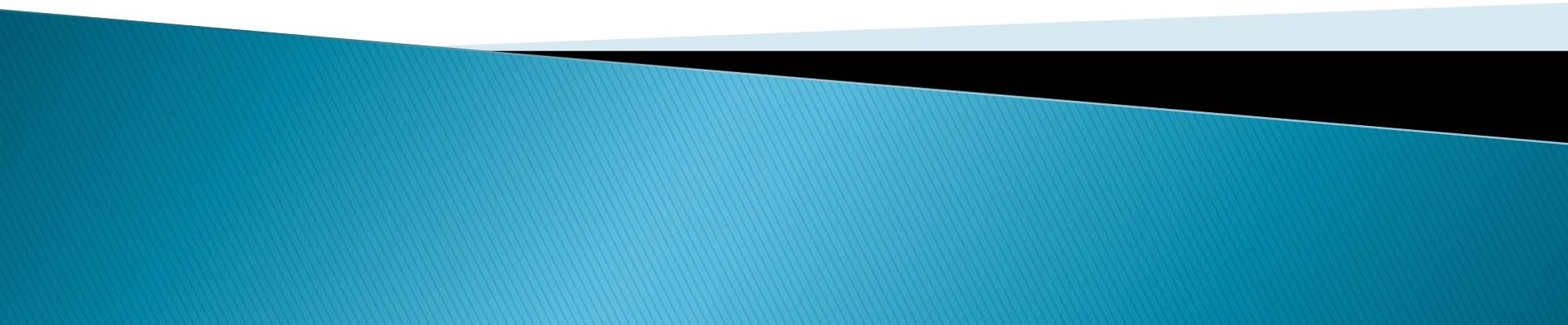


# Programming languages

## WSTI

Types, operators, expressions



# The appearance of the environment

The screenshot shows the Code::Blocks IDE interface. At the top is a menu bar with options: File, Edit, View, Search, Project, Build, Debug, wxSmith, Tools, Plugins, Settings, Help. Below the menu bar is a toolbar with various icons. A red box labeled "Menu/ Toolbar" points to the menu bar and toolbar area. Below the toolbar is a "Build target:" dropdown menu. A red box labeled "Compilation buttons" points to a set of icons including a play button, a refresh button, and a stop button. On the left side, there is a "Management" panel with tabs for "Projects", "Symbols", and "Resol". A red box labeled "Shows open projects and a list of used symbols" points to this panel. The main workspace area contains a banner for Code::Blocks with the text "Code::Blocks The open source, cross-platform IDE http://www.codeblocks.org" and a logo of four colored blocks. A red box labeled "The code editor will usually appear here with files in individual tabs." points to this area. Below the banner are several links: "Create a new project", "Open an existing project", "Visit the Code::Blocks forums", "Report a bug", and "Request a new feature". Below these links is a "Recent projects" section with a link to "C:\Users\Rex\Documents\CodeBlocks\projects\Test#Test.cbn". At the bottom, there is a "Logs & others" panel with tabs for "Code::Blocks", "Search results", "Build log", "Build messages", and "Debugger". A red box labeled "Panels displaying search results, the compilation process etc." points to this panel.

Start here - Code::Blocks 10.05

File Edit View Search Project Build Debug wxSmith Tools Plugins Settings Help

Menu/ Toolbar

Build target:

Compilation buttons

Management

Projects Symbols Resol

Workspace

Start here

Shows open projects and a list of used symbols

The code editor will usually appear here with files in individual tabs.

