

C Computer Science

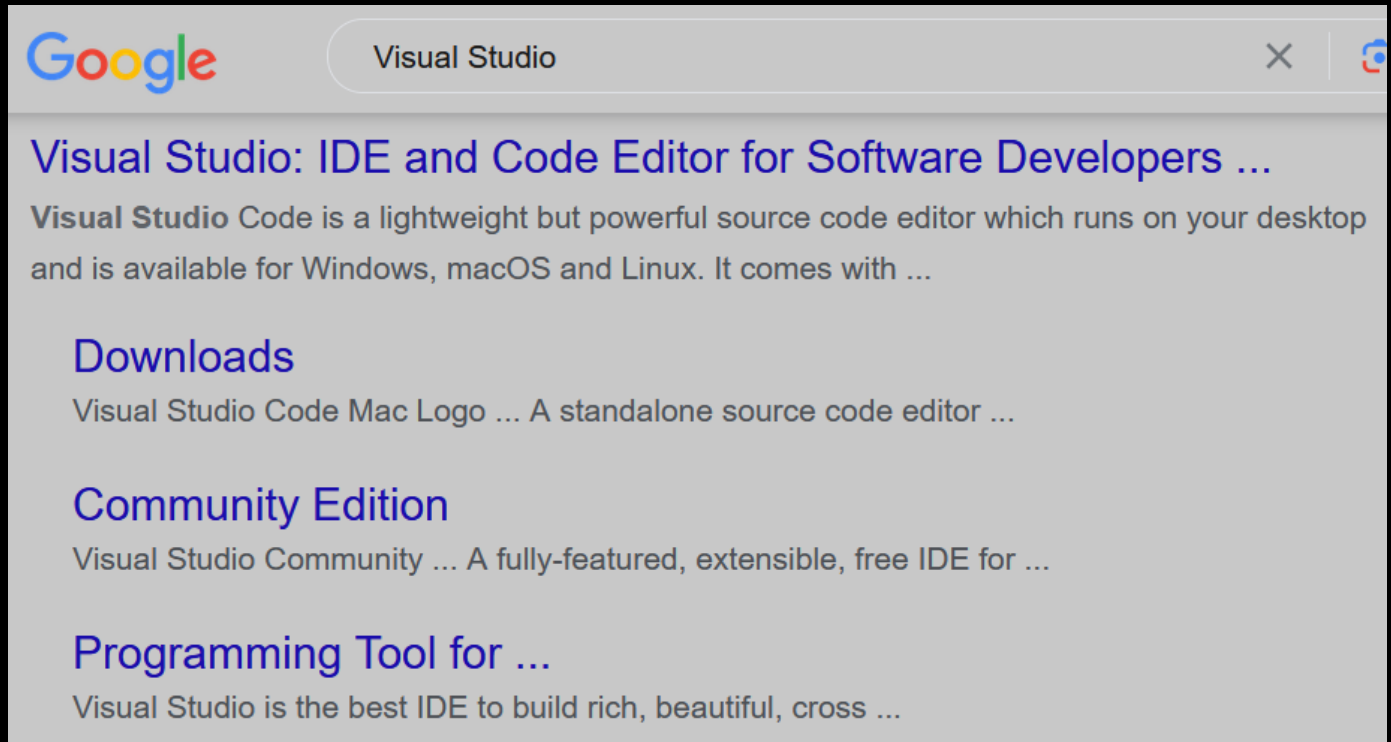
by

Christopher Andrew Topalian

Copyright 2000-2025
All Rights Reserved

Dedicated to God the Father

Download Visual Studio - Search Google



We Go To: google.com

We Search for: Visual Studio

We Left Click on: Downloads

**Or, we can go directly
to the Visual Studio website
as shown on the next page.**

Download Visual Studio - Directly from Website



We Go To:

<https://visualstudio.microsoft.com/vs/community>

We Download: Visual Studio Installer

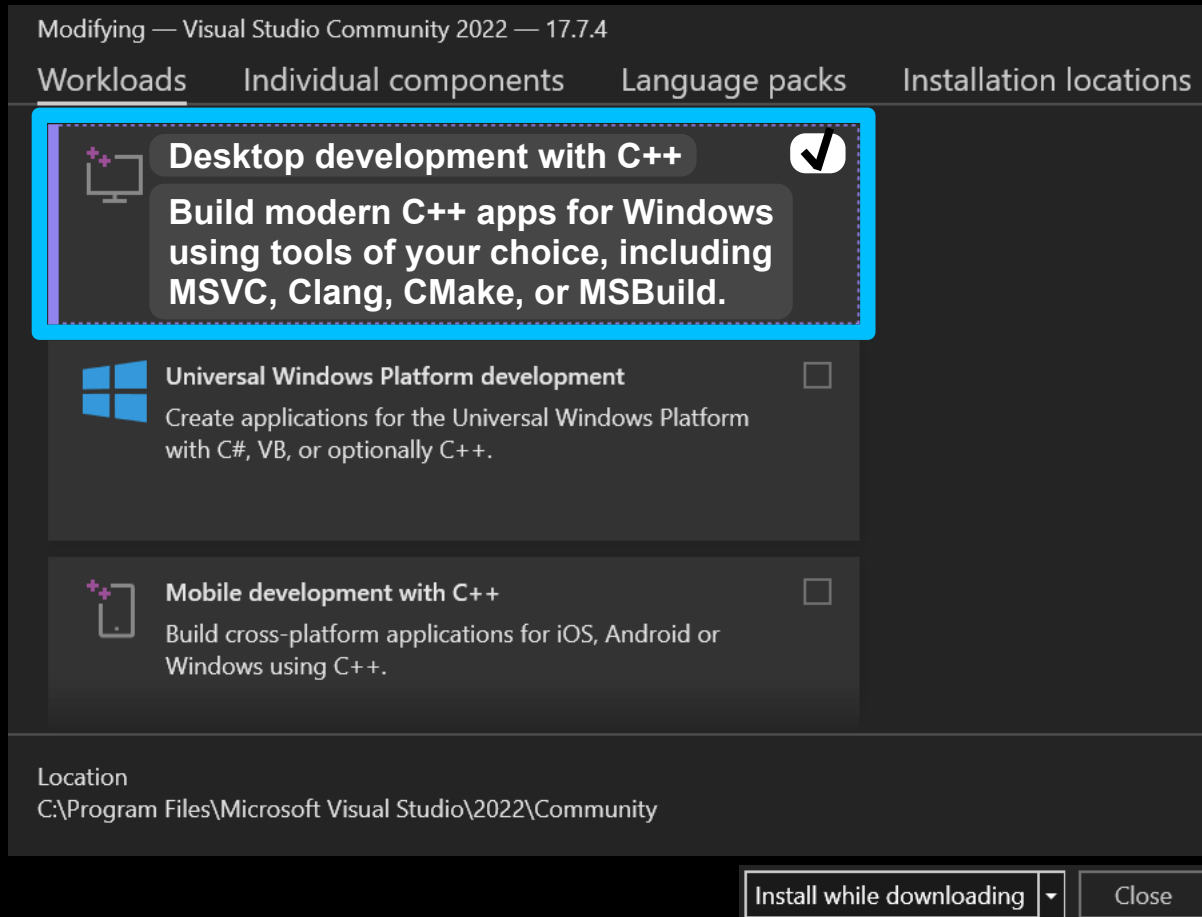
We Go To our Downloads Folder and:

Double Left Click the Install file to Install it.

