1. Introduction To Computer

A computer is an electronic device that is designed to perform various operations automatically or under the control of instructions stored in its memory. It can accept input, process data according to predefined rules or algorithms, and produce output based on the processed data. Computers are used for a variety of tasks such as data processing, calculations, communication, entertainment, and more.

Characteristics of Computer:

Computers possess several key characteristics that define their capabilities and functionalities. Here are some fundamental characteristics of computers:

1. **Speed:**
    - Computers can perform tasks at incredible speeds, processing data and executing instructions in fractions of a second.
2. **Accuracy:**
    - Computers are highly accurate in executing instructions and calculations, minimizing errors that may arise from human factors.
3. **Versatility:**
    - Computers can perform a wide range of tasks and handle diverse types of information, from simple calculations to complex data processing and analysis.
4. **Storage:**
    - Computers can store vast amounts of data in various forms, including text, images, videos, and more. This data can be retrieved and accessed quickly when needed.
5. **Automation:**
    - Computers can automate repetitive tasks, allowing for increased efficiency and productivity. This is particularly evident in manufacturing, business processes, and data analysis.
6. **Diligence:**
    - Computers can maintain a high level of performance without experiencing fatigue or a decline in accuracy, making them suitable for continuous and repetitive tasks.
7. **Reliability:**
    - When properly maintained, computers are reliable in terms of consistent performance. However, they may be susceptible to hardware failures or software issues.
8. **Multitasking:**
    - Modern computers are capable of handling multiple tasks simultaneously, allowing users to run various applications and processes concurrently.
9. **Connectivity:**
    - Computers can communicate with each other and with external devices through networks. This connectivity enables data sharing, collaboration, and access to online resources.
10. **Scalability:**

- Computers can be easily scaled up in terms of processing power, memory, and storage capacity to accommodate growing computational needs.

11. **Precision:**
    - Computers can perform precise calculations and operations, making them essential in scientific, engineering, and research applications.
12. **Digital Processing:**
    - Computers operate in the digital domain, processing data in the form of binary code (0s and 1s). This digital nature allows for accurate representation, storage, and manipulation of information.
13. **Programmability:**
    - Computers can be programmed to perform specific tasks by executing sequences of instructions. This programmability allows for flexibility and adaptation to various applications.
14. **Cost-Effectiveness:**
    - In many cases, computers offer cost-effective solutions for tasks that would be time-consuming or impractical for humans to perform manually.
15. **User Interface:**
    - Computers provide user interfaces, including graphical user interfaces (GUIs) and command-line interfaces, to facilitate interaction between users and the computer system.

Understanding these characteristics helps in appreciating the versatility and impact of computers across different industries and aspects of daily life.

Different Areas of Application of Computer

Computers play a crucial role in various domains, and their applications span across numerous fields. Here are some different areas of application of computers:

1. **Business and Finance:**
    - Accounting and financial management.
    - Electronic transactions and online banking.
    - Market analysis and data modelling.
2. **Education:**
    - E-learning platforms and online courses.
    - Educational software and simulations.
    - Student information systems and administration.
3. **Healthcare:**
    - Electronic health records (EHR) and patient management.
    - Medical imaging and diagnostics.
    - Drug discovery and research.
4. **Entertainment:**
    - Video games and virtual reality.
    - Streaming services for music, movies, and TV shows.
    - Digital content creation and editing.
5. **Communication:**
    - Email and instant messaging.

  - Video conferencing and collaboration tools.
  - Social media platforms.
6. **Science and Research:**
  - Data analysis and simulation.
  - Scientific modeling and visualization.
  - Research collaboration through online platforms.
7. **Manufacturing and Industry:**
  - Computer-aided design (CAD) and manufacturing (CAM).
  - Automation and control systems.
  - Supply chain management and logistics.
8. **Agriculture:**
  - Precision farming and agricultural automation.
  - Crop monitoring and management systems.
  - Weather prediction and analysis.
9. **Government:**
  - E-governance and digital services.
  - Geographic Information System (GIS) for mapping and planning.
  - Public safety and security systems.
10. **Transportation:**
  - Traffic control and management.
  - Vehicle navigation and tracking systems.
  - Logistics and supply chain optimization.
11. **Space Exploration:**
  - Satellite communication and control.
  - Data analysis for space research.
  - Simulation and modeling for mission planning.
12. **Artificial Intelligence and Machine Learning:**
  - Natural language processing and chatbots.
  - Predictive analytics and recommendation systems.
  - Image and speech recognition.
13. **Environmental Monitoring:**
  - Climate modeling and simulation.
  - Remote sensing and satellite data analysis.
  - Pollution monitoring and control.
14. **Cybersecurity:**
  - Intrusion detection systems.
  - Encryption and secure communication.
  - Threat intelligence and analysis.
15. **Personal Use:**
  - Personal productivity tools (word processing, spreadsheets, etc.).
  - Internet browsing and online shopping.
  - Home automation and smart devices.

These examples highlight the diverse and widespread impact of computers on various aspects of modern life. The continued advancement of technology is likely to bring even more innovative applications in the future.

Short History of Computer:

The history of computers can be traced back to the invention of the abacus, a simple calculating tool used by ancient civilizations. However, modern computers as we know them today are a product of the 20th century and have evolved over time to become faster, more powerful, and more efficient.

In the 1800s, inventors developed machines capable of performing mathematical calculations. The first such device was Charles Babbage's Analytical Engine, a mechanical calculator that could perform complex calculations automatically. Although it was never built in his lifetime, his ideas laid the foundation for modern computing and therefore he is known as father of Computer.