Create a new project Open an existing project

Visit the Code::Blocks forums Report a bug Request a new feature

Recent projects

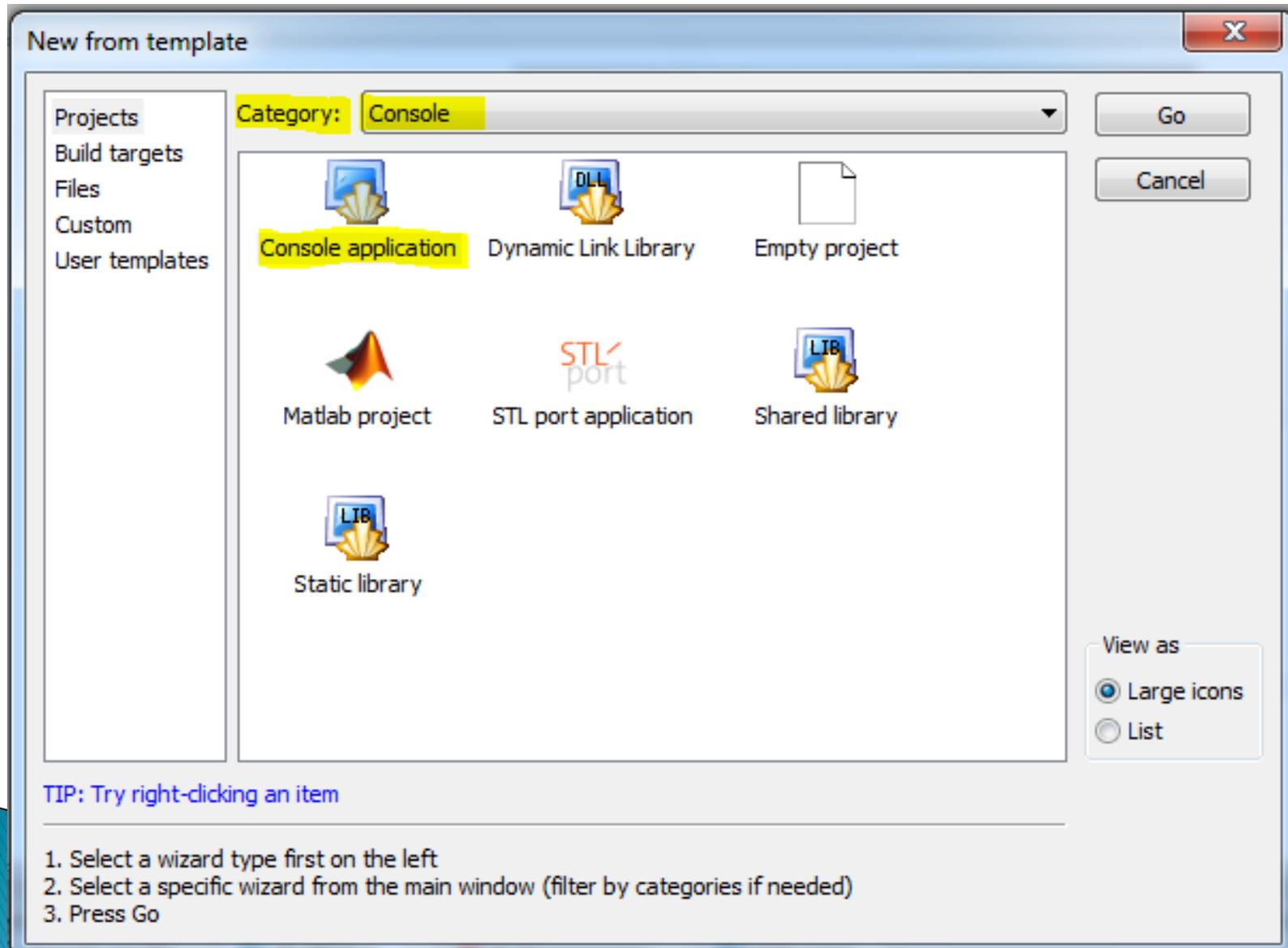
C:\Users\Rex\Documents\CodeBlocks\projects\Test#Test.cbn

Logs & others

Code::Blocks Search results Build log Build messages Debugger

Panels displaying search results, the compilation process etc.

