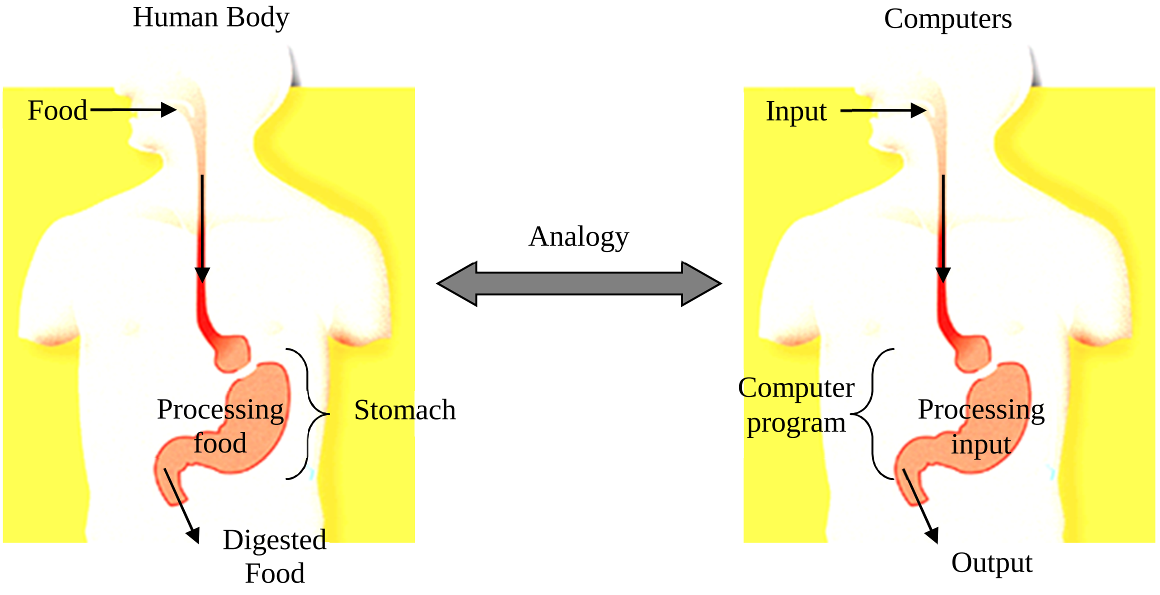
Unit 1: Introduction

1. What is a ‘computer program’?

- *Example*: ‘Windows’ is a computer program.
- Computer programs make your work easier to do.
    - *Example*: Microsoft Word is a program; you type your assignments with Microsoft Word.
- A computer program has purposes, makes a process.
    - *Analogy*: Stomach, eating process – Internet Explorer, shows Web pages.

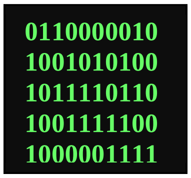
2. What is ‘input’ and ‘output’?



3. What is ‘machine language’ and ‘compiler’?

Question: When you talk to computer in English, does it understand you?

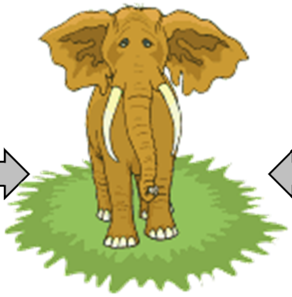
Machine Language:
A lot of ‘0’s and ‘1’s



```
0110000010
1001010100
1011110110
1001111100
1000001111
```

Question: Is it hard to learn and understand?

Tommy knows English

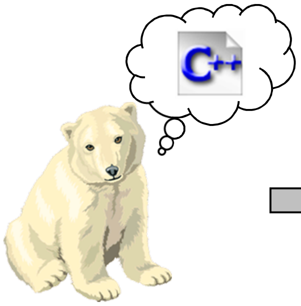


Freddy knows both English and Russian

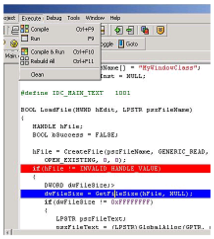
Brady knows Russian

Question: How can Tommy communicate with Brady?

