INDIAN INSTITUTE OF TECHNOLOGY ROORKEE

EEC-101
Programming with C++

Ramanuja Panigrahi

About Subject

- Subject code: EEC-101
- Course Title: Programming with C++
- Subject area: PCC
- Objective: To familiarize the students with the fundamentals of programming in C++ and the concepts of object-oriented programming (OOPS)
- Contact hours
  - 3 lectures/week
    - Tuesday: 4.05-5.00 pm
    - Thursday: 3.00-3.55 pm
    - Friday: 4.05-5.00 pm
  - 1 Practical/week
    - Practical will start after one week. Dates will be announced on MS Teams

Evaluation

- Relative weightage
  - Class Work Sessional (CWS): 15
  - Practical Sessional (PRS): 25
  - Mid-term Examination (MTE): 20
  - End-term Examination (ETE): 40

- Distribution of CWS
  - Two Quizzes: 7.5 marks each
    - Date will be announced a week before

You are expected to attend all Lectures and Lab sessions

Join Teams

- All announcements for the course will be made through MS TEAMs.
- Join with the code below.

Content

S. No.	Course Contents	Contact Hours
1.	**Basic Programming in C++:** Concepts of algorithm & flow charts; Input/output, constants, variables, expressions and operators; Naming conventions and styles; Conditions and selection statements; Looping and control structures (while, for, do-while, break and continue); File I/O, header files, string processing; Pre-processor directives such as #include, #define, #ifdef, #ifndef; Compiling and linking.	**9**
2.	**Programming through Functional Decomposition:** Functions (void and value returning), parameters, scope and lifetime of variables, passing by value, passing by reference, passing arguments by constant reference; Design of functions and their interfaces (concept of functional decomposition), recursive functions; Function overloading and default arguments; Library functions; Matters of style, naming conventions, comments.	**10**
3.	**Aggregate Data-types:** Arrays and pointers; Structures; Dynamic data and pointers, dynamic arrays.	**4**

Course Content (cont.)

S. No.	Contents	Contact Hours
4.	**Object Oriented Programming Concepts:** Data hiding, abstract data types, classes and access control; Class implementation-default constructor, constructors, copy constructor, destructor, operator overloading, friend functions; Introduction to Templates	**12**
5.	**Object Oriented Design:** Inheritance and composition; Dynamic binding and virtual functions; Polymorphism; Dynamic data in classes.	**7**

1. Dietel H.M. & Dietel P.J., "C ++ How to Program", Prentice Hall Publications, 10th Edition.
2. Bjarne Stroustrup – Programming: Principles and Practice Using C++ -Addison-Wesley Professional (2024)

Computers, People, and Programming

- Computers are built by people for the use of people.
- A computer is a very generic tool; it can be used for an unimaginable range of tasks.
- It takes a program to make it useful to someone. In other words, a computer is just a piece of hardware until someone — some programmer — writes code for it to do something useful.
- We often forget about the software. Even more often, we forget about the programmer.

Why programming?

- Our civilization runs on software
  - Most engineering activities involve software
- Note: most programs do not run on things that look like a PC
  - a screen, a keyboard, a box under the table

- Design
- Communication
- Control
- Display
- Routing

- Signal processing
- “Gadget” control
- telemetry

Healthcare

Finance

Entertainment

Energy

- Control
- Monitoring
- Analysis
- Design

- Communications
- Visualization
- Manufacturing
- Transport

Program

- To get a computer to do something, you (or someone else) have to tell it exactly – in excruciating detail – what to do. Such a description of ‘‘what to do’’ is called a program.

- And programming is the activity of writing and testing such programs.

- Humans tend to compensate for poor instructions by using common sense, but computers don’t.
- In contrast, computers are really dumb.
- They have to have everything described precisely and in detail.

- To be able to describe ‘‘things’’ precisely for a computer, we need a precisely defined language with a specific grammar (English is far too loosely structured for that) and a well-defined vocabulary for the kinds of actions we want to perform.

- Such a language is called a programming language, and C++ is a programming language designed for a wide selection of programming tasks.

- When a computer can perform a complex task given simple instructions, it is because someone has taught it to do so by providing a program.

Why C++

- You can't learn to program without a programming language
- The purpose of a programming language is to allow you to express your ideas in code
- C++ is the most widely used language in engineering areas

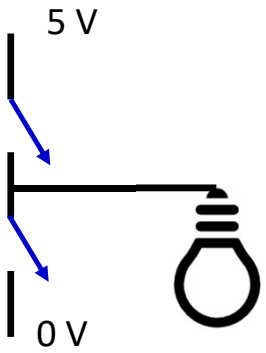
- Other languages often use C++ for computation intensive tasks
  - E.g., Python doing AI
- The implementation of many languages are C++ programs
  - E.g., Java and Javascript

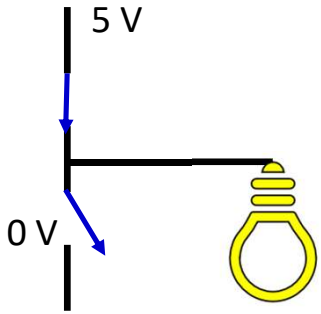
Aug 2024	Aug 2023	Change	Programming Language	Ratings	Change
1	1		Python	18.04%	+4.71%
2	3	↑	C++	10.04%	-0.59%
3	2	↓	C	9.17%	-2.24%
4	4		Java	9.16%	-1.16%
5	5		C#	6.39%	-0.65%
6	6		JavaScript	3.91%	+0.62%
7	8	↑	SQL	2.21%	+0.68%
8	7	↓	Visual Basic	2.18%	-0.45%
9	12	↑	Go	2.03%	+0.87%
10	14	⇈	Fortran	1.79%	+0.75%
11	13	↑	MATLAB	1.72%	+0.67%