# Creating a new project



# The first application

Build & Run / F9

Header files

Declaration of namespace

The screenshot shows an IDE interface with a project tree on the left and a code editor on the right. The project tree shows a workspace named 'Test' containing a source file 'main.cpp'. The code editor displays the following C++ code:

```
1 #include <iostream>
2
3 using namespace std;
4
5 int main()
6 {
7     cout << "Hello world!" << endl;
8     return 0;
9 }
10
```

Annotations with red arrows point to specific parts of the code:

- An arrow points from the 'Header files' label to the `#include <iostream>` line.
- An arrow points from the 'Declaration of namespace' label to the `using namespace std;` line.
- An arrow points from the 'Output of the output stream content' label to the `cout << "Hello world!" << endl;` line.
- An arrow points from the 'Exit code' label to the `return 0;` line.

Output of the output stream content

Exit code

- Sources – displays files with the extensions \*.c;\*.cpp;.
- ASM Sources – contains \*.s;\*.S;\*.ss;\*.asm files.
- Headers – mainly header files – \*.h;.
- Resources – contains \*.res;\*.xrc files (for wxWidgets)

# Get to know your environment, part 1

## ► Abbreviation mechanism

Keyword/code pairs

Keywords:	Code:
struct	1 struct
whileb	2 {
if	3 →
guard	4 };
nowl	5
while	
nowlu	
ifei	
ifb	
todayu	
ife	
for	
forb	
tday	
wdu	
nowu	
tdayu	
switch	
today	
now	
class	

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

Ctrl + J

```
switch ()
{
    case :
        break;

    default:
        break;
}
```

# Get to know your environment, part 2

- ▶ Search and navigation in the code

**Ctrl + B** – create a bookmark

**Alt + PgUp** \ **Alt + PgDown** – tab navigation

**Ctrl + F** – find in code

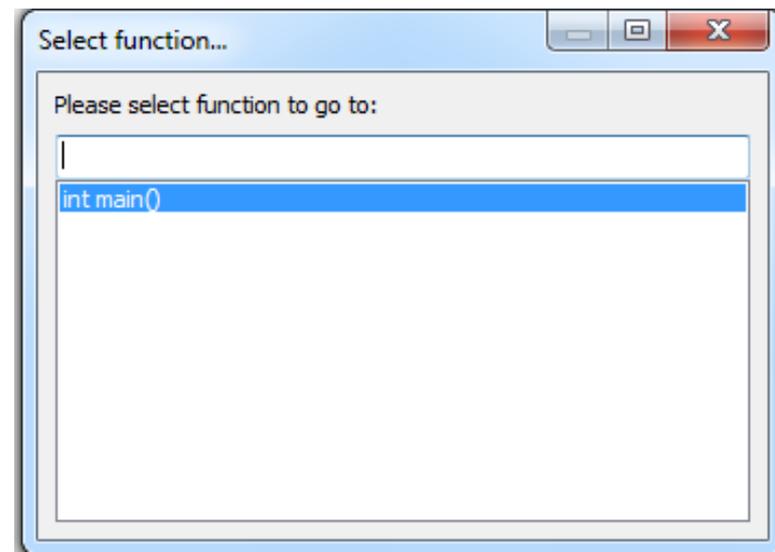
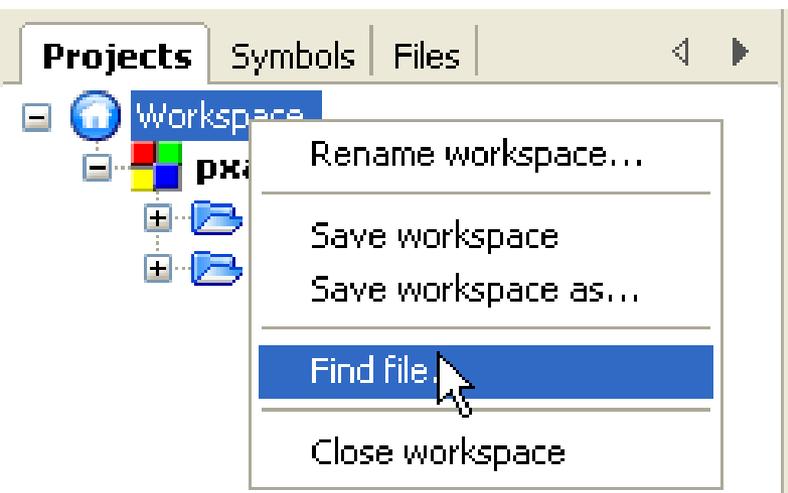
**Ctrl + Shift + F** – find in files