The first electronic computers were developed in the 1930s and 1940s, with notable examples including the Atanasoff-Berry Computer (ABC), the Colossus, and the Harvard Mark I. These early computers were used mainly for scientific and military purposes, such as codebreaking during World War II.

In the 1950s and 1960s, the first commercial computers were introduced, including the IBM 701 and the UNIVAC I. These computers were expensive and only affordable for large businesses and government agencies.

The invention of the microprocessor in the early 1970s made it possible to create smaller, cheaper computers that were accessible to a wider range of users. This led to the development of personal computers (PCs) in the late 1970s and early 1980s, such as the Apple II and the IBM PC.

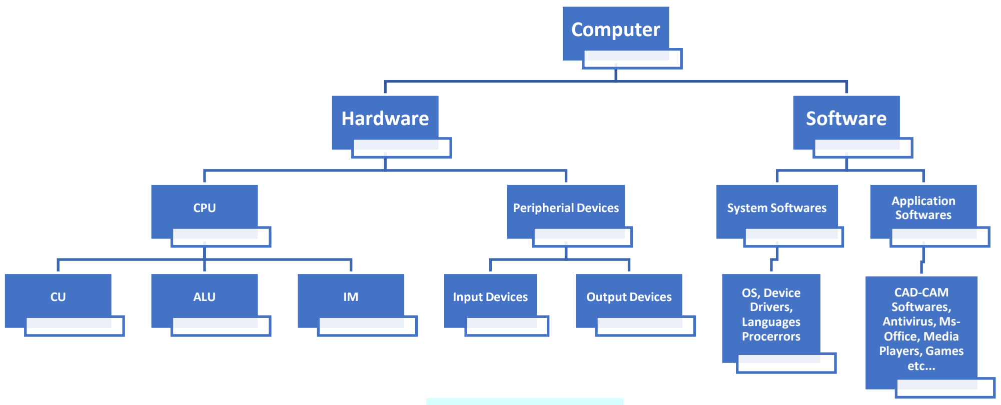
The 1990s saw the rise of the internet and the World Wide Web, which transformed the way people communicate, work, and access information. This era also saw the development of more powerful and sophisticated computers, including laptops and smartphones.

Hierarchical and Functional block diagram of Computer:

Basically, computer is known as a data processing unit. It will take data as input and provides the well processed data as output, known as information.

A hierarchical diagram of a computer is a visual representation of the different components of a computer system organized in a hierarchical structure. The diagram shows how the components of a computer system are interconnected and work together to perform various functions.

Here's an example of a hierarchical diagram of a computer system:



Hardware: Computer hardware refers to the physical components and devices that make up a computer system. Computer hardware is designed to work together to perform tasks such as running software applications, processing data, and displaying information to users. These include:

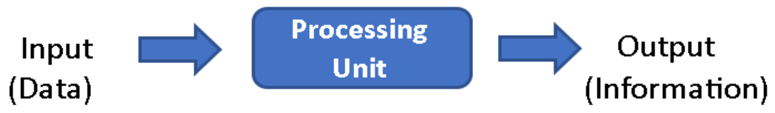
- **CPU:** (Central Processing Unit) - also known as the processor, it is the brain of the computer that performs calculations and instructions. There are three main parts of CPU.
    - **Control Unit (CU):** The Control Unit (CU) is a component of the Central Processing Unit (CPU) that manages and coordinates the operations of the entire CPU. Its main function is to control the flow of data between the CPU and other parts of the computer system, including input and output devices, memory, and storage. The CU fetches instructions from memory and interprets them to determine the appropriate operations to be performed by the CPU. CU manages the execution of the computer's instruction set by controlling the sequence of operations performed by the CPU
    - **Arithmetic and Logical Unit (ALU):** The Arithmetic Logic Unit (ALU) is a component of the Central Processing Unit (CPU) that is responsible for performing arithmetic and logical operations on data. It is considered the "heart" of the CPU because it is responsible for carrying out the actual computational tasks. The ALU performs basic arithmetic operations such as addition, subtraction, multiplication, and division. It also performs logical operations such as AND, OR, NOT, and XOR. These operations are performed on binary numbers, which are made up of 1s and 0s.
    - **Internal Memory (IM):** The internal memory in the CPU refers to the memory storage units that are directly integrated within the CPU chip. the internal memory in the CPU plays a critical role in the CPU's operation by providing fast access to frequently used data and instructions. By storing this data and instructions internally, the CPU can reduce the amount of

time it takes to retrieve the data and execute instructions, resulting in faster processing speeds.

- **Peripheral Devices:** Peripheral devices are external hardware devices that are connected to a computer and are used to input, output, and store data. They can be divided into three main categories: input devices, output devices, and storage devices.
    - **Input devices:** These are devices that allow the user to input data into the computer. Examples of input devices include:
        - ✓ **Keyboard:** Used to input text and commands into the computer.
        - ✓ **Mouse:** Used to navigate the cursor and select items on the computer screen.
        - ✓ **Scanner:** Used to scan and digitize physical documents or images.
        - ✓ **Digital cameras:** Used to capture and transfer digital images to the computer.
        - ✓ **Microphone:** Used to input sound or voice into the computer.
    - **Output devices:** These are devices that display or output data from the computer. Examples of output devices include:
        - ✓ **Monitor**: Displays images and text on the computer screen.
        - ✓ **Printer**: Outputs documents or images onto paper.
        - ✓ **Speakers**: Output sound or audio from the computer.
- **Software:** software refers to a set of instructions, programs that are designed to perform specific tasks on a computer. It is a collection of programs, data, and instructions that tell the computer what to do and how to do it. Software can be broadly categorized into two main types: system software and application software.
    - **System software:** This is the software that manages and controls the computer hardware and provides a platform for running application software. It includes the operating system, device drivers, utilities, and other tools that are necessary for the proper functioning of the computer system.
    - **Application software:** This is the software that is designed to perform specific tasks or applications for the user. Examples of application software include word processors, spreadsheets, graphics editors, media players, web browsers, and more.