- Early 1970s
  - C Programming Language
  - Dennis Ritchie
- 1979
  - Bjarne Stroustrup
  - C with Classes
- 1983
  - Name changed to C++
- 1989
  - First commercial release
- 1998
  - C++98 Standard
- 2003
  - C++03 Standard
- **2011**
  - **C++11 Standard**
- **2014**
  - **C++14 Standard**
- **2017**
  - **C++17 Standard**

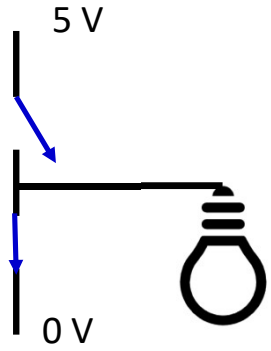
The Goal

BASE-2 or Binary





Represent 1



Represent 0

Binary Digit = Bit

- Switches are called Transistors. Computers have Billions of Transistors.
- Intel i7-940 had 731 million transistors.

Dec	8	4	2	1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

To evaluate the decimal number 593.68, the digit in each position is multiplied by the corresponding weight:

$(593.68)_{10} = 5\times10^{2} + 9\times10^{1} + 3\times10^{0} + 6\times10^{-1} + 8\times10^{-2}$

In Binary,

$(1101.101)_2 = 1\times2^{3} + 1\times2^{2} + 1\times2^{0} + 1\times2^{-1} + 1\times2^{-3}$

$= 8 + 4 + 1 + 0.5 + 0.125 = (13.625)_{10}$

$(11111111)_2 = ?$

Bit to Represent Information

How to Represent 'A' ?

'A' 65 01000001

(01001000 01101001) means?

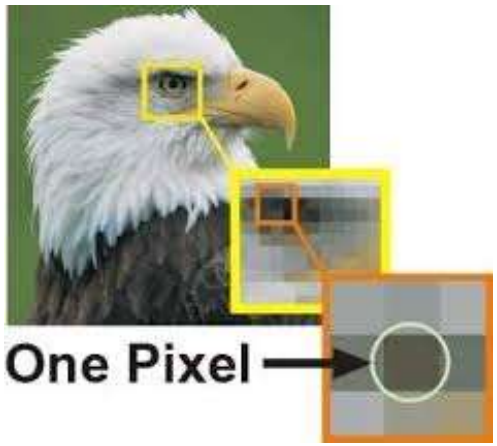
ASCII (American Standard Code for Information Interchange) **a standard data-encoding format for electronic communication between computers**.

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[END OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

01010111 01100101 01101100 01100011 01101111
01101101 01100101 00100000 01110100 01101111
00100000 01000101 01000101 01000011 00101101
00110001 00110000 00110001

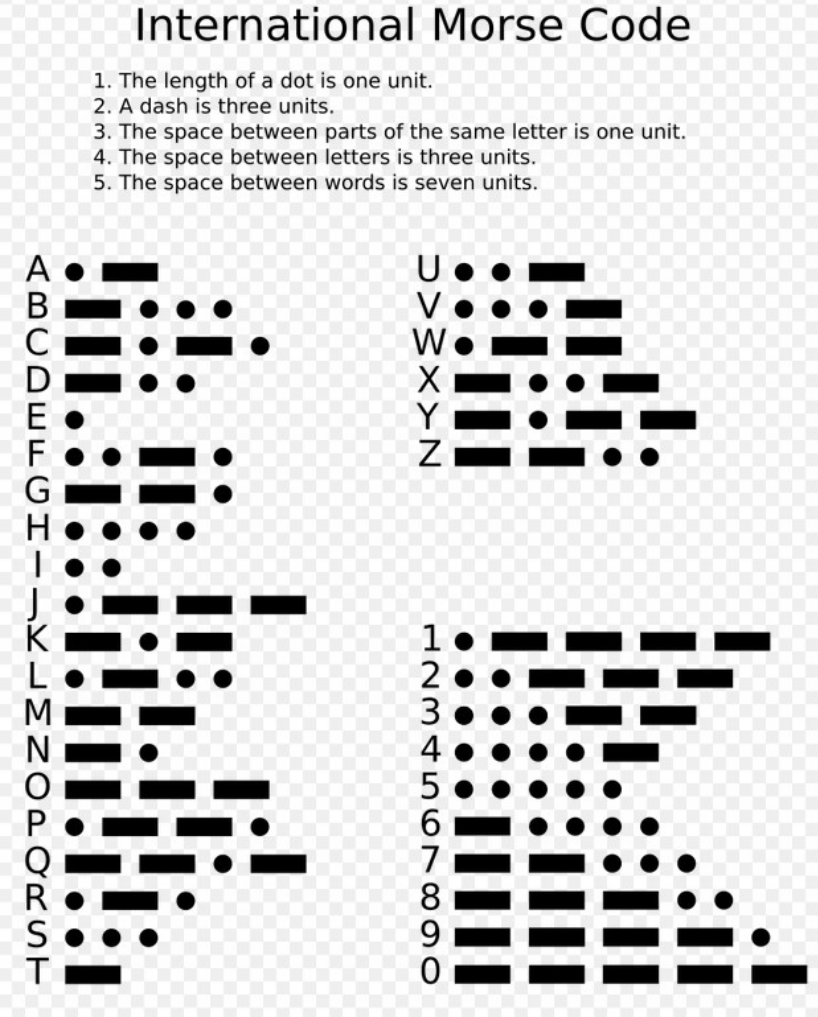
Welcome to EEC-101

Unicode and ASCII are the most popular character encoding standards that are currently being used all over the world. Unicode is the universal character encoding used to process, store and facilitate the interchange of text data in any language while ASCII is used for the representation of text such as symbols, letters, digits, etc. in computers.