**Ctrl + Alt + G** – find a function

**Ctrl + PgUp** \ **Ctrl + PgDown** – function navigation

**Ctrl + Alt** – find a file to open

**Ctrl + I** – incremental search.



# Get to know your environment, part 3

- ▶ Navigating header files
  - Place the cursor on the `#include` directive and select **open include file** from the pop-up menu.
  - Choose **Swap header/source**.

src\clock.c | src\ledserver.c | **bsp\TriBoard-TC1130\src\rs232.c** X

```
1  /*-----  
2  * Project: Board Support Package (BSP)  
3  * Developed using:  
4  * Function: Transmit and receive characters via TriCore's serial line  
5  *           (interrupt-driven).  
6  *  
7  * Copyright HighTec EDV-Systeme GmbH 1982-2006  
8  *-----*/  
9  
10 #include <machine/cint.h>  
11 #include <machine/wdtcon.h>  
12 #include "rs232.h"  
13  
14 #include <tc1130b/port-struct.h>  
15 #include <tc1130b/asc-struct.h>  
16  
17 #ifndef BAUDRATE  
18 #define BAUDRATE          38400  
19 #endif /* BAUDRATE */
```

Toggle breakpoint

Run to cursor

Open #include file: 'tc1130b/asc-struct.h'

# Get to know your environment, part 4

- ▶ **Code folding**

```
//{
```

Very long code

```
//}
```

- ▶ **Code auto complete**

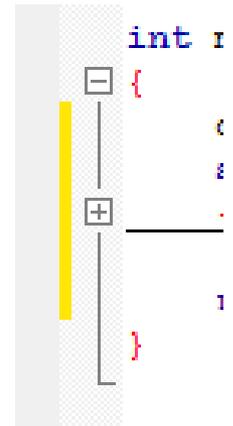
- Ctr + space – forcing auto-complete.

- ▶ **Code export**

- File / Export Menu
  - html
  - rtf
  - odt
  - pdf

- ▶ **Line numbers**

- Settings -> General Settings-> **Show line numbers**



# Keyboard shortcuts

Function	Shortcut Key
Undo last action	Ctrl-Z
Redo last action	Ctrl-Shift-Z
Swap header / source	F11
Comment highlighted code	Ctrl-Shift-C
Uncomment highlighted code	Ctrl-Shift-X
Auto-complete / Abbreviations	Ctrl-Space/Ctrl-J
Toggle bookmark	Ctrl-B
Goto previous bookmark	Alt-PgUp
Goto next bookmark	Alt-PgDown

Function	Shortcut Key
New file or project	Ctrl-N
Open existing file or project	Ctrl-O
Save current file	Ctrl-S
Save all files	Ctrl-Shift-S
Close current file	Ctrl-F4/Ctrl-W
Close all files	Ctrl-Shift-F4/Ctrl-Shift-W

Create or delete a bookmark	Ctrl-F2
Go to next bookmark	F2
Select to next bookmark	Alt-F2
Find selection.	Ctrl-F3
Find selection backwards.	Ctrl-Shift-F3
Find matching preprocessor conditional, skipping nested ones.	Ctrl-K

Function	Shortcut Key
Find	Ctrl-F
Find next	F3
Find previous	Shift-F3
Find in files	Ctrl-Shift-F
Replace	Ctrl-R
Replace in files	Ctrl-Shift-R
Goto line	Ctrl-G
Goto next changed line	Ctrl-F3
Goto previous changed line	Ctrl-Shift-F3
Goto file	Alt-G
Goto function	Ctrl-Alt-G
Goto previous function	Ctrl-PgUp
Goto next function	Ctrl-PgDn
Goto declaration	Ctrl-Shift-.
Goto implementation	Ctrl-.
Open include file	Ctrl-Alt-.