Software can also be categorized based on its distribution and licensing models. Proprietary software is software that is owned by a company or an individual and cannot be modified or distributed without permission. Open-source software, on the other hand, is software that is freely available to use, modify, and distribute.

Functional Block Diagram of Computer:



Data: Data is facts, properties, mesurements or characteristics of entity under consideration. It refers to any piece of information that a program can process or manipulate. This information can take many forms, such as numbers, text, images, audio, or video. Data can be either input into a program or generated by it as output.

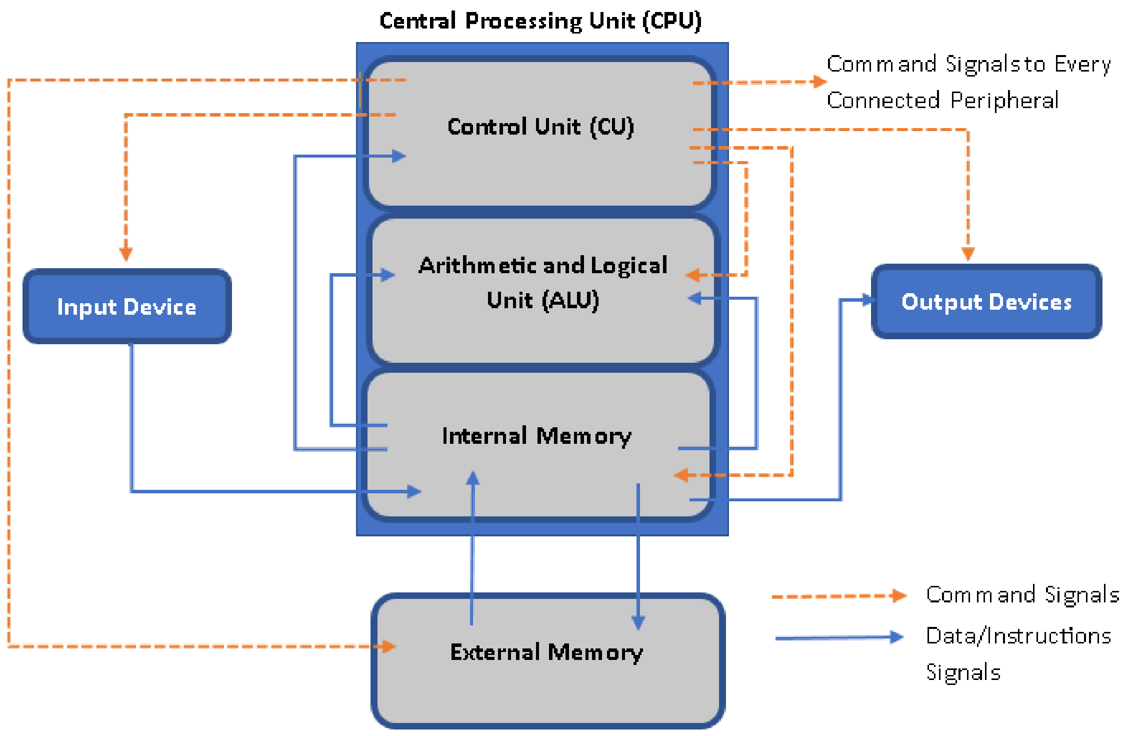
Information: Information is well processed data. It refers to any meaningful or useful knowledge or data. e.g. Personal details, Exam Details, subjects and marks obtained is Data but marksheet is an information.

Computer an be defined as a data processing unit, which converts the data into information. Where data is input to computer and information is well processed data.

Or in another words,

Computer is two state, multipurpose, programmable, electronic device which take data from user, store it, process it and gives the output in desired format.

Let's elaborate the above diagram,


Central Processing Unit (CPU)
Control Unit (CU)
Arithmetic and Logical Unit (ALU)
Internal Memory
External Memory
Input Device
Output Devices
Command Signals to Every Connected Peripheral
Command Signals
Data/Instructions Signals

What is Memory and its different types?

Memory refers to the physical devices used to store data and program code for a computer to access and use. It plays a crucial role in the functioning of a computer system, allowing it to perform tasks and store information temporarily or permanently. Memory is typically categorized into different types based on its characteristics, purpose, and access speed. Here are the main types of memory in a computer system:

```
                      Memory
                        |
              -------------------------
               |                      |
       Internal/Primary         External/Secondary
               |                      |
      -----------------         -------------------------
       |               |         |    |       |         |
Tmp/volatile/   Per/Non-Vola.  HDD   CD     DVD    PD,Memory cards
short-term      long-term
       |               |
     RAM             ROM
   - SRAM              |
   - DRAM    ---------------------------
             |         |      |        |
          Simple    PROM    EPROM  EEPROM
           ROM
```

1. Primary Memory

It is also known as main memory in computer, it communicates directly with the CPU, Cache and Auxiliary memory. This type of computer memory keeps data and programs when the process is active to use them.