Each Pixel is a combination of RED+ BLUE+ GREEN

CODE



Each Morse code symbol is formed by a sequence of *dits* and *dahs*. The *dit* duration can vary for signal clarity and operator skill, but for any one message, once established it is the basic unit of time measurement in Morse code. The duration of a *dah* is three times the duration of a *dit* (although some telegraphers deliberately exaggerate the length of a *dah* for clearer signaling). Each *dit* or *dah* within an encoded character is followed by a period of signal absence, called a *space*, equal to the *dit* duration. The letters of a word are separated by a space of duration equal to three *dits*, and words are separated by a space equal to seven *dits*.

https://en.wikipedia.org/wiki/Morse_code

- Although Morse code has absolutely nothing to do with computers, becoming familiar with the nature of codes is an essential preliminary to achieving a deep understanding of the hidden languages and inner structures of computer hardware and software.

- The word code usually means a system for transferring information among people, between people and computers, or within computers themselves.

- A code lets you communicate. Sometimes codes are secret, but most codes are not. Indeed, most codes must be well understood because they are the basis of human communication.

Computer Programs

- Many of today's personal computers can perform billions of calculations in one second—more than a human can perform in a lifetime.

- China's National University of Defense Technology's Tianhe-2 supercomputer can perform over 33 quadrillion calculations per second (33.86 petaflops)!
  - To put that in perspective, the Tianhe-2 supercomputer can perform in one second about 3 million calculations for every person on the planet!
- Computers process data under the control of sequences of instructions called computer programs.
- The set of programs that run on a computer are referred to as software.

Machine Languages, Assembly Languages, and High-Level Languages

Programmers write instructions in various programming languages, some directly understandable by computers and others requiring intermediate translation steps.

Machine Languages

- Any computer can directly understand only its own machine language (also called machine code), defined by its hardware architecture.
- Machine languages generally consist of numbers (ultimately reduced to 1s and 0s). Such languages are cumbersome for humans.
- An example of a machine language instruction is a simple addition operation: 01100110 00001010. This binary sequence represents an instruction that tells the computer to add two numbers together.

Assembly Languages

- Programming in machine language was simply too slow and tedious for most programmers. Instead, they began using English-like abbreviations to represent elementary operations. These abbreviations formed the basis of assembly languages.
- Translator programs called assemblers were developed to convert assembly-language programs to machine language.
- Although assembly-language code is clearer to humans, it's incomprehensible to computers until translated to machine language.

Example of code Written in Assembly Languages

```
section    .text
  global _start     ;must be declared for linker (ld)

_start:             ;tells linker entry point
  mov      edx,len     ;message length
  mov      ecx,msg     ;message to write
  mov      ebx,1       ;file descriptor (stdout)
  mov      eax,4       ;system call number (sys_write)
  int      0x80        ;call kernel

  mov      eax,1       ;system call number (sys_exit)
  int      0x80        ;call kernel

section    .data
msg db 'Hello, world!', 0xa  ;string to be printed
len equ $ - msg     ;length of the string
```

The Hello World Program in Assembly

The following assembly language code displays the string 'Hello World' on the screen

High-Level Languages

- To speed up the programming process further, high-level languages were developed in which single statements could be written to accomplish substantial tasks.

- High-level languages, such as C, C++, Java, C#, Swift, and Visual Basic, allow you to write instructions that look more like everyday English and contain commonly used mathematical notations.

- Translator programs called **compilers** convert high-level language programs into machine language

```
#include <iostream>
#include <string>
using namespace std;
int main ()
{
    cout << "Hello world!"<<endl;
    return (0);

}
```

The Hello World Program in C++

The following C++ code displays the string 'Hello World' on the screen

- The process of compiling a large high-level language program into machine language can take a considerable amount of computer time.
- Interpreter programs were developed to execute high-level language programs directly (without the need for compilation), although more slowly than compiled programs. Scripting languages such as the popular web languages JavaScript and PHP are processed by interpreters.

- Note:

Interpreters have an advantage over compilers in Internet scripting. An interpreted program can begin executing as soon as it's downloaded to the client's machine without needing to be compiled before it can execute. On the downside, interpreted scripts generally run slower and consume more memory than compiled code

