

# Introduction to Programming I & Programming Software Systems I

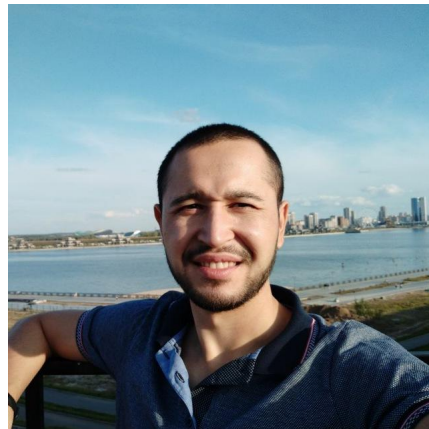
Sirojiddin Komolov, Mansur Khazeev and Munir Makhmutov

# Your TA

Sirojiddin Komolov.

You can contact me at Telegram: **@siroj09**

Email: [s.komolov@innopolis.ru](mailto:s.komolov@innopolis.ru)



- Have 7 years of experience as a Programmer in industry
- Developed many web/mobile applications (mostly Android apps) - (the most successful - android app of kwork.ru 100K+ downloads)
- Participated in many startups, btw most of them failed :)
- Have experience in Project management, Requirement engineering, Designing web/mobile applications
- Top rated freelancer in upwork

# Tools

What do you need to program in C?

# Tools

You only need:

(a) Text Editor

(b) The C Compiler.



# Compilers

The source code written in source file is the human readable source for your program. It needs to be "compiled", into machine language so that your CPU can actually execute the program as per the instructions given.

We will use GNU C/C++ which is mostly known as **gcc** compiler.

**Me:** Misses semicolon

**My compiler:**



4276 errors, 1872 warnings.

<https://me.me/i/me-misses-semicolon-my-compiler-ladies-and-gentlemen-fasten-your-52020e7e45e24e3e88c256798571a1bb>

# Installation of gcc - on UNIX/Linux.

```
$ gcc -v
```

If you have GNU compiler installed on your machine, then it should print a message as follows –

```
Using built-in specs.  
Target: i386-redhat-linux  
Configured with: ../configure --prefix=/usr .....  
Thread model: posix  
gcc version 4.1.2 20080704 (Red Hat 4.1.2-46)
```

If GCC is not installed, then you will have to install it yourself using the detailed instructions available at <https://gcc.gnu.org/install/>

# Installation of gcc - on Mac OS.

If you use Mac OS X, the easiest way to obtain GCC is to download the Xcode development environment from Apple's web site and follow the simple installation instructions. Once you have Xcode setup, you will be able to use GNU compiler for C/C++.

Xcode is currently available at [developer.apple.com/technologies/tools/](https://developer.apple.com/technologies/tools/).

# Installation of gcc - on Windows.

To install GCC on Windows, you need to install MinGW. To install MinGW, go to the MinGW homepage, [www.mingw.org](http://www.mingw.org), and follow the link to the MinGW download page. Download the latest version of the MinGW installation program, which should be named MinGW-<version>.exe.

[Linux subsystem for Windows](#)



# More about gcc

- Run **gcc**:

***gcc -Wall infilename.c -o outfilename.o***

- **-Wall** enables most compiler warnings
- More complicated forms exist
  - multiple source files
  - auxiliary directories
  - optimization, linking
- Embed debugging info and disable optimization:

***gcc -g -O0 -Wall infilename.c -o outfilename.o***

# More about gcc. Compiling simple programs

Say you have a file **hello.c** as follows :

```
#include  
void main ()  
{  
    printf("hello");  
}
```

# More about gcc. Compiling simple programs

You can compile and run it from the unix prompt as follows :

```
% gcc hello.c
```

This creates an **executable** called "a.out". You can run it by typing

```
% ./a.out
```

at the prompt. a.out is the default name for an executable. Use the "-o" option to change the name :

```
% gcc -o hello hello.c
```

creates an executable called "hello".

# More about gcc. Include Directories

Sometimes the header files that you write are not in the same directory as the .c file that `#include`'s it. E.g. you have a structure in a file "foo.h" that resides in /homes/me/include. If I want to include that file in hello.c I can do the following :

- add  
`#include <foo.h >`
- to hello.c
- compile with the -I option :  
***gcc -o hello hello.c -I /homes/me/include***

This basically tells gcc to look for `#include`'s in /homes/me/include **in addition** to other directories you specify with -I

# More about gcc. Compiling multiple files

Basic idea : compile each .c file into a .o file, then compile the .o files (along with any libraries) into an executable. One of these .c files obviously has to define the main function, or else the linker will gag.