After it is installed, we can open VS Studio.

Once open, we can then install the C++ package, which has in it, the ability to create C apps too.

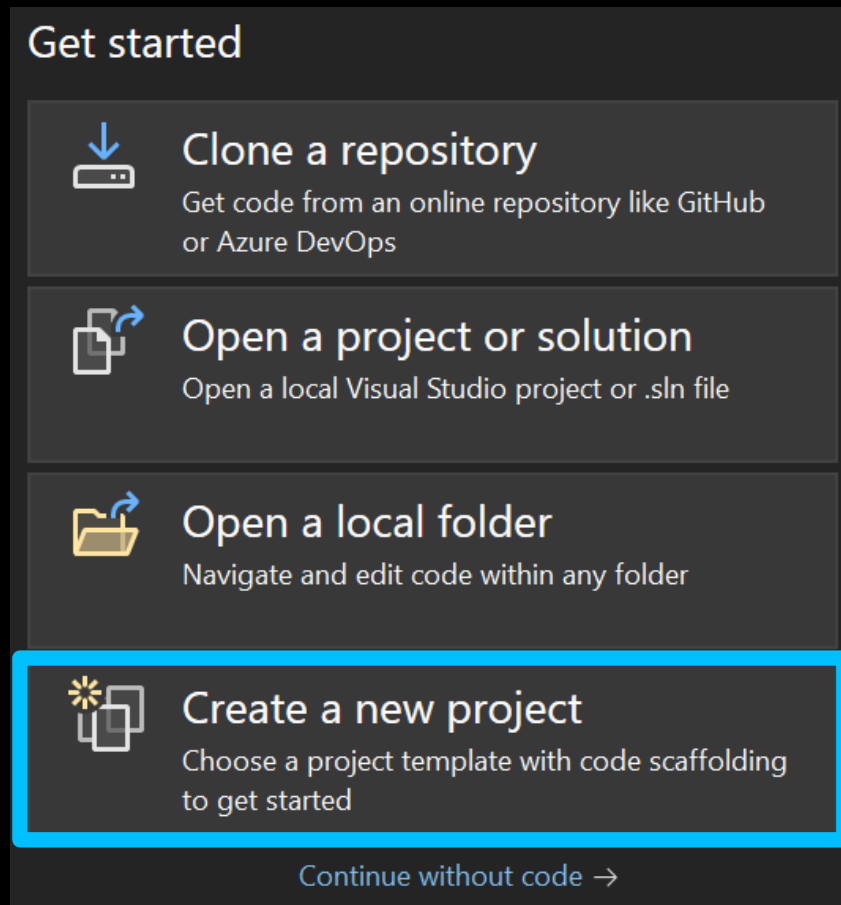
We Download and Install: Desktop development with C++



We Put a Checkmark in the box
and then Left Click the
Install while downloading button

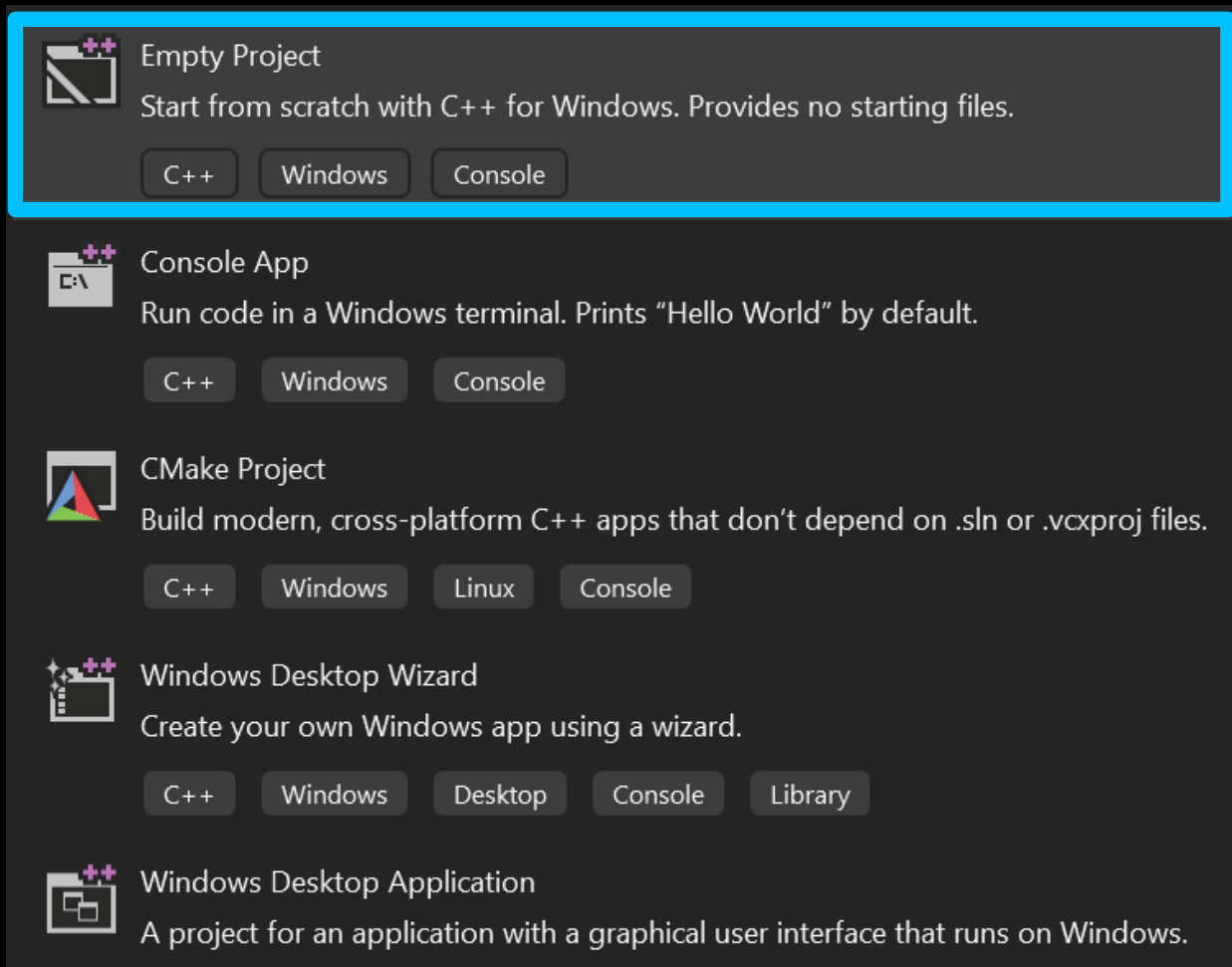
This will download and install the ability to use
Visual Studio to create C++ Desktop
Applications, but also, C applications too.