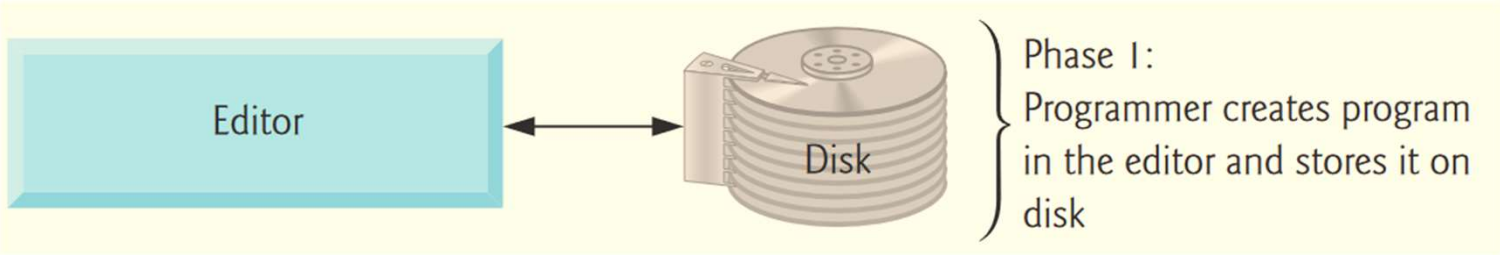
Typical C++ Development Environment

- C++ systems generally consist of three parts:
  - a program development environment,
  - The language
  - the C++ Standard Library.
- C++ programs typically go through six phases:
  - Edit
  - preprocess,
  - compile,
  - link,
  - load
  - execute.

Step-1:Editing a Program

- Phase 1 consists of editing a file with an editor program, normally known simply as an editor.
- You type a C++ program (typically referred to as source code) using the editor, make any necessary corrections, and save the program on your computer's disk.

- C++ source code filenames often end with the .cpp, .cxx, .cc or .C (uppercase) extensions which indicate that a file contains C++ source code.



Phase 2: Compiling a C++ Program

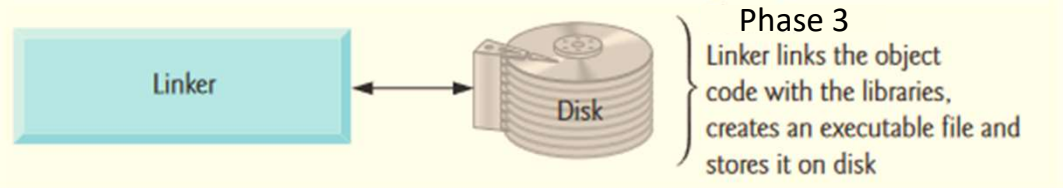
- In Phase 2, you give the command to compile the program. In a C++ system, a preprocessor program executes automatically before the compiler's translation phase begins (so we call preprocessing Phase 2a and compiling Phase 2b).
- The C++ preprocessor obeys commands called preprocessing directives, which indicate that certain manipulations are to be performed on the program before compilation.
- These manipulations usually include (i.e., copy into the program file) other text files to be compiled, and perform various text replacements.

The most common preprocessing directives are discussed in the early chapters

In Phase 2b, the compiler translates the C++ program into machine-language code—also referred to as object code .