Function	Shortcut Key
Build	Ctrl-F9
Compile current file	Ctrl-Shift-F9
Run	Ctrl-F10
Build and Run	F9
Rebuild	Ctrl-F11

# Streams

```
std::cout << "I know " << 219 << " ways to make an  
omelette." << std::endl;
```

**std::endl** - represents the end of the line  
(which moves to the next one).  
Flushes the output stream buffer.

**'\n'** - special character sequence -  
represents the transition to a new line

**cout** - object representing a program's  
standard output stream

**<<** - inserter. Data is output in accordance  
with their type.



# Types of variables

Type	Description	Usage
<b>int</b>	A signed integer	<code>int i = - 5;</code>
<b>short (int)</b>	Short integer (minimum 2 bytes)	<code>short a = -10;</code>
<b>long (int)</b>	Long integer (minimum 4 bytes)	<code>long z = 20;</code>
<b>long long (int)</b>	Very long integer (minimum 8 bytes)	<code>long long b = 30;</code>
<b>unsigned ... (int)</b>	Numbers $\geq 0$	<code>unsigned int i = 4;</code>
<b>float</b>	Single precision floating point numbers	<code>float f = 7.5f;</code>
<b>double</b>	Double precision floating point numbers	<code>double d = 7.5;</code>
<b>long double</b>	Large floating point numbers.	<code>long double d = 2.5L</code>
<b>char</b>	A single character	<code>char ch = 'a';</code>
<b>char16_t</b>	Single character (16-bit)	<code>char ch16 = u'a';</code>
<b>char32_t</b>	Single character (32-bit)	<code>char ch32 = U'a';</code>
<b>wchar_t</b>	A single character (size depends on the compiler)	<code>wchar_t = L'a'</code>
<b>bool</b>	true / false (zero or no 0)	<code>bool b = false;</code>
<b>auto</b>	The compiler will automatically select the type	<code>auto i = 20;</code>
<b>decltype (wyrażenie)</b>	The type that was given as a parameter	<code>decltype(int) j = 8;</code>

# Special sequences

- ▶ They represent characters generally without graphic equivalents

<code>\a</code>	<code>0x07</code>	<code>BEL</code>	Audible bell
<code>\b</code>	<code>0x08</code>	<code>BS</code>	Backspace
<code>\f</code>	<code>0x0C</code>	<code>FF</code>	Formfeed
<code>\n</code>	<code>0x0A</code>	<code>LF</code>	Newline (linefeed)
<code>\r</code>	<code>0x0D</code>	<code>CR</code>	Carriage return
<code>\t</code>	<code>0x09</code>	<code>HT</code>	Tab (horizontal)
<code>\v</code>	<code>0x0B</code>	<code>VT</code>	Vertical tab
<code>\\</code>	<code>0x5c</code>	<code>\</code>	Backslash
<code>\'</code>	<code>0x27</code>	<code>'</code>	Apostrof
<code>\"</code>	<code>0x22</code>	<code>"</code>	Cudzysłów
<code>\?</code>	<code>0x3F</code>	<code>?</code>	Pytajnik
<code>\O</code>		any	O = łańcuch ósemkowych cyfr
<code>\xH</code>		any	H = łańcuch szesnastkowych cyfr
<code>\XH</code>		any	H = łańcuch szesnastkowych cyfr

# Types of variables in practice

```
(C99) char: -128..127
      short int: -32768..32767
      int: -2147483648..2147483647
      long int: -2147483648..2147483647
      long long int: -9223372036854775808..9223372036854775807
      unsigned char: 0..255
      unsigned short int: 0..65535
      unsigned int: 0..4294967295
      unsigned long int: 0..4294967295
(C99) unsigned long long int: 0..18446744073709551615
```

## ▶ Additionally:

`sizeof( char ) <= sizeof( short int ) <= sizeof( int ) <= sizeof( long int )`

## ▶ And:

`unsigned char` = odpowiednik bajta (byte)

`unsigned short int` = odpowiednik słowa (word)

Most compilers will display a **warning** when using uninitialized variables, and some development environments will display a **run-time error** when we want to retrieve an uninitialized value.