Question: What should we do to communicate with computer?



Learn a *programming language*



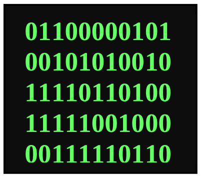
```
                    Name[] = "MyWindowClass";
                    Inst = NULL;

#define IDC_MAIN_TEXT   1001

BOOL LoadFile(HWND hEdit, LPSTR pszFileName)
{
    HANDLE hFile;
    BOOL bSuccess = FALSE;

    hFile = CreateFile(pszFileName, GENERIC_READ,
        OPEN_EXISTING, 0, 0);
    if(hFile != INVALID_HANDLE_VALUE)
    {
        DWORD dwFileSize;>
        dwFileSize = GetFileSize(hFile, NULL);
        if(dwFileSize != 0xFFFFFFFF)
        {
            LPSTR pszFileText;
            pszFileText = (LPSTR)GlobalAlloc(GPTR,
```

Compiler is a computer program that translates programming language to machine language



```
01100000101
00101010010
11110110100
11111001000
00111110110
```

Machine language

4. Code::Blocks

We will learn C++ language to communicate with the computer.

Code::Blocks is a compiler that can translate C++ language into machine language.

- *Editor* is the part you write in C++ language.

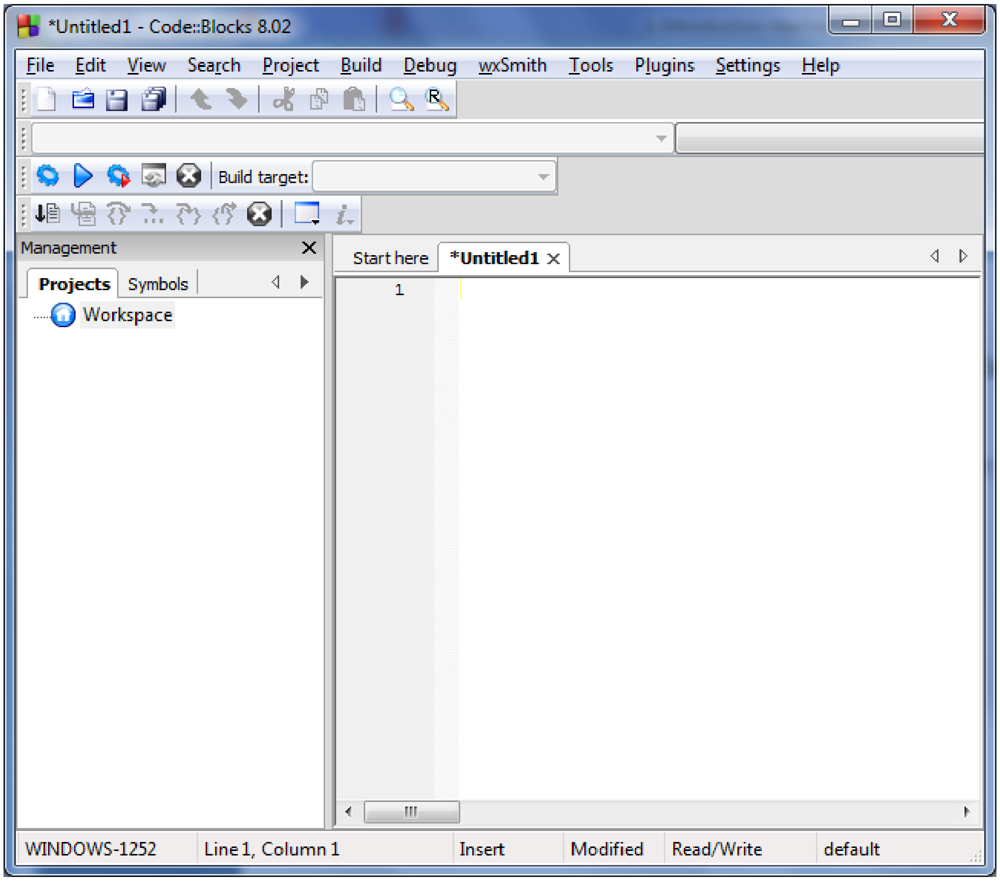


Figure. Code::Blocks editor is the box on the right side.