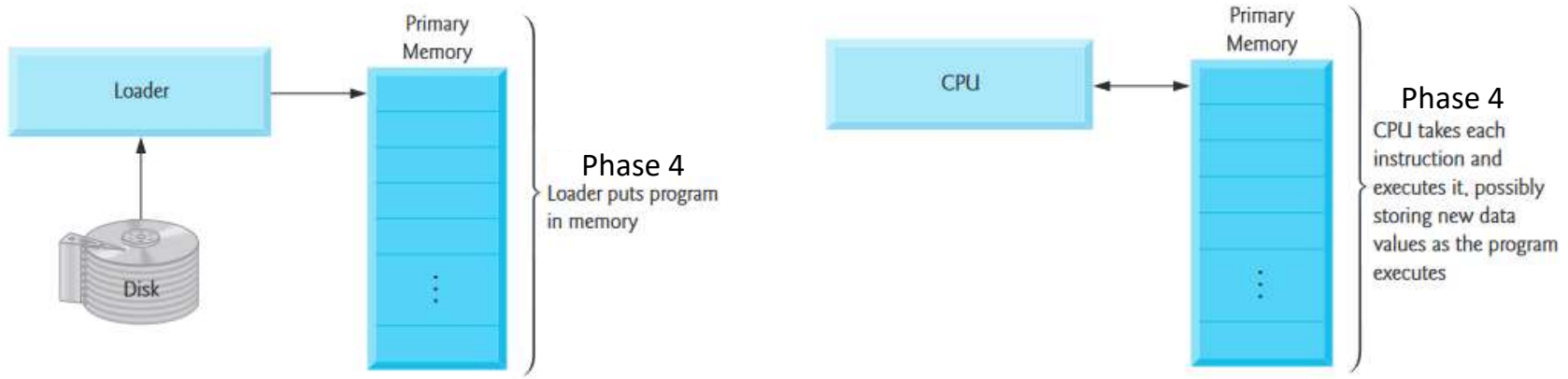
Phase 3: Linking

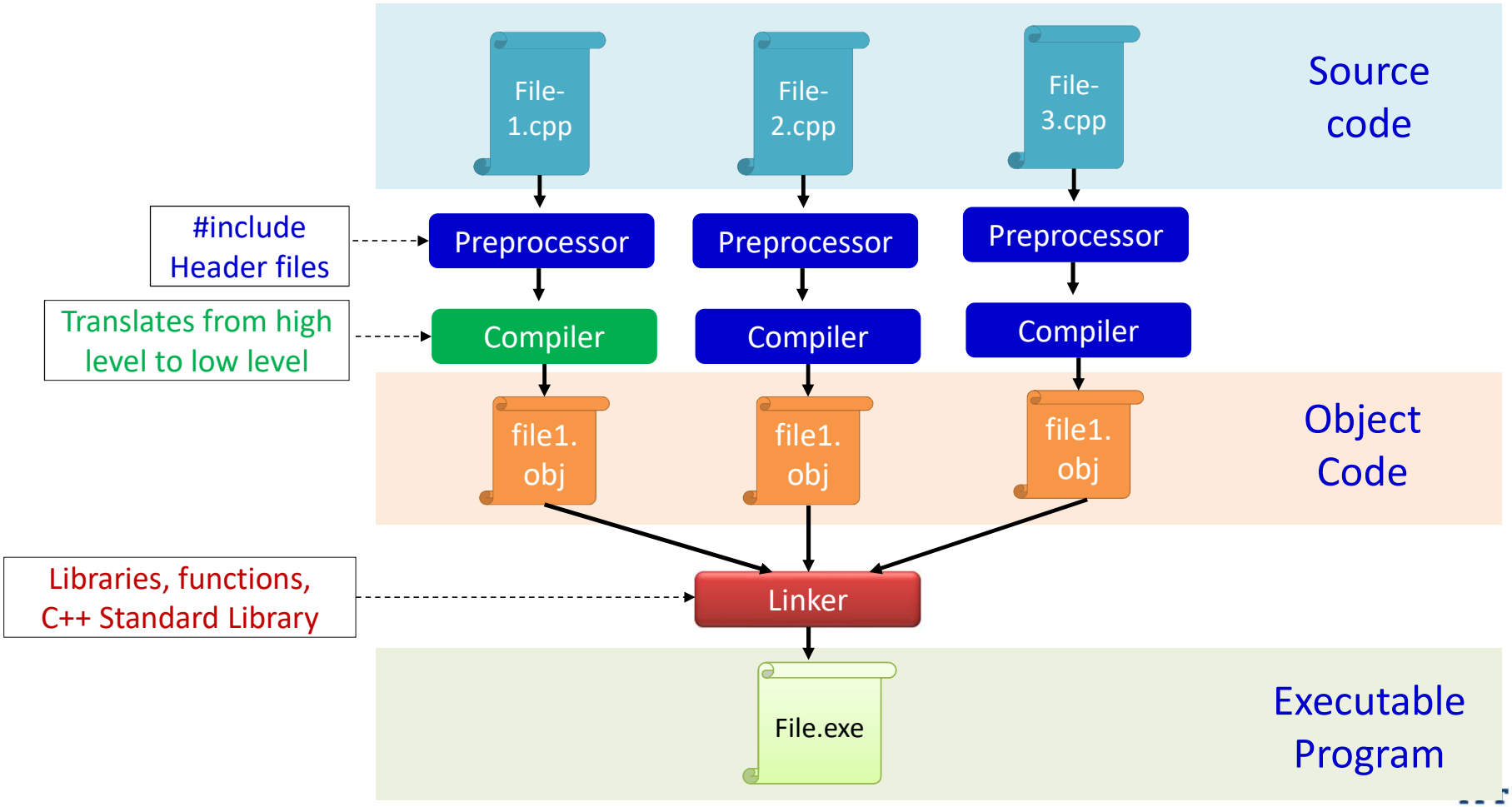
- C++ programs typically contain references to functions and data defined elsewhere, such as in the standard libraries or in the private libraries of groups of programmers working on a particular project.
- The object code produced by the C++ compiler typically contains “holes” due to these missing parts.
- A linker links the object code with the code for the missing functions to produce an executable program (with no missing pieces). If the program compiles and links correctly, an executable image is produced.



Loading and Execution

- Phase 4 is called loading. Before a program can be executed, it must first be placed in
- Memory. This is done by the loader, which takes the executable image from disk and transfers it to memory. Additional components from shared libraries that support the program are also loaded.
- Finally, the computer, under the control of its CPU, executes the program one instruction at a time.



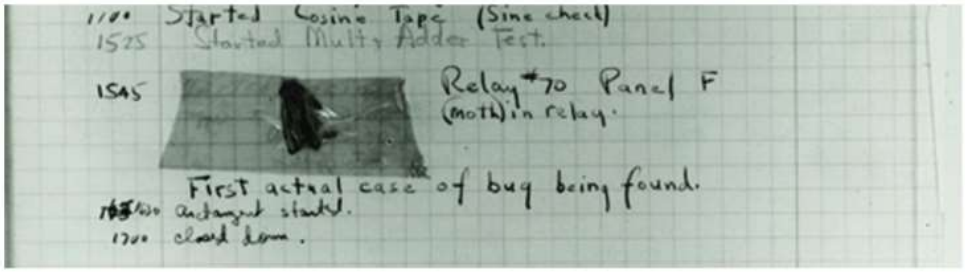

File-1.cpp
File-2.cpp
File-3.cpp
Source code
#include Header files
Preprocessor
Preprocessor
Preprocessor
Translates from high level to low level
Compiler
Compiler
Compiler
file1.obj
file1.obj
file1.obj
Object Code
Libraries, functions, C++ Standard Library
Linker
File.exe
Executable Program

Integrated development environments (IDEs)

- To program, we use a programming language. We also use a compiler to translate our source code into object code and a linker to link our object code into an executable program.
- In addition, we use some program to enter our source code text into the computer and to edit it. These are just the first and most crucial tools that constitute our programmer's tool chest or ''program development environment.''
- IDEs usually include an editor with helpful features like color coding to help distinguish between comments, keywords, and other parts of your program source code, plus other facilities to help you debug your code, compile it, and run it. Debugging is the activity of finding errors in a program and removing them; you'll hear a lot about that along the way.
- An error in a program is often called a bug, hence the term ''debugging.''
- It is also believed that **Microsoft's Visual Basic was the first IDE launched in 1991.**