Create a New Project

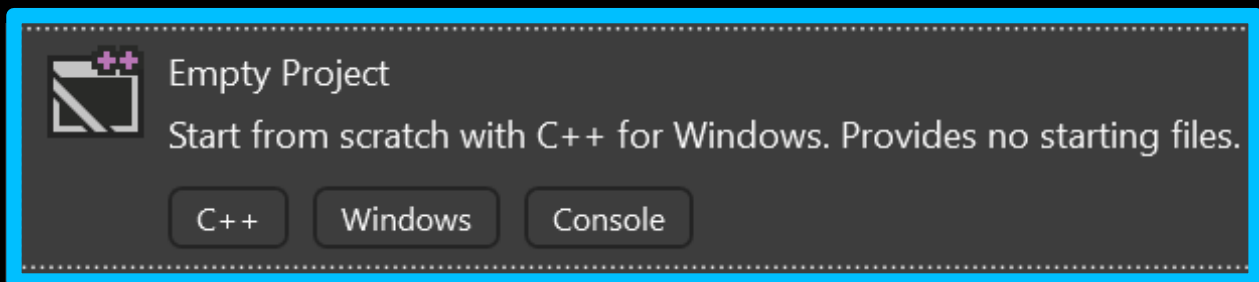


We Choose: Create a new project

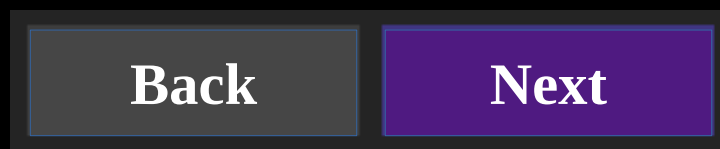
Choices for a New Project



We Left Click: Empty Project



We Left Click: Next Button



Project Name - 001

Configure your new project

Empty Project C++ Windows Console

Project name

001

Location

D:_1Code\C\001

Solution name ⓘ

OutputMessage

☒ Place solution and project in the same directory

Back Create

We name our first project as: 001

We put a Checkmark in: Place solution and project in the same directory

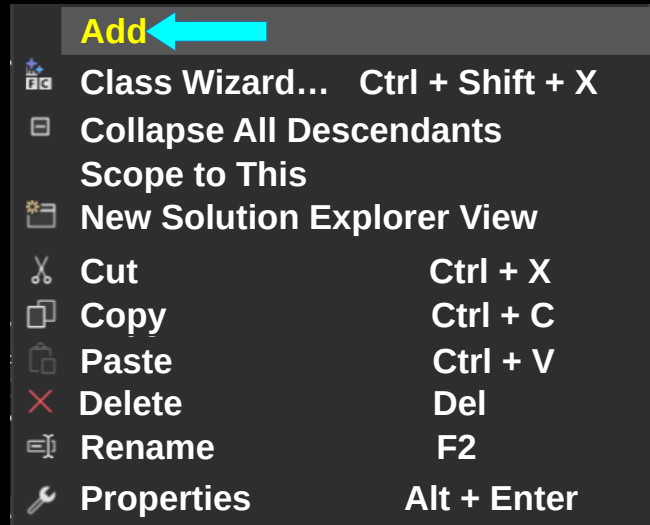
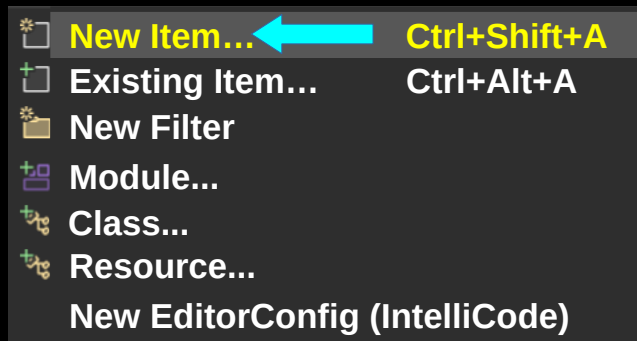
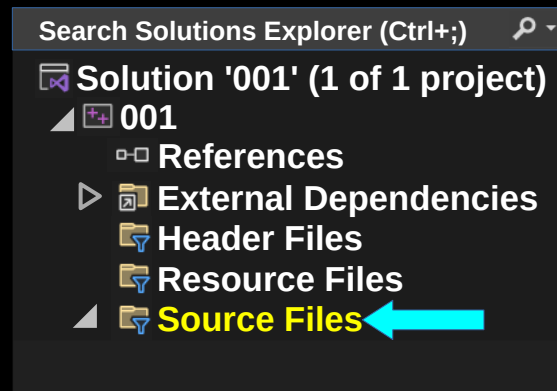
We Left Click: Create Button

Creating our main.c file in Source Files Folder

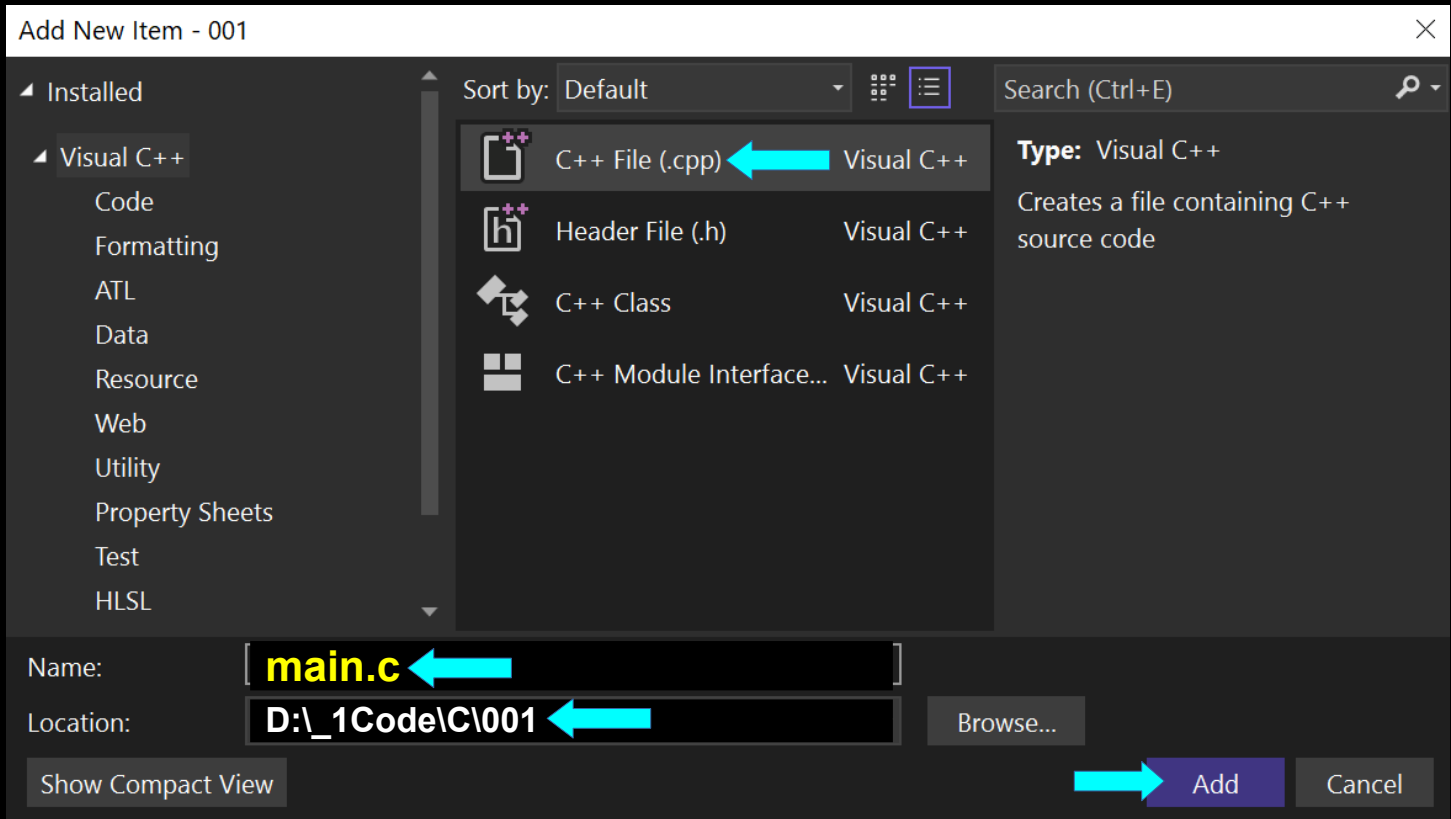
We Right click on: **Source Files** Folder