When a program or the data is activated for execution, the processor loads instructions from the secondary memory into the main memory and then starts execution. It is a volatile memory due to which any unsaved data is lost when a power cut occurs. Primary memory is of two types: RAM and ROM.

1.1 RAM

RAM is hardware that temporarily stores data and programs. It is the faster part of the main memory which can be directly accessed by the CPU. It reads and writes programs until the computer is switched on. RAM is of two types: DRAM and SRAM.

- DRAM full form is Dynamic Random-Access Memory. It is a type of RAM that is used for dynamic data storage. Every cell in DRAM consists of one-bit information. A cell is composed of a transistor and a capacitor. This capacitor and transistor are extremely small in size. The capacitor needs a continuous refresh to retain information since it is volatile.
- SRAM full form is Static Random-Access Memory. This type of RAM stores static data in memory which remains active until there is a power supply. Same sized SRAM chip holds less data than DRAM. Unlike DRAM, it does not require a continuous refresh.

1.2 ROM

Read-Only Memory (ROM) is a permanent storage type. This is a type of read-only memory that only reads the stored information, but it does not have the capability to modify or write. Since it is a non-volatile type of memory in computer, the information stays even after a power cut or when the system has been shut down. ROM is of the following five types:

- Simple ROM: It is the oldest ROM whose data is pre-configured via integrated circuit manufacture during the time of manufacturing. Due to this pre-configuration, the user cannot change the instruction stored within the Simple ROM chip.
- PROM: It is a digital ROM which only once allows writing any information or program. This is done using a special PROM programmer or burner device.
- EPROM: In this type of ROM, data can be erased as well as reprogrammed only once. It can store data for a minimum of 10-20 years. To erase and reprogram EPROM, the user needs to pass UV light for 40 minutes. Post this, the data can be recreated.
- EEPROM: The full form of EEPROM is Electrically Erasable and Programmable Read Only Memory. It is an electrically erasable and programmable ROM. This allows data to be erased using a high-voltage electrical charge. After this, it can be reprogrammed up to thousands of times.

2. Secondary Memory

It is a permanent type of memory in computer that holds a large amount of data. This is an external memory that represents different storage media on which data and programs can be saved for long term. It is not directly accessible by the CPU and is available as external devices such as CDs, DVDs and USBs. They are cheaper than primary memory but slower than primary memory.

- Hard disk: It is a type of permanent computer memory that stores programs, files and data. It is stored on the motherboard of the computer that does not lose data even when there is a power outage or when the system has been switched off.
- Compact Disc (CD): It is an optical disk storage device that stores different types of data, such as audio, video, files, and other information. CD uses light to read and write data from CDs.
- Pen Drive: This portable device is a type of secondary memory in computer that is used for permanently storing data. It is also known as a USB flash drive that stores and transfers.

Memory Read-Write Operation

When we enter the data, that data flows in terms of electrical signals throughout the system. But when we want to store that data, we need to store it in the magnetic form because we know that we are not capable of storing the electrical signals. We have different number systems but binary is suitable, because we consider electrical high signal as a one and in the memory, it is a represented by charged sector, and the low signal considered as zero and it is represented by uncharted sector in the memory. Due to this convenience in the mapping, binary is preferred in the computers. As every character associated with a fixed binary string the read write operation is possible and also as these conventions are accepted all over the world the information interchange become very easy and therefore these codes are known as

ASCII codes that is "American Standard Codes for Information Interchange." The diagram below shows how a read write operation happened actually in the memory.

```
                          A

                      10011010    (assume it as a ascii of character A)

                  __/\__/\/\_/\_
                 _________________________________________________________
                | @--@@-@-                                                |
     Write      |                                                         |
  ------------  |                                                         |
     Read       |                                                         |
                |                                                         |
                |                                                         |
                |_________________________________________________________|

                  __/\__/\/\_/\_

                   10011010

                       A
```

Memory Units: Memory in a computer is measured in various units, each representing a different scale of storage capacity. Here are the common units used to measure memory:

SYMBOL	QUANTITY
1 BIT	BINARY 0 OR 1
4 BITS	1/2 BYTE
8 BITS	1 BYTE
1024 BYTE	1 KILOBYTE
1024 KILOBYTE	1 MEGABYTE
1024 MEGABYTE	1 GIGABYTE
1024 GIGABYTE	1 TERABYTE
1024 TERABYTE	1 PETABYTE
1024 PETABYTE	1 HEXABYTE
1024 HEXABYTE	1 ZEETABYTE

What is Programming Language?

A programming language is a set of instructions and rules used to communicate with a computer in order to create programs and software applications. It allows programmers to write code using a structured and standardized syntax, which is then converted into machine code that a computer can understand and execute.

Programming languages vary in complexity and purpose, with some being designed for specific tasks or platforms, while others are more general-purpose and can be used for a variety of applications. Examples of popular programming languages include Java, Python, C++, JavaScript, and Ruby.

In addition to providing a means for programmers to communicate with computers, programming languages also allow for collaboration and sharing of code between developers. This has led to the development of vast libraries of pre-written code, making it easier for programmers to build complex applications more quickly and efficiently.

History Of Programming Languages

The history of programming languages dates back to the mid-1800s, when mathematician Ada Lovelace wrote what is considered to be the first computer program for Charles Babbage's Analytical Engine. However, it was not until the mid-20th century that programming languages began to take shape as we know them today.