Operator	Description	Usage
=	A binary assignment operator	<code>int i = 1;</code>
!	Unary operator of logical negation	<code>bool b = !true;</code>
+	Binary operators: addition	<code>int i = 3 + 2;</code>
-	subtraction	<code>int i = 3 - 2;</code>
*	multiplication	<code>int i = 3 * 2;</code>
/	division (result dependent on arguments)	<code>int i = 3 / 2;</code>
%	Binary modulo operator	<code>int reszta = 3 % 2;</code>
++	Unary incrementation operator	<code>int i = 1;</code>
--	Unary decrementation operator	<code>i++;</code> or <code>++i;</code>
~	Unary complement (bit inversion)	<code>int z = ~1</code>
&	Bitwise AND	<code>int z = 2 &amp; 3;</code>
	Bitwise inclusive OR	<code>int z = 2   3;</code>
<<	Shift bits left	<code>int z = 4 &lt;&lt; 1;</code>
>>	Shift bits right	
^	Bitwise exclusive OR (XOR)	<code>int z = 2 ^ 3;</code>
&&	Logical AND	<code>if (a &gt; 5 &amp;&amp; a &lt; 10)</code>
	Logical OR	<code>if(a == 2    a == 3)</code>
==	Relational and comparison operators: equal to not equal to	<code>if (a == 3)</code>
!=		<code>if (a != 3)</code>
<, <=	Less then (or equal to)	<code>if (a &lt; 3)</code>
>, >=	Greater then (or equal to)	<code>if (a &gt;= 3)</code>

# Identifiers

- ▶ An identifier is a sequence of letters, numbers and underscore characters beginning with a letter, with the underscore being treated as a letter.
- ▶ Uppercase and lowercase letters are distinguished.
- ▶ Note – Polish characters are not treated as letters! Keywords are also protected.
- ▶ Which are correct?

Control

K21

Pi

PI

\_Control

J-92

\_2Control

Wartość

2controls

This test

# Exercise 1

1. Declare several variables: float, double, int. Assign them a starting value. Print them to the screen (using cout).
2. What is the largest integer that can be entered into a C++ program?
3. Declare a constant and try to change it. What will happen?

We declare constants with the use of const e.g.

```
const int i = 1;
```

# Input and output

- ▶ **cin** — a stream representing the standard program input.
- ▶ **cout** — a stream representing the standard output of the program.
- ▶ **cerr** — **unbuffered** error output stream.
- ▶ **clog** — **buffered** error output stream.

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

using namespace std;

int main()
{
    int liczba;

    cout << "Podaj liczbe: ";
    cin >> liczba;
    cout << '\n' << flush;

    if (cin.good())
        cout << "Dobry strumien" << endl;
    else
        cout << "Bledy w strumieniu" << endl;

    cout << "Podales wartosc: " << liczba << endl;

    cin.ignore();
    cin.get();

    return EXIT_SUCCESS;
}
```

# Exercise 2

1. Declare an int variable representing the user's age. Input your age (using the keyboard) and display it.
2. Try to type something incorrect (like „Alice has a cat”). What's happening?

# Exercise 2

1. Declare an int variable representing the user's age. Input your age (using the keyboard) and display it.
2. Try to type something incorrect (like „Alice has a cat”). What's happening?

**Validate user input!**

Errors?