We Choose: **Add**

We Choose: **New Item...**



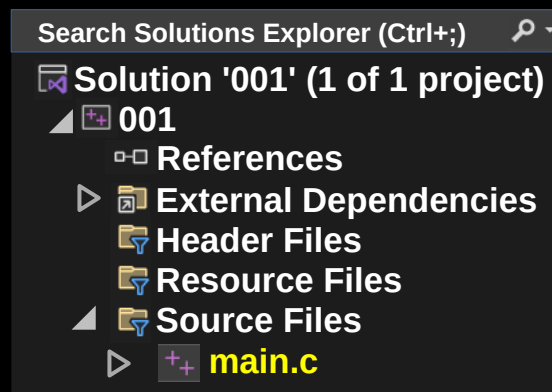
We Choose: C++ File (.cpp)



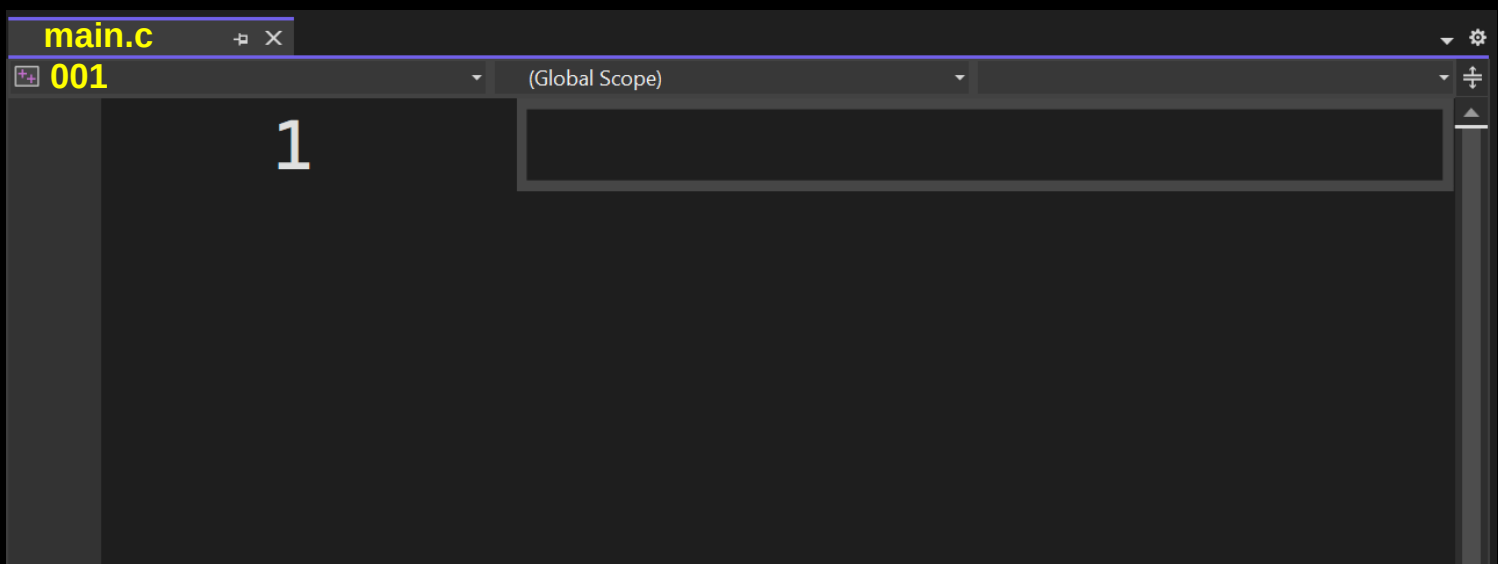
We Name our File: **main.c**

We Left Click: Add button

We see our created file: **main.c**



main.c is now open



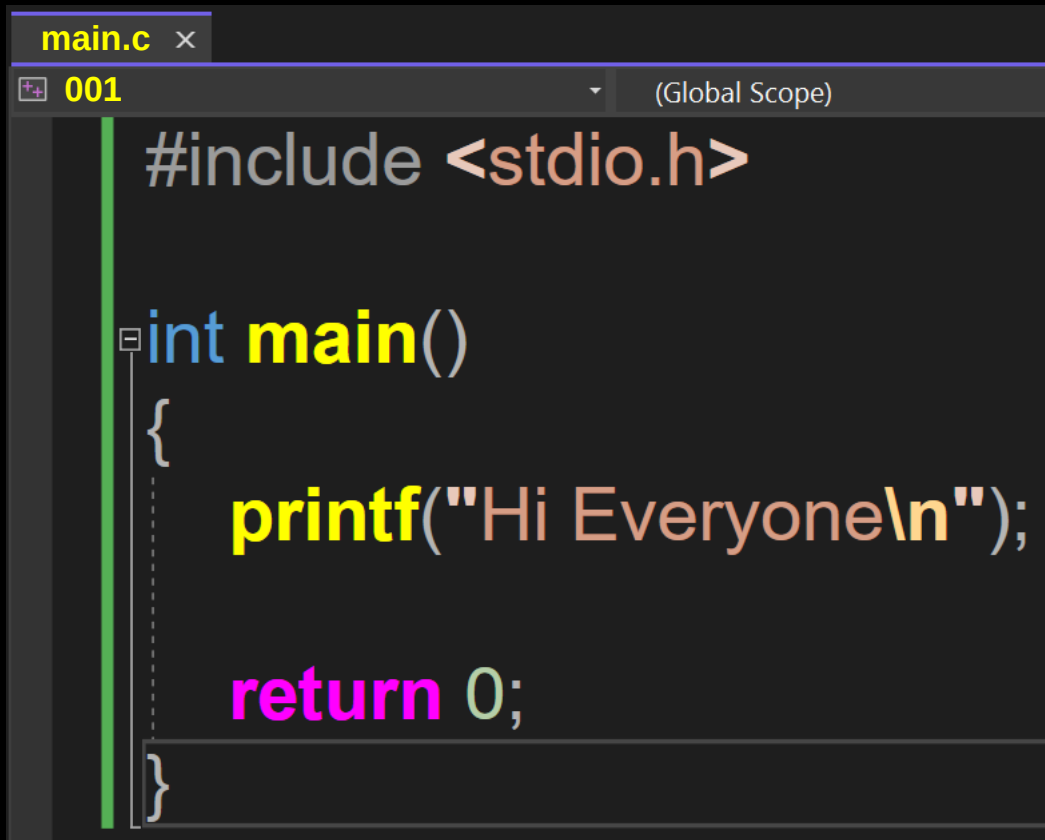
We can now: Type our Code :-)