The first high-level programming language was FORTRAN (FORmula TRANslation), developed in the 1950s by IBM for scientific and engineering applications. This was followed by COBOL (Common Business Oriented Language), which was designed for business applications.

In the 1960s, programming languages continued to evolve with the development of languages such as BASIC (Beginner's All-purpose Symbolic Instruction Code) and PL/I (Programming Language One), which were designed to be more user-friendly and versatile.

In the 1970s, C and Pascal were developed, which became popular for systems programming and education respectively. The 1980s saw the development of object-oriented programming languages such as Smalltalk and C++, which allowed for more complex and efficient code.

The 1990s saw the rise of scripting languages such as Perl and Python, as well as the development of Java, which became popular for web-based applications. The 2000s and 2010s saw the development of languages such as Ruby, Swift, and Go, as well as the continued popularity of languages such as C++, Java, and Python.

```
              SIMULA
              COBOL                                                     |-- Sun Microsystem --> Java
Assembly --> Fortran ----> ALGOL60 --> CPL --> BCPL-----> B -----> C ----> C++ ====|-- Microsoft corpo.--> .Net
              RPG            (1960)    (1963)    (1967)    (1970)   (1972) (1983-84) |-- .......
              BASIC                                                     |
              Pascal
              ....
```

Today, there are hundreds of programming languages in use, each with its own strengths and weaknesses, and new languages continue to be developed to meet the needs of specific applications and industries.

Language Processors and Its Types:

A language processor is a type of software tool that translates programming languages into machine-readable code that can be executed by a computer. There are three main types of language processors: compilers, interpreters, and assemblers.

1. **Compilers:** A compiler is a program that translates the entire source code of a program into machine code in one go. The output of the compiler is an executable file that can be run on a computer without the need for the original source code. Examples of languages that use compilers include C, C++, and Java.
2. **Interpreters:** An interpreter is a program that reads and executes the source code of a program line by line. The interpreter translates each line of code into machine code

and executes it before moving on to the next line. Examples of languages that use interpreters include Python, JavaScript, and Ruby.

3. **Assemblers:** An assembler is a program that translates assembly language, which is a low-level programming language that uses mnemonics to represent machine instructions, into machine code. Assembly language is often used in systems programming, where direct hardware access and control is required.

In addition to these main types of language processors, there are also hybrid tools that combine elements of both compilers and interpreters. **Just-in-time (JIT) compilers**, for example, dynamically compile and execute code at runtime, combining the speed of compiled code with the flexibility of interpreted code. Another example is the use of bytecode, which is an intermediate form of code that can be interpreted by a virtual machine, as used in languages such as Java and Python.

Editor and IDE:

An editor is a software tool that allows a programmer to create and modify source code for a program. It typically provides basic text-editing features such as syntax highlighting, indentation, and search and replace. An Integrated Development Environment (IDE) is a more advanced type of software tool that combines an editor with additional features such as code completion, debugging, testing, and version control.

Here are some commonly used editors and IDEs for popular programming languages:

1. Python: IDLE (built-in Python IDE), PyCharm, Visual Studio Code, Jupyter
2. Java: Eclipse, NetBeans, IntelliJ IDEA, Visual Studio Code
3. JavaScript: Visual Studio Code, Atom, Sublime Text, WebStorm
4. C/C++: Visual Studio Code, Code::Blocks, Eclipse, NetBeans, CLion
5. PHP: Visual Studio Code, PhpStorm, Sublime Text, Atom
6. Ruby: Visual Studio Code, RubyMine, Sublime Text, Atom
7. HTML/CSS: Visual Studio Code, Atom, Sublime Text, Brackets
8. Swift: Xcode (built-in IDE for Mac), Visual Studio Code

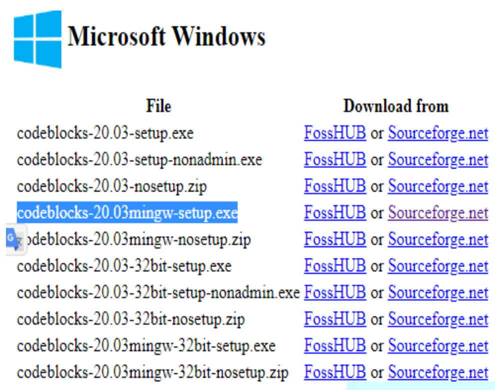
These are just a few examples, and there are many other editors and IDEs available for different programming languages. The choice of editor or IDE often comes down to personal preference, as well as the specific features and capabilities required for the programming task at hand.

Installation of Code::Blocks IDE:

To install Code::Blocks on Windows for C and C++ programming, follow these steps:

1. Visit the official Code::Blocks website: http://www.codeblocks.org/.
2. Click on the "Download" link on the homepage to navigate to the download page.