Bug

- The reason for calling an error "a bug" is that in a very early system a program failed because an insect had found its way into the computer
- Grace Murray Hopper is often credited with being the first person to call an error in a computer a "bug." She certainly was among the early users of the term and documented a use:



As can be seen, that bug was real (a moth), and it affected the hardware directly. Most modern bugs appear to be in the software and have less graphical appeal

Popular IDEs

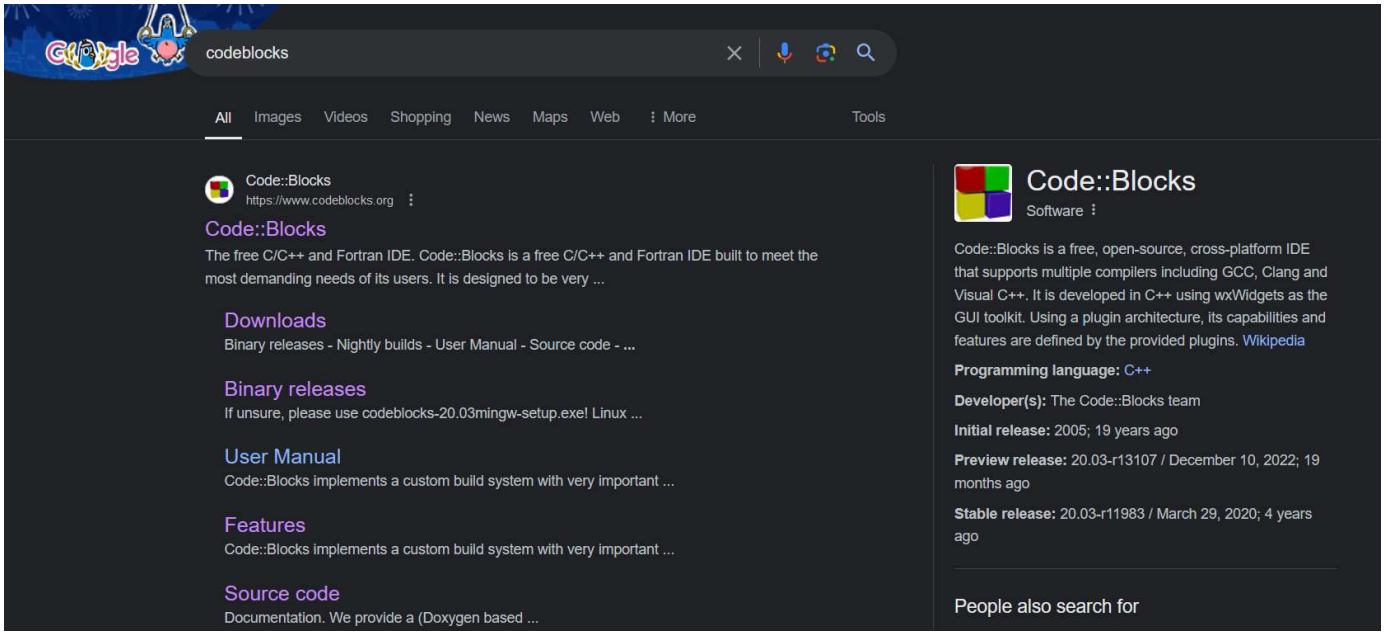
- Code::Blocks
- Visual Studio
- DEV C++
- CLion
- NetBeans
- Eclipse
- CodeLite
- Xcode
- Visual Studio

We will use “Code::Blocks” in this course

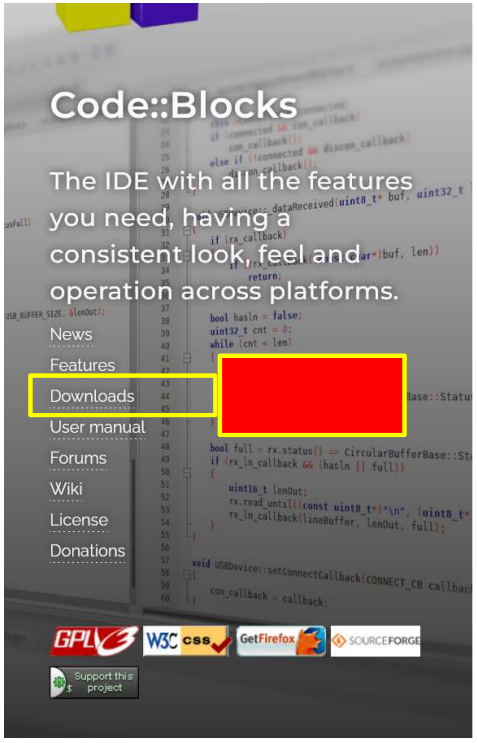
Instruction for Installing Code Blocks and Run the First Program

https://www.youtube.com/watch?v=onJUE1x6nqQ

Step 1: Open Your Web Browser
Step 2: Go to the Search Panel and Search for “Code Blocks”
Step 3: Click on the First Result shown



Step 4: Click on the “Downloads” Section.



Code::Blocks

Code::Blocks

The free C/C++ and Fortran IDE.

Code::Blocks is a free C/C++ and Fortran IDE built to meet the most demanding needs of its users. It is designed to be very extensible and fully configurable.