We make Bigger Font by:
Control + Scroll Wheel Forward

We make Smaller Font by:
Control + Scroll Wheel Backward

main.c Code - Screenshot

Here is a screenshot of: Our C Code



```
main.c x
001
(Global Scope)

#include <stdio.h>

int main()
{
    printf("Hi Everyone\n");

    return 0;
}
```

On the next page
we show the same code,
but with better font.

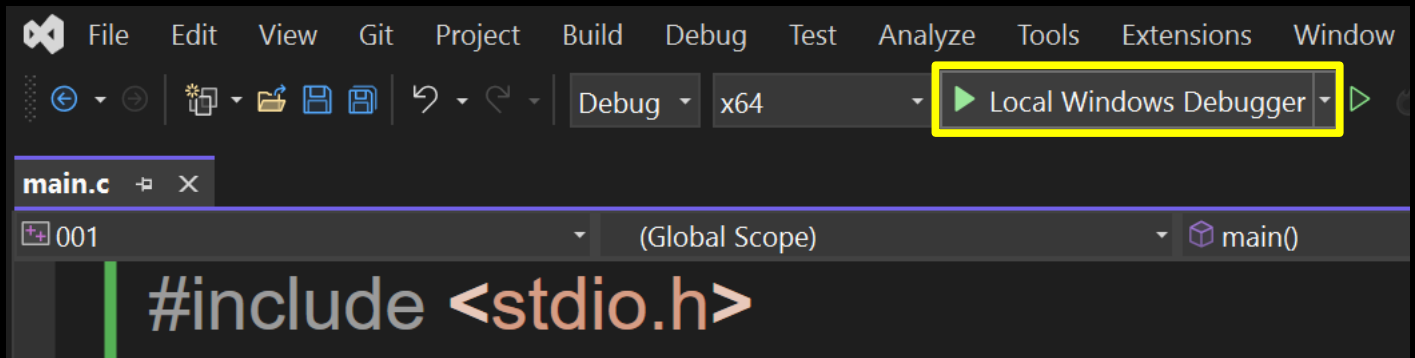
```
// Outputting Text  
// main.c
```

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

```
int main()  
{  
    printf("Hi Everyone\n");  
  
    return 0;  
}
```

// Building and Running our App

We Left Click on: Local Windows Debugger



Our app opens in the Debug Console Window, with the message of: Hi Everyone



We make Bigger Console Font by: Control + Scroll Wheel Forward

We make Smaller Console Font by: Control + Scroll Wheel Backward

// Outputting Text, Exit by Pressing Enter

// main.c

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

```
int main()
```

```
{
```

```
    printf("Hi Everyone\n");
```

```
    printf("Press Enter to Exit\n");
```

```
    // wait for user to press Enter
```

```
    getchar();
```

```
    return 0;
```

```
}
```

```
// Input from user  
// main.c
```

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

```
int main()  
{
```

```
    // max name is 100 chars + null  
    terminator
```

```
    char name[101];
```

```
    printf("Enter First Name: ");
```

```
    // read input from user and prevent  
    buffer overflow
```

```
    scanf_s("%s", name, (unsigned  
int)sizeof(name));
```

```
    printf("Hi %s\n", name);
```



```
printf("\nPress Enter to Exit\n");
```

```
// remove newline char left in input  
buffer
```


```
getchar();
```

```
// wait for user to press Enter
```

```
getchar();
```

```
return 0;
```

```
}
```

 D:_1Code\C\001\x64\Debug\001.exe

Enter First Name: Chris

Hi Chris

Press Enter to Exit

// Custom Function - askName

// main.c

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

```
void askName(char* name)
```

```
{
```

```
    printf("Enter First Name: ");
```

```
    // read input from user and prevent  
    buffer overflow
```

```
    scanf_s("%s", name, (unsigned  
int)sizeof(name));
```

```
}
```

```
int main()
```

```
{
```

```
    // max name is 100 chars + null  
    terminator
```

```
    char userName[101];
```


```
    askName(userName);
```

```
printf("Hi %s\n", userName);

printf("\nPress Enter to Exit\n");
// remove newline char left in input
buffer by scanf_s
getchar();

// wait for user to press Enter
getchar();

return 0;
}
```

 D:_1Code\C\001\x64\Debug\001.exe

```
Enter First Name: Chris
Hi Chris

Press Enter to Exit
```

```
// Custom Function - consoleLog  
// main.c
```

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

```
void consoleLog(const char *message)  
{  
    printf("%s\n", message);  
}
```

```
int main()  
{  
    consoleLog("Hi Everyone");  
  
    printf("Press Enter to Exit\n");  
  
    // wait for user to press Enter  
    getchar();  
  
    return 0;  
}
```

Header File - We define our function in a header file for easy use

Instead of pasting this useful function in every script in our application, we will instead type it once in a header file and put it in the Header Files Folder.

Using a header file is easier, because we place the header files in the Header Files folder and then include that header file with a reference in our main.c and other files.