5. Writing to the Screen

Use ‘`cout`’ command to write to the screen.

Note: Be careful, C++ is case-sensitive.

Example 1: Write ‘`hello world!`’ to the screen.

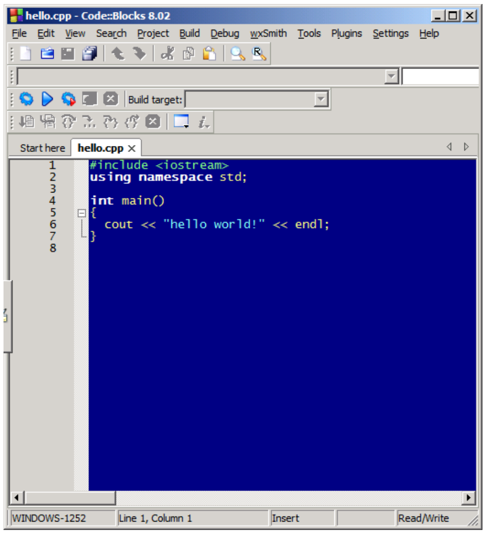
Open a new file, write the code below, name it as ‘`hello.cpp`’, save it on the desktop and run it.

```
#include <iostream>
using namespace std;

int main()
{
  cout << "hello world!" << endl;
}
```

Note 1: ‘Screen’ refers to the black output window.

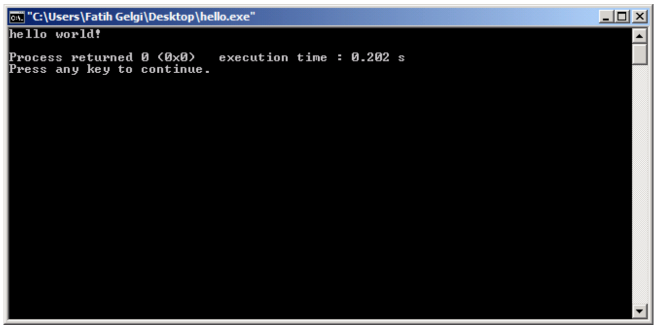
Note 2: Do not forget the semicolons at the end of the lines.



```
#include <iostream>
using namespace std;

int main()
{
   cout << "hello world!" << endl;
}
```

(a)



```
hello world!

Process returned 0 (0x0)   execution time : 0.202 s
Press any key to continue.
```

(b)

As shown in the figure on the left, after we write ‘hello world!’ program in the editor and run it, the program will write ‘hello world!’ to the screen as shown in the figure on the right. We write our program in the editor and see the output of our program in a black window as shown in the figures on the left and right respectively.

Exercise 1: Write your name to the screen.

Open a new file, name it as ‘`myname.cpp`’, save it on the desktop and run it.

Note: Do not forget to put double quotes before and after your name.

Exercise 2: Write your name twice to the screen.

Open a new file, name it as ‘`myname2.cpp`’, save it on the desktop and run it.

Example 2: Write ‘`1232`’ on the screen.

Open a new file, write the code below, name it as ‘`number.cpp`’, save it on the desktop and run it.

```
#include <iostream>
using namespace std;

int main()
{
  cout << 1232 << endl;
}
```

Exercise 3: Write your lucky number on the screen.

Open a new file, name it as ‘`lucky.cpp`’, save it on the desktop and run it.

`endl` is to used to pass to next line. The next time we use `cout`, it starts from the beginning of the next line. If we don’t use it, the output immediately starts from the location stopped last time. For instance,

```
cout << "quick brown fox" << endl;
cout << "jumped over the lazy dog" << endl;
```

will write;

```
quick brown fox
jumped over the lazy dog
```

on the screen. On the other hand,

```
cout << "quick brown fox";
cout << "jumped over the lazy dog" << endl;
```

will output;

```
quick brown foxjumped over the lazy dog
```

Note that there is no space between ‘`fox`’ and ‘`jumped`’.