E.g. Suppose we have main.c, foo.c and bar.c and want to create an executable fubar, and suppose further that we need the math library:

```
gcc -c foo.c
```

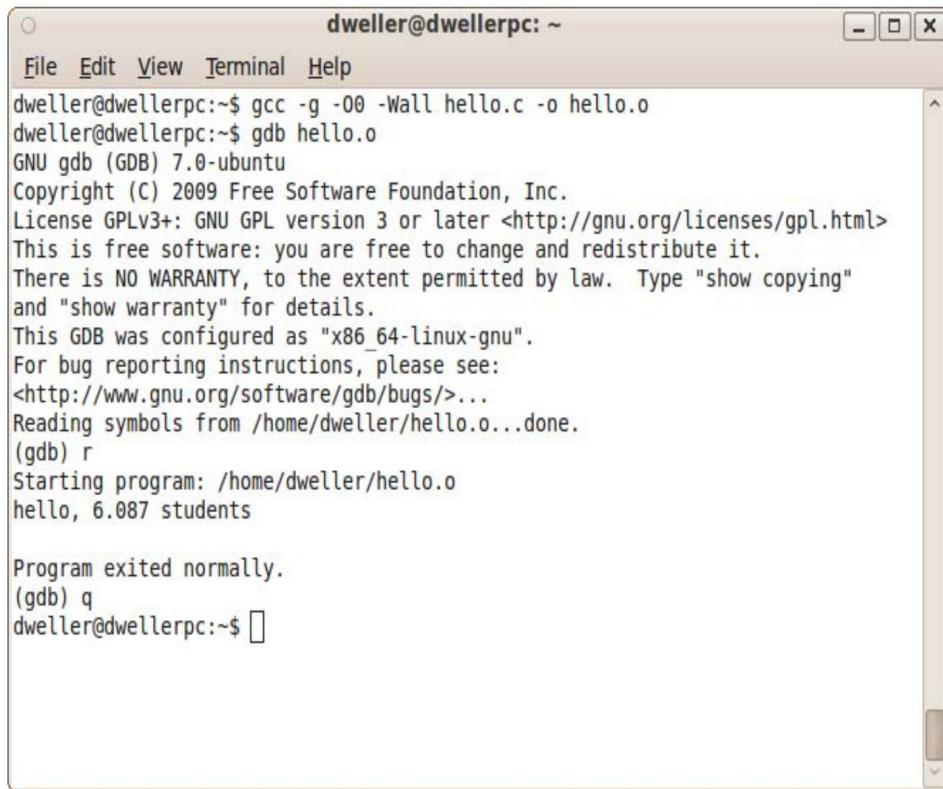
```
gcc -c main.c
```

```
gcc -c bar.c
```

```
gcc -o fubar foo.o main.o bar.o -lm
```

The first three commands generate foo.o, main.o and bar.o respectively. The last line links them together along with the math library.

# Debugging



```
dweller@dwellerpc: ~  
File Edit View Terminal Help  
dweller@dwellerpc:~$ gcc -g -O0 -Wall hello.c -o hello.o  
dweller@dwellerpc:~$ gdb hello.o  
GNU gdb (GDB) 7.0-ubuntu  
Copyright (C) 2009 Free Software Foundation, Inc.  
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>  
This is free software: you are free to change and redistribute it.  
There is NO WARRANTY, to the extent permitted by law. Type "show copying"  
and "show warranty" for details.  
This GDB was configured as "x86_64-linux-gnu".  
For bug reporting instructions, please see:  
<http://www.gnu.org/software/gdb/bugs/>...  
Reading symbols from /home/dweller/hello.o...done.  
(gdb) r  
Starting program: /home/dweller/hello.o  
hello, 6.087 students  
  
Program exited normally.  
(gdb) q  
dweller@dwellerpc:~$
```

Figure: gdb: command-line debugger

# Debugging

Some useful commands:

- ***break linenumber*** – create breakpoint at specified line
- ***break file:linenumber*** – create breakpoint at line in file
- **run** – run program
- **c** – continue execution
- **next** – execute next line
- **step** – execute next line or step into function
- **quit** – quit gdb
- **print expression** – print current value of the specified expression
- **help command** – in-program help

# Structure of a .c file

*/\* Begin with comments about file contents \*/*

Insert ***#include*** statements and preprocessor definitions

Function prototypes and variable declarations

***Define main() function {***

***Function body***

***}***

***Define other function***

***{***

***Function body***

***}***



# Comments

- Comments: `/* this is a simple comment */`
- Can span multiple lines

`/* This comment spans  
multiple lines */`

- Completely ignored by compiler
- Can appear almost anywhere

# The #include macro

- Header files: constants, functions, other declarations
- #include <stdio.h> – read the contents of the header file stdio.h
- stdio.h: standard I/O functions for console, files

```
#include <stdio .h> /* basic I /O facilities */
```