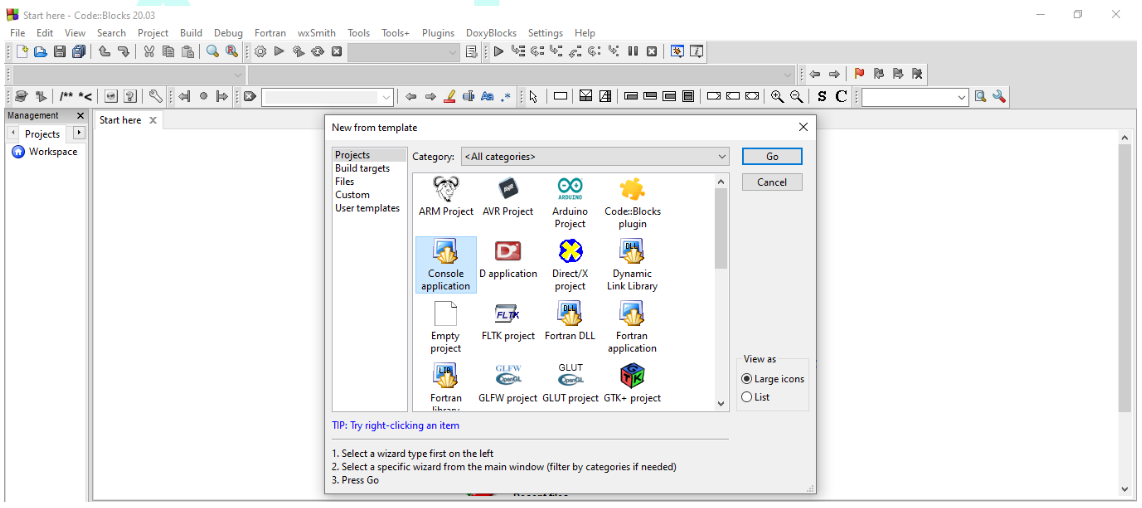
3. On the download page, you'll find various installers available for different operating systems. Locate the "Windows XP/Vista/7/8.x/10" section and click on the link corresponding to the version you want to download (e.g., codeblocks-20.03mingw-setup.exe).



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

4. Once the installer is downloaded, run the executable file.
5. The installation wizard will guide you through the installation process. Click "Next" to proceed.
6. Accept the license agreement and click "Next."
7. Choose the installation directory or accept the default location, and click "Install."
8. Once the installation is complete, you will see a "Completing the Code::Blocks Setup Wizard" screen. Make sure the "Launch Code::Blocks" option is checked, and click "Finish."
9. Code::Blocks will launch, and you can start using it for C and C++ programming.
10. Select File Menu →New →Project → Select Console Application → Go → Next

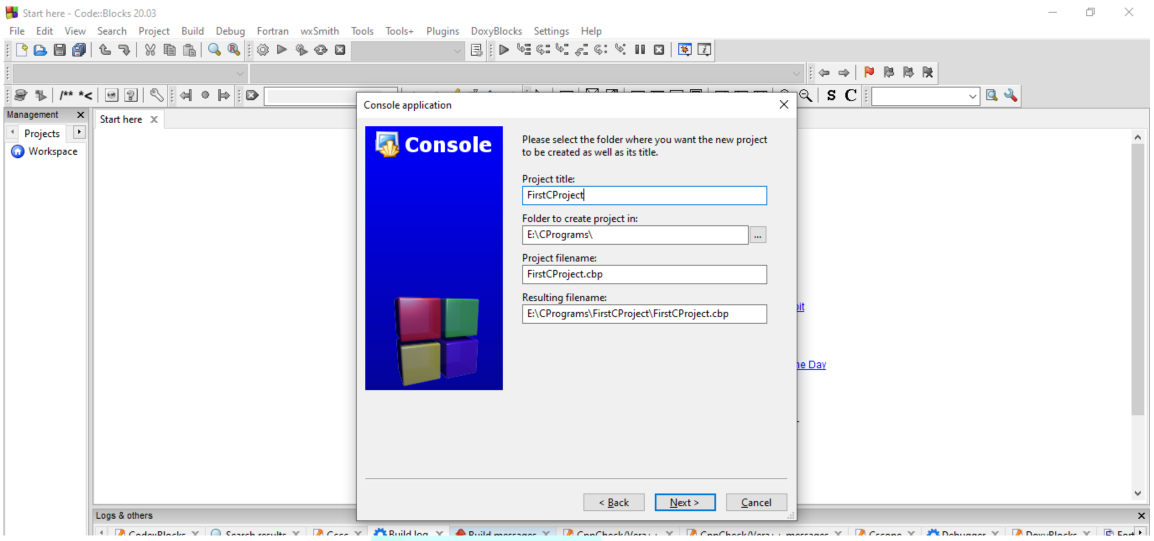


TIP: Try right-clicking an item

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

11. Again, select the C language → Next

12. Write a Project Title and browse Folder to create project in (Location to store the project, do not change any other textboxes) and click on the Next


Start here - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Management
Projects
Workspace
Start here
Console application
Console
< Back
Next >
Cancel
Logs & others

Please select the folder where you want the new project to be created as well as its title.