```
std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
```

**OR**

```
std::cin.sync();
```

# Streams and errors

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

using namespace std;

int main()
{
    int liczba;

    cout << "Podaj liczbe: ";
    cin >> liczba;
    cout << '\n' << flush;
    cout << "Podales wartosc: " << liczba << endl;

    cout << "Jeszcze jedna prosze..." << endl;
    cin.clear();
    cin.ignore( numeric_limits< streamsize >::max( ), '\n' );
    cin >> liczba;
    cout << "Podana druga wartosc to: " << liczba << endl;

    return EXIT_SUCCESS;
}
```

# Exercise 3

1. What will be the result of: *float a = 7 / 2; cout << a;*
2. How can I make the previous expression give the correct result?
3. What will be the result of the following pieces of code?

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

using namespace std;

int main()
{
    int a = 1;
    a = a++;
    a = ++a;
    a = a++ + ++a;
    cout << ++a << " " << ++a << endl;
    cout << a++ << " " << a++ << endl;

    return EXIT_SUCCESS;
}
```

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

using namespace std;

int main()
{
    int a = 2147483647; // MAX_INT = 2147483647
    cout << a++ << endl;
    cout << a << endl;

    return EXIT_SUCCESS;
}
```

<http://c-faq.com/expr/evalorder2.html>

# Exercise 4

- ▶ What will be the result of:

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

using namespace std;

int main()
{
    int a = 1, b = 2;
    if (++a = b)
        cout << "Rowne" << endl;
    else
        cout << "Rozne" << endl;
    cout << a << " " << b << endl;

    return EXIT_SUCCESS;
}
```

Zero means false,  
true otherwise.

# Exercise 5

1. Declare three variables named A, B, C, which will be the coefficients of the quadratic equation written in the form::

$$Ax^2 + Bx + C = 0$$

Write a program that solves this quadratic equation for the coefficients given by the keyboard. Take care of the correctness (validity) of the input data.

Square root: `#include <cmath>`

`sqrt()`

# Language reference: loops

- ▶ While loop:

```
int j = 2;
while (j < 9) {
    cout << j << " ";
    j += 2;
}
```

- ▶ For loop:

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

- ▶ Do.. while loop:

```
int i = 11;
do {
    i = i + 10;
    cout << i << " ";
} while (i < 10);
```

- ▶ Range-based loop:

```
vector<int> v = {0, 1, 2, 3, 4, 5};
for (auto i : v)
    std::cout << i << ' ';
```

**Break and continue works as expected**

# Exercise 6

1. What will be the result of the program:

```
int main()
{
    int i;
    for (i = 1; i <= 5; ++i ) cout << i << endl;
    for(; i >= 1; i--) cout << i << endl;

    return EXIT_SUCCESS;
}
```

2. What will the following loops do:

```
int a = 2;
for (;;) { }
for (;1;) {}
for (a;a;a) {}
while(1) { }
do { } while(1);
```

How to stop them?

# Exercise 7

1. Write a program that reads natural numbers until you specify 0. Then it prints the minimum, maximum of the given numbers and their average.
2. Determine all prime numbers from 0 to 100.
3. Using loops, write a program that displays even integers from the range 31 to 52.
4. Write a program that displays on screen numbers between 1 and 100 divisible by 4, but indivisible by 8 and indivisible by 10. Use the continue statement for this purpose.
5. Write a program, defining how many years you have to save in a bank on a 5% deposit, with earnings of 12,000 \$ per year to have at least 200000 zł. Assume that no tax is charged on interest or income.

# Exercise 8

1. Write a program calculating the largest common divisor of two integers given by the user using the Euclid's algorithm.
2. Write a program which calculates a factorial of a given number.
3. Write a calculator program that will perform the following operations:
  - Addition of two numbers
  - Substraction of two numbers
  - Division of two numbers
  - Multiplication of two numbers
  - Determination of the square root of a number
  - Calculating the percentage from a number.
  - Determining the remainder of the division of two numbers.
  - Determination of any power of a given number.

The calculator should allow selection of operations as long as the user wants.

You should use the instructions `cin.good()` or `cin.fail()` in your programs.