In our header file, we type the terms

#ifndef

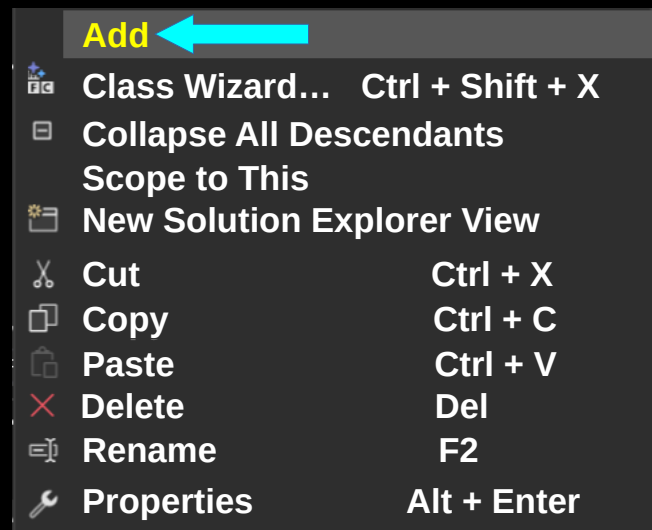
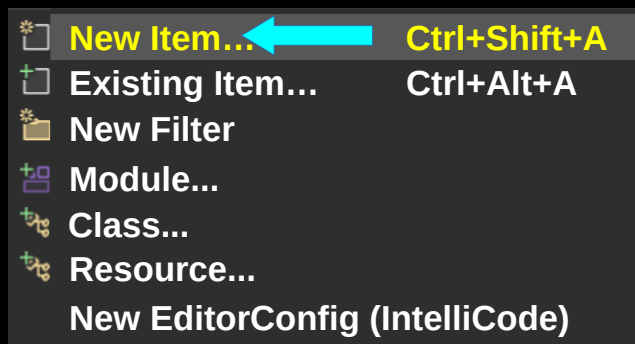
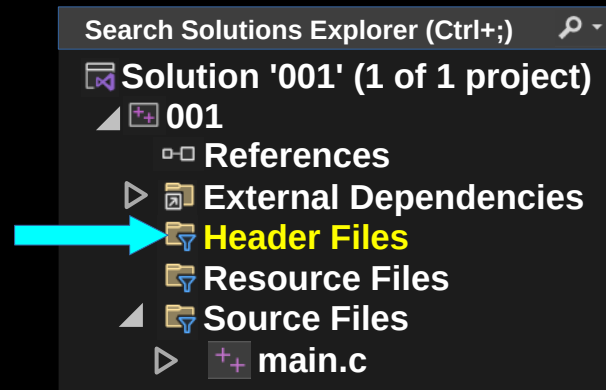
and

#define

to designate that it will be used in other files.

Header File - Add - New Item

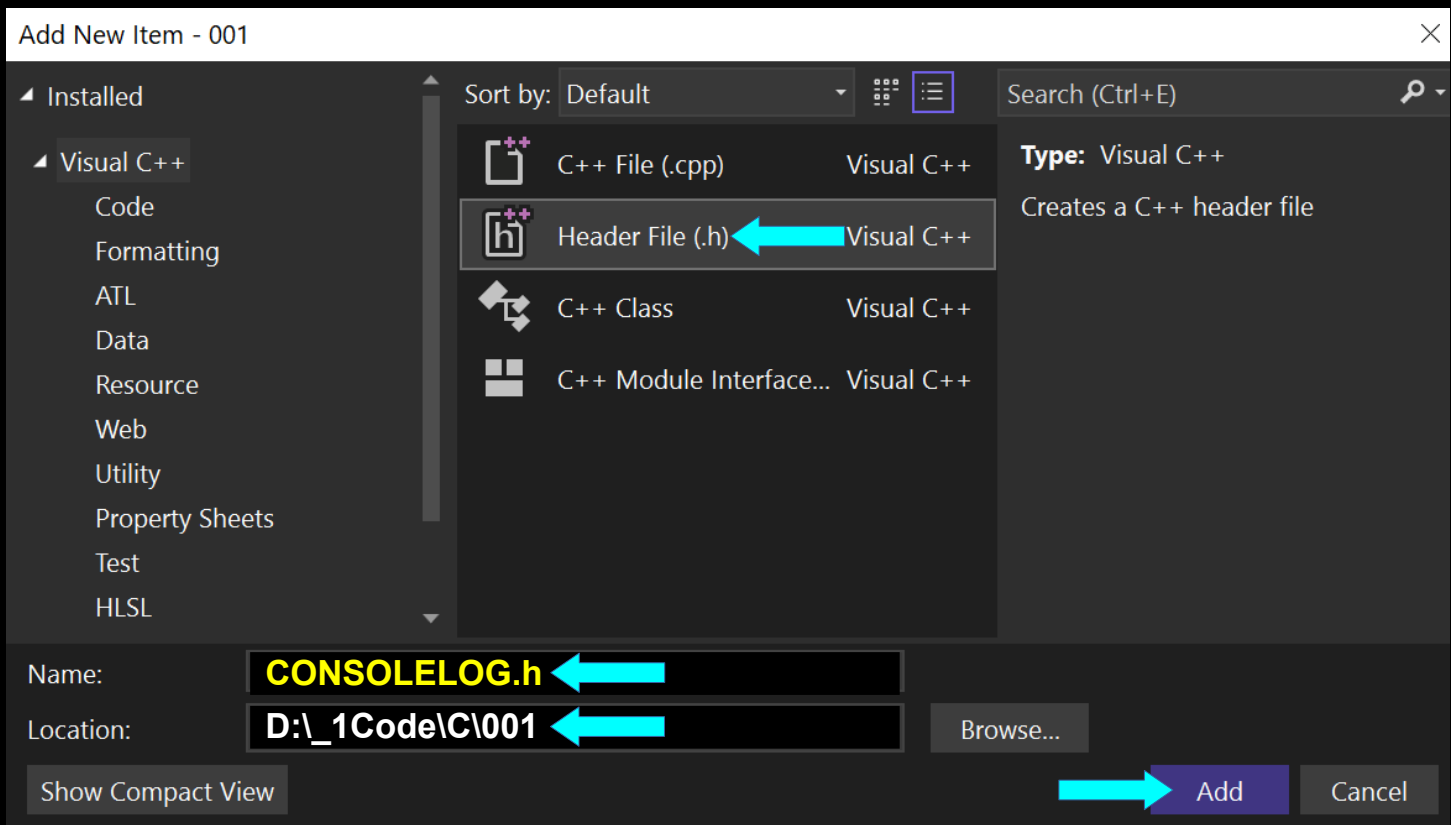
We right click on: **Header Files** folder



We choose: **Add**

We choose: **New Item**

We choose: Header File .h



We name it: CONSOLELOG.h

We Left Click: Add button

```
// CONSOLELOG.h header file  
// CONSOLELOG.h
```

```
#ifndef CONSOLELOG  
#define CONSOLELOG  
#include <stdio.h> // printf
```

```
void consoleLog(const char *message)  
{  
    printf("%s\n", message);  
}
```

```
#endif
```


// Our main.c uses CONSOLELOG.h header file
// main.c

```
#include "CONSOLELOG.h"  
#include <stdio.h> // printf
```

```
int main()  
{  
    consoleLog("Hi Everyone");  
  
    printf("Press Enter to Exit\n");  
  
    // wait for user to press Enter  
    getchar();  
  
    return 0;  
}
```

```
// PROMPT.h header file
```

```
// PROMPT.h
```

```
#ifndef PROMPT
```

```
#define PROMPT
```

```
#include <stdio.h> // scanf_s
```

```
#include <string.h> // strlen
```

```
void prompt(char* userInput)
```

```
{
```

```
    // read input from user and prevent buffer  
overflow
```

```
    scanf_s("%s", userInput, 101);
```

```
    // wait for user to press Enter
```

```
    getchar();
```

```
}
```