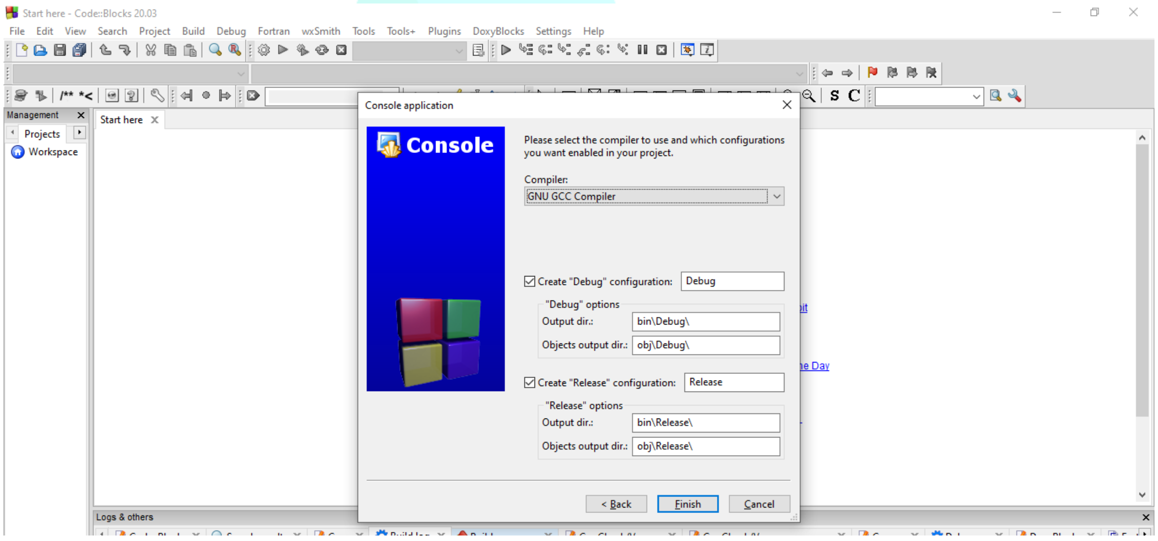
Project title: FirstCProject

Folder to create project in: E:\CPrograms\

Project filename: FirstCProject.cbp

Resulting filename: E:\CPrograms\FirstCProject\FirstCProject.cbp

13. You will get the *GNU GCC Compiler* selected as a default compiler. → finish


Start here - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Management
Projects
Workspace
Start here
Console application
Console
< Back
Finish
Cancel
Logs & others

Please select the compiler to use and which configurations you want enabled in your project.

Compiler: GNU GCC Compiler

☑ Create "Debug" configuration: Debug

"Debug" options

Output dir.: bin\Debug\

Objects output dir.: obj\Debug\

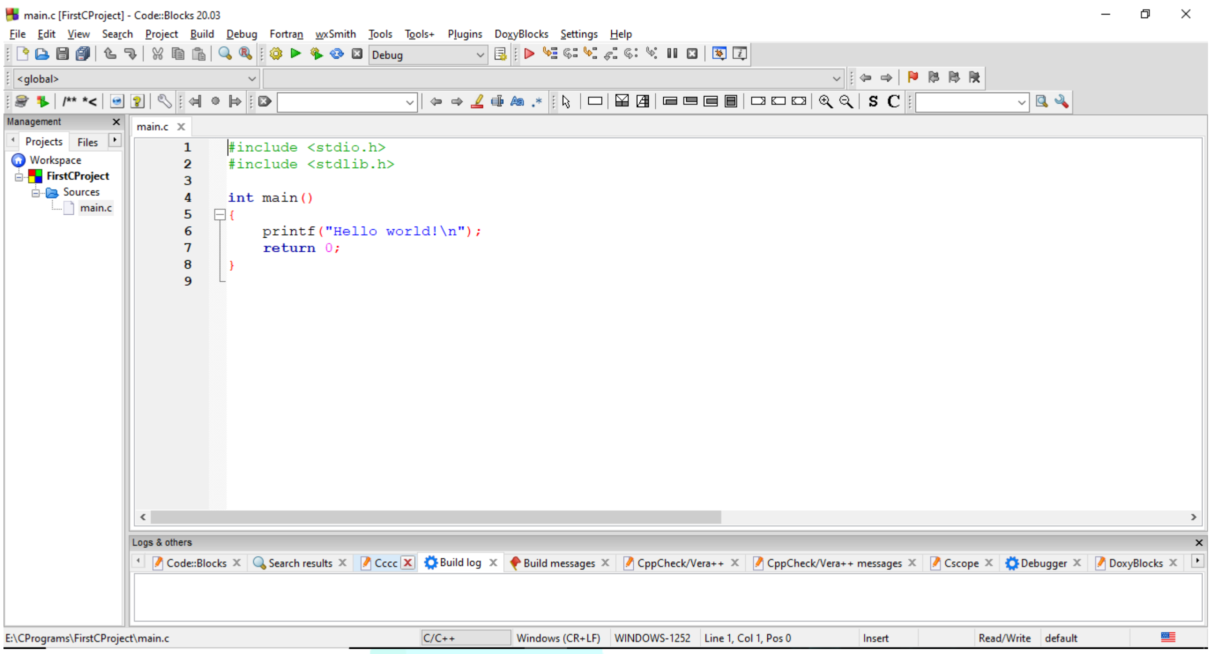
☑ Create "Release" configuration: Release