Built around a plugin framework, Code::Blocks can be extended with plugins. Any kind of functionality can be added by installing/coding a plugin. For instance, event compiling and debugging functionality is provided by plugins!

If you 're new here, you can read the **user manual** or visit the **Wiki** for documentation. And don't forget to visit and join our **forums** to find help or general discussion about Code::Blocks.

We hope you enjoy using Code::Blocks!

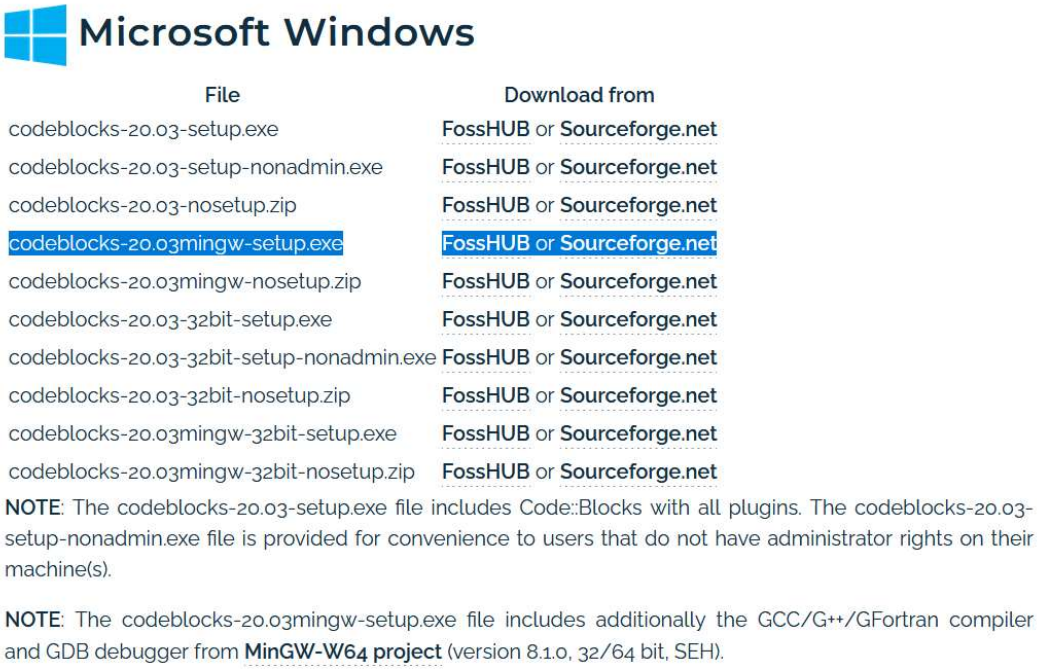
The Code::Blocks Team

Latest news

Migration successful

We are very happy to announce that the process of migrating to the new infrastructure has completed successfully!

Step 5: Click On “Download the binary release”



Microsoft Windows

File	Download from
codeblocks-20.03-setup.exe	FossHUB or Sourceforge.net
codeblocks-20.03-setup-nonadmin.exe	FossHUB or Sourceforge.net
codeblocks-20.03-nosetup.zip	FossHUB or Sourceforge.net
codeblocks-20.03mingw-setup.exe	FossHUB or Sourceforge.net
codeblocks-20.03mingw-nosetup.zip	FossHUB or Sourceforge.net
codeblocks-20.03-32bit-setup.exe	FossHUB or Sourceforge.net
codeblocks-20.03-32bit-setup-nonadmin.exe	FossHUB or Sourceforge.net
codeblocks-20.03-32bit-nosetup.zip	FossHUB or Sourceforge.net
codeblocks-20.03mingw-32bit-setup.exe	FossHUB or Sourceforge.net
codeblocks-20.03mingw-32bit-nosetup.zip	FossHUB or Sourceforge.net

NOTE: The codeblocks-20.03-setup.exe file includes Code::Blocks with all plugins. The codeblocks-20.03-setup-nonadmin.exe file is provided for convenience to users that do not have administrator rights on their machine(s).

NOTE: The codeblocks-20.03mingw-setup.exe file includes additionally the GCC/G++/GFortran compiler and GDB debugger from **MinGW-W64 project** (version 8.1.0, 32/64 bit, SEH).

5.2.3 Code::Blocks Configuration

Go to your Compiler settings:

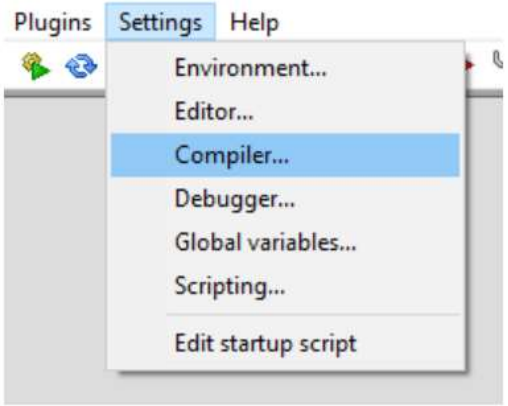


Figure 5.1: Compiler Settings

And then under the "Toolchain executables" tab (red arrow), click on the ellipsis ("...", blue arrow) and choose the root directory where you installed MinGW (64-bit here). Once you have that directory chosen, in the "Program Files" sub-tab (green arrow) area fill out the fields as shown. If you aren't using the MinGW 64-bit toolchain there might be minor variation in the executable names. If you choose the blue arrow ellipsis first then for each ellipsis you click on under "Program Files" you will already be in your MinGW 64-bit bin directory where the actual programs are.