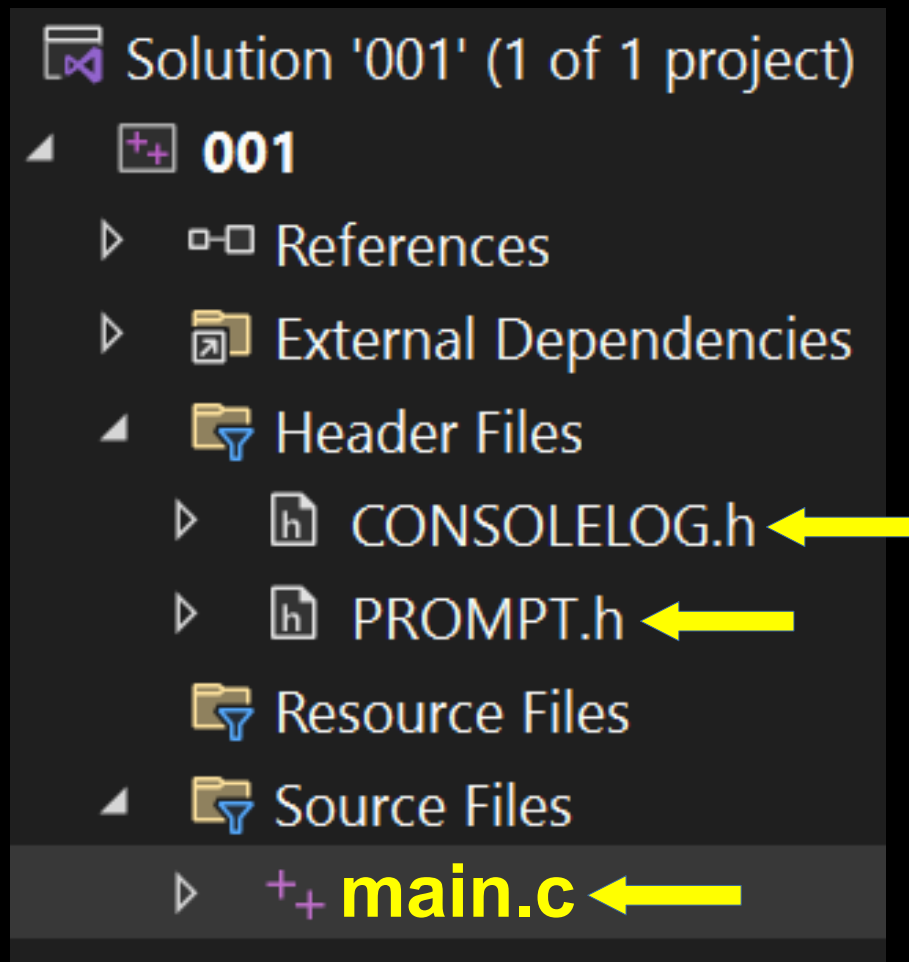
```
#endif
```

```
// main.c uses CONSOLELOG.h and PROMPT.h  
// main.c
```

```
#include "PROMPT.h"  
#include "CONSOLELOG.h"  
#include <stdio.h> // printf, scanf
```

```
int main()  
{  
    // max input is 100 chars + null terminator  
    char input[101];  
  
    consoleLog("Enter First Name");  
    prompt(input);  
  
    printf("Hi %s\n", input);  
  
    consoleLog("Press Enter to Exit\n");  
  
    // wait for user to press Enter  
    getchar();  
  
    return 0;  
}
```

File Structure of the previous Examples



We have 2 Header Files:

CONSOLELOG.h

and

PROMPT.h

We have 1 main.c file:

main.c uses **CONSOLELOG.h** and **PROMPT.h** header files

// Array of Objects

// main.c

```
#include <stdio.h> // printf
```

```
// define a structure to represent a person
```

```
struct Person
```

```
{
```

```
    char name[50];
```

```
    int age;
```

```
};
```

```
int main()
```

```
{
```

```
    // create an array of Person structs
```

```
    struct Person people[] =
```

```
    {
```

```
        { "John", 25 },
```

```
        { "Jane", 30 },
```

```
        { "Fiona", 28 }
```

```
    };
```

```
    // calculate number of elements in array
```

```
    int numPeople = sizeof(people) /
```


```
    sizeof(people[0]);
```

```
// iterate over each person in the array
for (int i = 0; i < numPeople; i++)
{
    printf("Name: %s, Age: %d\n",
people[i].name, people[i].age);
}

printf("\nPress Enter to Exit");

// wait for user to press Enter
getchar();

return 0;
}
```