"Release" options

Output dir.: bin\Release\

Objects output dir.: obj\Release\

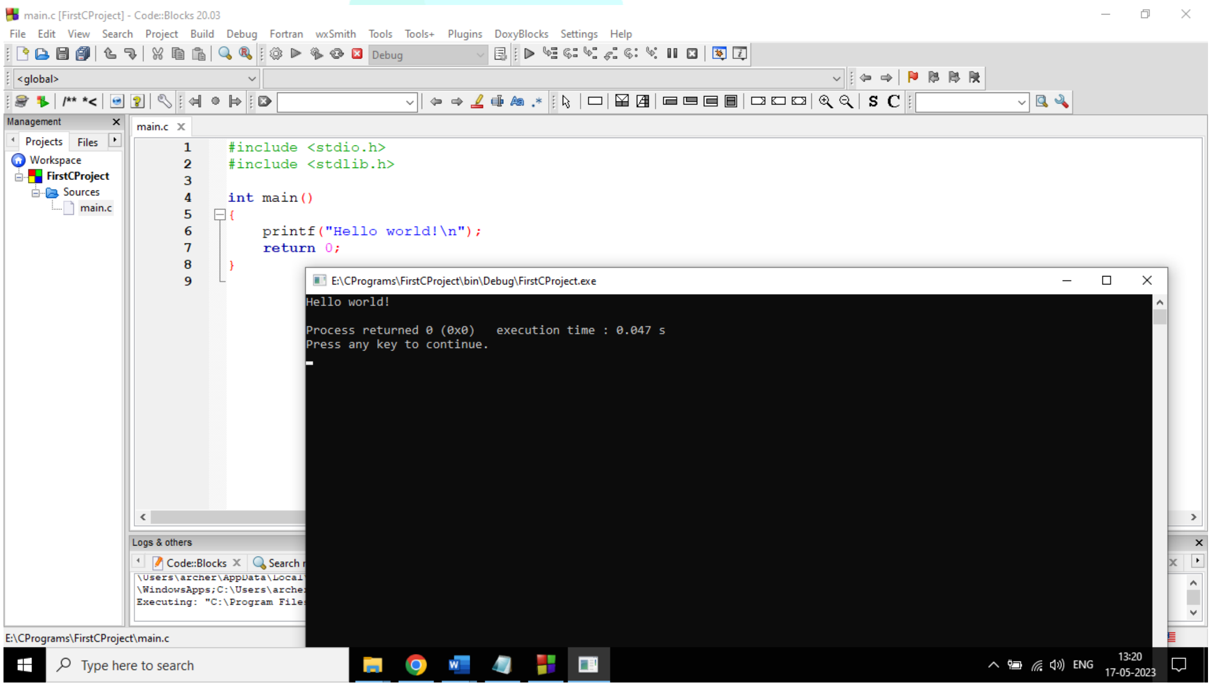
14. Explore the source folder from project management pane at left and double click on main to get the editor.


main.c [FirstCProject] - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Management
Projects Files
Workspace
FirstCProject
Sources
main.c
Logs & others
Code::Blocks Search results Cccc Build log Build messages CppCheck/Vera++ CppCheck/Vera++ messages Cscope Debugger DoxyBlocks
E:\CPrograms\FirstCProject\main.c C/C++ Windows (CR+LF) WINDOWS-1252 Line 1, Col 1, Pos 0 Insert Read/Write default

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    printf("Hello world!\n");
    return 0;
}
```

15. You will get the default code. Just click on built and then Run to execute code


main.c [FirstCProject] - Code::Blocks 20.03
E:\CPrograms\FirstCProject\bin\Debug\FirstCProject.exe
Type here to search
ENG
13:20
17-05-2023

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
    printf("Hello world!\n");
    return 0;
}
```

```
Hello world!

Process returned 0 (0x0)   execution time : 0.047 s
Press any key to continue.
```

Execution Flow of C Program:

When a C program is executed, it goes through a series of steps known as the execution flow. Here is a detailed explanation of the execution flow of a typical C program:

1. Pre-processing:

   - The pre-processor scans the source code for directives starting with a '#' symbol, such as #include and #define.
   - It processes these directives and performs tasks like including header files, macro substitution, and conditional compilation.

2. Compilation:

    - The pre-processed code is then passed to the compiler, which translates the code into assembly language or machine code.
    - The compiler performs lexical analysis, syntax analysis, semantic analysis, and generates an object file containing machine code instructions.

```
        Instruction             Program_Name
                |________________|
                        |
Source code         *.c/*.cpp <------Debug--------|
                        |                         |
                        |                         |
                      Compile ------------------->| compile-time error
                        |                         | (systax error)
                        |                         |
Backup file          *.bak                        |
                        |                         |
                        |                         |
        Linker -------->|  ---------------------->| Linker error
                        |                         |
object code file     *.obj                        |
                        |                         |
                        |                         |
Executable file       *.exe                       |
                        |                         |
                       RUN ---------------------->| Runtime error
                        |
                     Output
```

3. Linking:

    - If the program uses external libraries or functions, the linker combines the object file with the necessary libraries to create an executable file.
    - It resolves references to functions or variables defined in other files or libraries.

4. Loading:

    - The operating system loads the executable file into memory.
    - It allocates memory for global and static variables and sets up the program's initial execution environment.

5. Program Execution:

    - The program's execution starts at the entry point, usually the "main" function.
    - The operating system transfers control to the "main" function, and the statements inside it begin to execute.

6. Execution of Statements:

- The program executes the statements in a sequential manner, one after another.
- It can also execute statements conditionally based on the result of logical expressions or perform repetitive tasks using loops.

7. Function Calls:
    - If the program encounters a function call, it temporarily suspends the execution of the current function and transfers control to the called function.
    - The called function executes its statements and returns the control back to the caller.
8. Variable Scope and Lifetime:
    - The program creates and destroys variables based on their scope and lifetime.
    - Variables declared inside a block have local scope and are destroyed when the block is exited.
    - Global variables have a longer lifetime and exist throughout the program's execution.
9. Flow Control:
    - The program can alter its execution flow using control statements such as if-else, switch-case, and loops (e.g., for, while, do-while).
    - Based on the conditions and control statements, the program can take different paths and execute different sets of statements.
10. Termination:
    - The program reaches the end of the "main" function or encounters a return statement, which indicates the termination of the program.
    - The operating system reclaims the memory allocated to the program and returns the control to the user or the calling process.

It's important to note that this is a general overview, and the exact execution flow can vary depending on the specific program, operating system, and compiler being used.