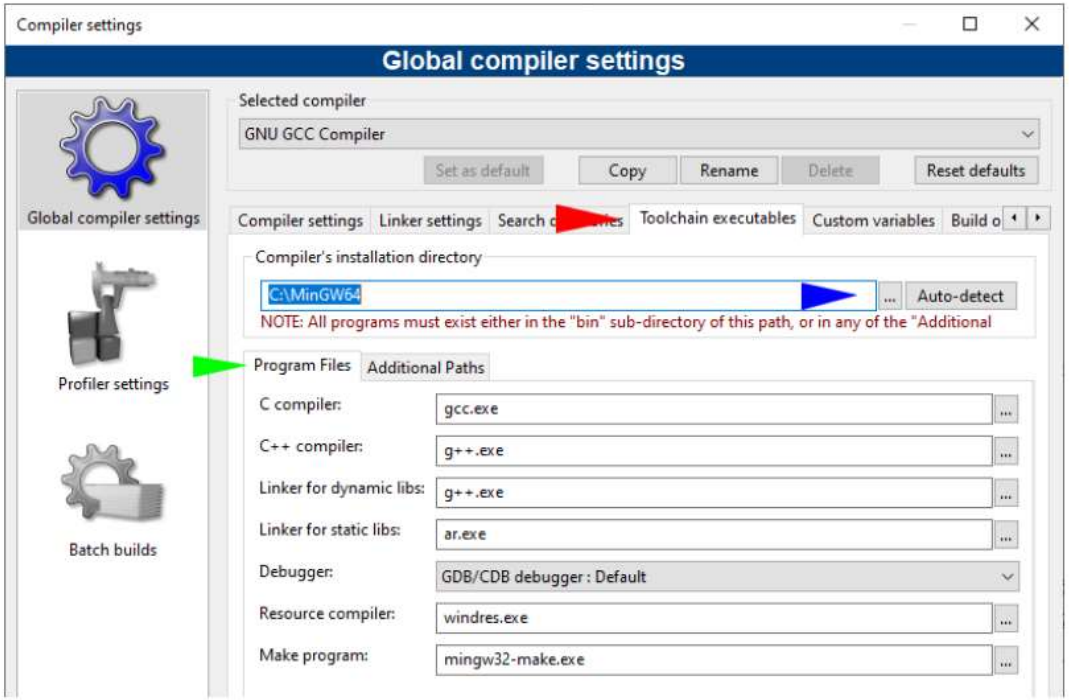


Figure 5.2: Code::Blocks Toolchain Configuration

Now, go to your Debugger settings:

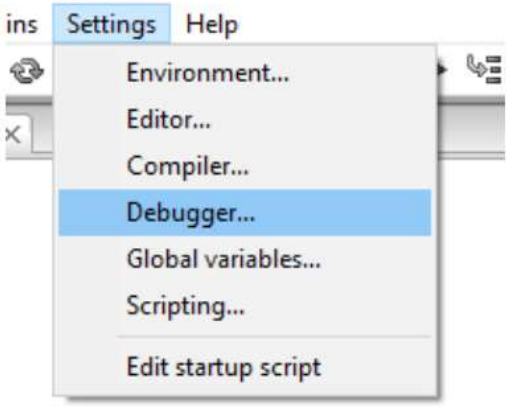
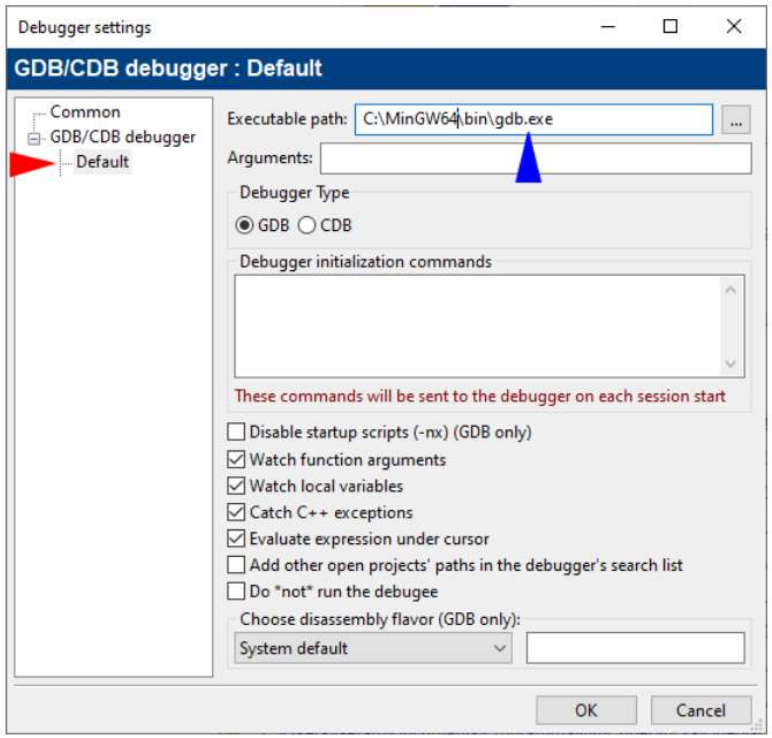


Figure 5.4: Settings Debugger

Choose your default debugger (red arrow), and then fill in the Executable path for it as shown for MinGW 64-bit (blue arrow).



Thanks