# More about #include

- **stdio.h** – part of the C Standard Library
- other important header files: **ctype.h**, **math.h**, **stdlib.h**, **string.h**, **time.h**, For the ugly details: visit [www.unix.org/single\\_unix\\_specification/](http://www.unix.org/single_unix_specification/) (registration required)
- Included files must be on include path
- **-Idirectory** with **gcc**: specify additional include directories
- standard include directories assumed by default
- **#include "stdio.h"** – searches ./ for stdio.h first

# Task 1: write Hello world example

```
#include <stdio.h>

int main() {
    /* my first program in C */
    printf("Hello, World! \n");

    return 0;
}
```

# Executing Hello world example

- Open a text editor and add the above-mentioned code.
- Save the file as *hello.c*
- Open a command prompt and go to the directory where you have saved the file.
- Type *gcc hello.c* and press enter to compile your code.
- If there are no errors in your code, the command prompt will take you to the next line and would generate *a.out* executable file.
- Now, type *a.out* to execute your program.
- You will see the output "*Hello World*" printed on the screen.

# Declaring variables

- Must declare variables before use
- Variable declaration:
  - *int n;*
  - *float phi;*
- **int** - integer data type
- **float** - floating-point data type
- Many other types (more next lecture. . . )

# Initializing variables

- Uninitialized, variable assumes a default value
- Variables initialized via assignment operator:

```
n = 3;
```

- Can also initialize at declaration:

```
float phi = 1.6180339887;
```

- Can declare/initialize multiple variables at once: `int a, b, c = 0, d = 4;`

# Arithmetic expressions

- Suppose  $x$  and  $y$  are variables

$x+y$ ,  $x-y$ ,  $x*y$ ,  $x/y$ ,  $x\%y$ : binary arithmetic

- A simple statement:  $y = x+3*x/(y-4);$
- Numeric literals like **3** or **4** valid in expressions
- Semicolon ends statement (not newline)
- $x += y, x -= y, x *= y, x /= y, x \%= y$ : arithmetic and assignment



## Task 2.

Write a program which declares/initializes 2 integer variables and prints their sum, subtraction, multiplication and division.

$X, Y$

$X + Y$

$X - Y$

$X * Y$

$X / Y$

# What Is Software Engineering?

The job of a software engineer:

- high-quality software
- agreed cost
- in timely manner

You need to know:

- How to plan your work
- How to work according to this plan
- How to produce the high quality products



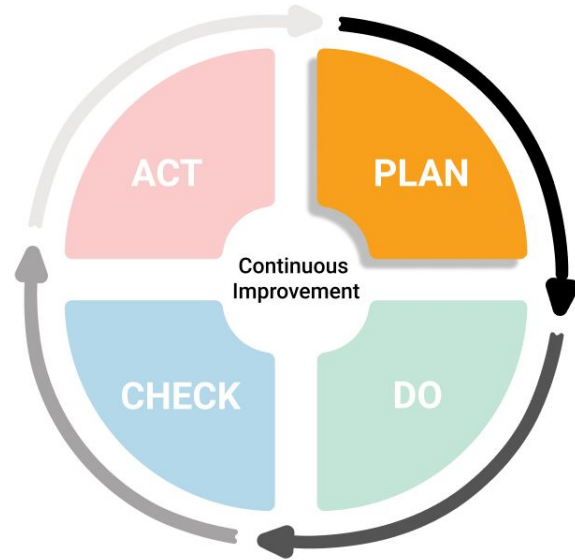
# Why Is Good Engineering Important?

- Business-critical: factories, banking, etc
- Safety and life-critical software
- ...
- Almost every product people use



# The Personal Software Process

- provides detailed estimating and planning methods
- shows how to track performance against these plans
- explains how defined processes can guide your work



# Survey

Please fill in the following survey:

[https://docs.google.com/forms/d/e/1FAIpQLSeiJW0VNLLVP1mXqBvHrMa4VwVs5K17\\_0M7rIJZN9HdtlhYFw/viewform?usp=sf\\_link](https://docs.google.com/forms/d/e/1FAIpQLSeiJW0VNLLVP1mXqBvHrMa4VwVs5K17_0M7rIJZN9HdtlhYFw/viewform?usp=sf_link)

# References

1. <https://www.geeksforgeeks.org/interesting-facts-about-c-language/>
2. <https://www.improgrammer.net/interesting-facts-c-language/>
3. <https://www.tutorialspoint.com/cprogramming/index.htm>
4. <https://courses.cs.washington.edu/courses/cse451/98sp/Section/gccintro.html>
5. <http://hilite.me/>
- 6.