 D:_1Code\C\001\x64\Debug\001.exe

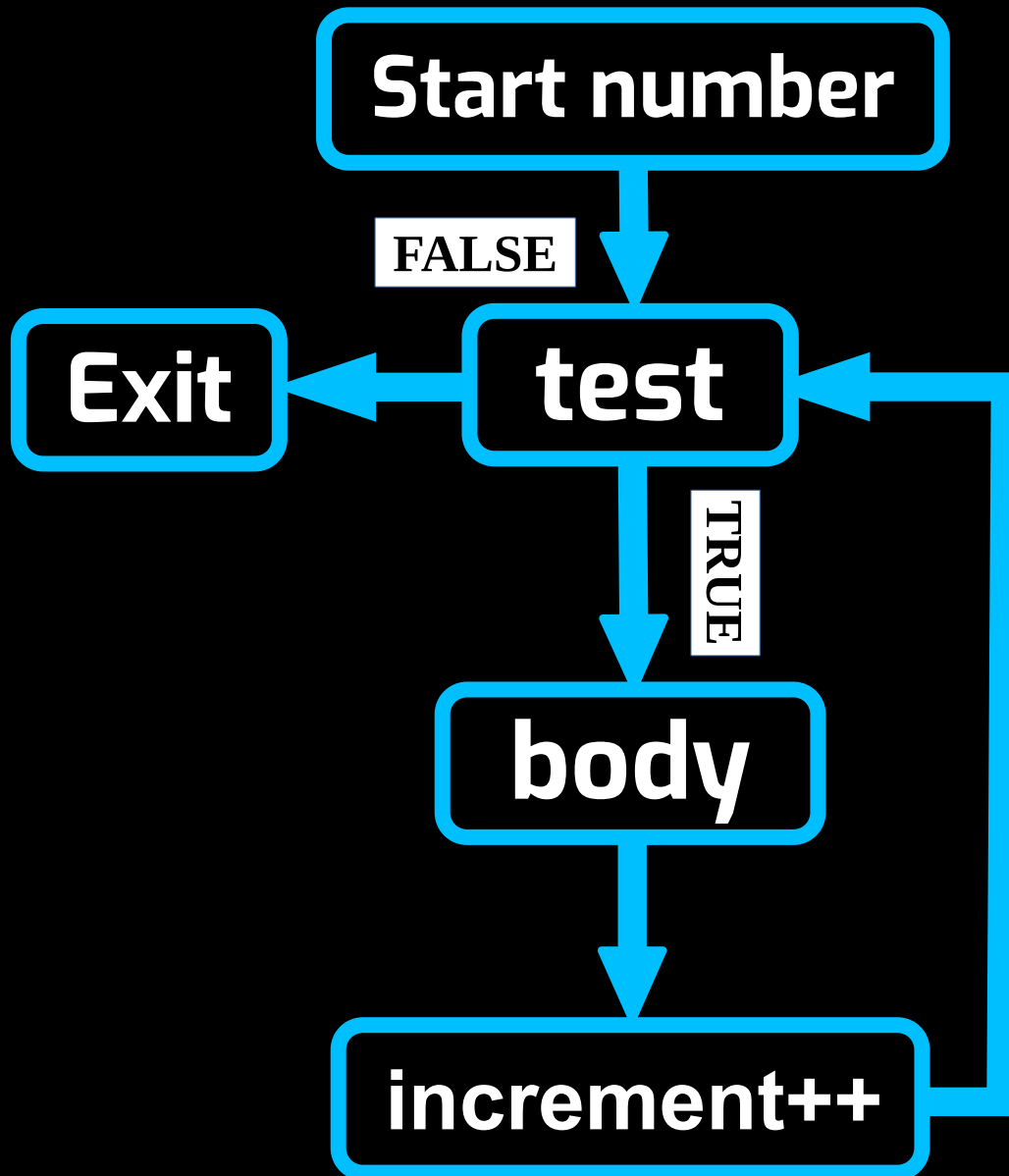
Name: John, Age: 25

Name: Jane, Age: 30

Name: Fiona, Age: 28

Press Enter to Exit_

// for loop diagram

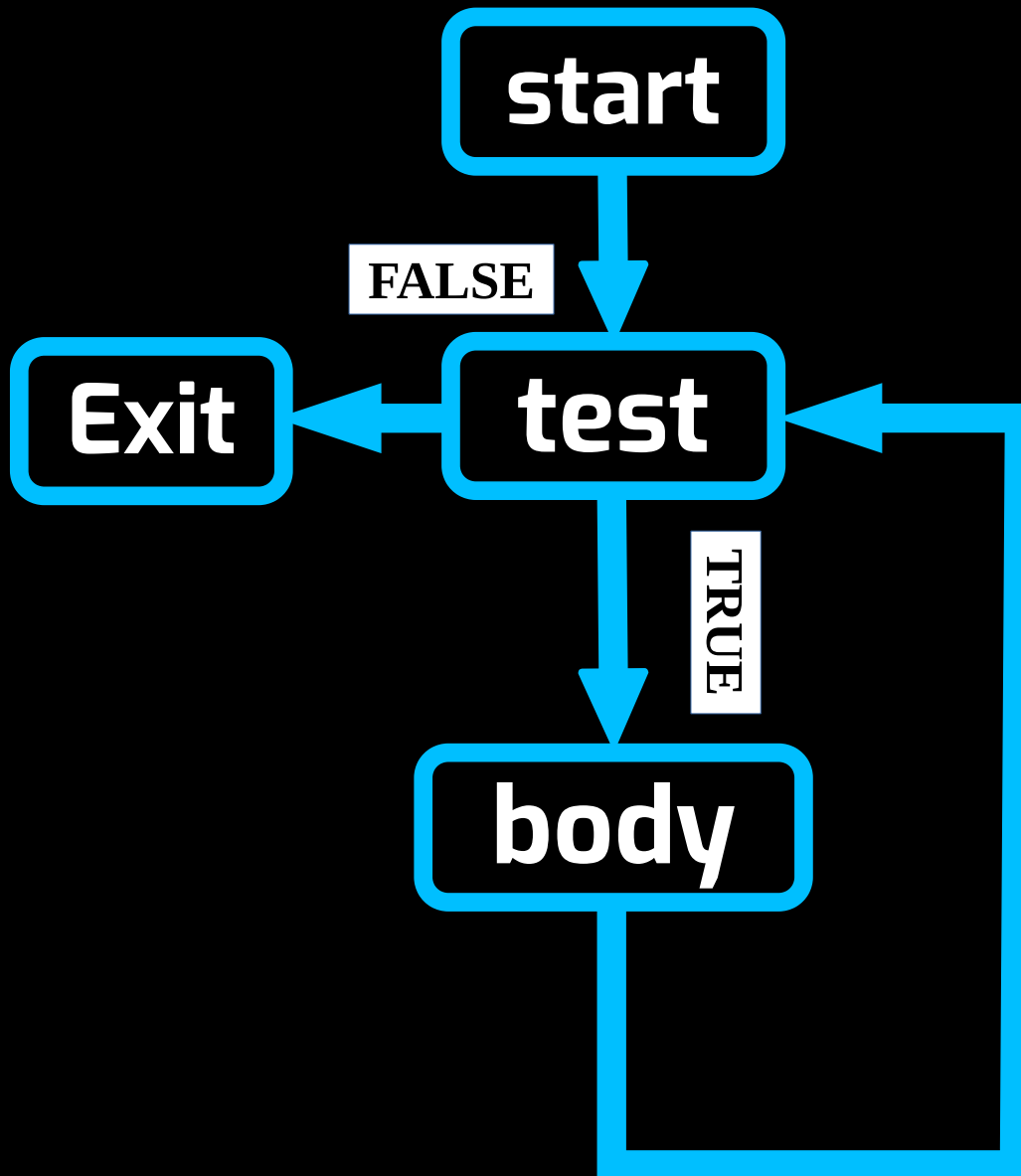


```
// for loop  
// main.c
```

```
#include <stdio.h> // printf
```

```
int main()  
{  
    for (int i = 1; i <= 100; i++)  
    {  
        printf("%d ", i);  
        printf("\n");  
    }  
  
    printf("\nPress Enter to Exit");  
  
    // wait for user to press Enter  
    getchar();  
  
    return 0;  
}
```


// while loop diagram



```
// whileLoop
```

```
// main.c
```

```
#include <stdio.h> // printf
```

```
int main()
```

```
{
```

```
    int i = 0; // initialize the counter
```

```
    while (i < 100)
```

```
    {
```

```
        printf("%d ", i);
```

```
        i++; // increment counter
```

```
    }
```

```
    printf("\n"); // print new line after loop
```

```
    printf("\nPress Enter to exit");
```

```
    getchar(); // wait for user input
```

```
    return 0;
```

```
}
```

// Using pointers to modify variables in functions - (pass by reference)

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

```
void increment(int* num)
{
    // *num is a pointer to i
    (*num)++; // this changes the variable i
}
```

```
int main()
{
    int i = 0;

    increment(&i); // passing the address of i

    printf("%d\n", i); // prints 1 because i was
    modified through the pointer

    printf("\nPress Enter to Exit");
    getchar();
    return 0;
}
```

// result: 1

```
// if else - strcmp, without null terminating the  
string  
// main.c
```

```
#include <stdio.h> // printf, scanf_s  
#include <string.h> // strcmp
```

```
int main()  
{  
    char name[101]; // buffer to store the name  
  
    printf("Enter your name: ");  
  
    // read input from user and prevent buffer  
    overflow  
    scanf_s("%s", name, (unsigned  
int)sizeof(name));  
  
    // remove newline char left in input buffer  
    getchar();  
  
    if (strcmp(name, "Chris") == 0)  
    {  
        printf("Hi Chris.\nIt is good that you are  
visiting Earth.\n");  
    }  
}
```

```
}  
else  
{  
    printf("Howdy %s. Tell Chris to Sign in  
later.\n", name);  
}  
  
printf("\nPress Enter to Exit");  
  
// wait for user to press Enter  
getchar();  
  
return 0;  
}
```

CS D:_1Code\C\001\x64\Debug\001.exe

```
Enter your name: John  
Howdy John. Tell Chris to Sign in later.  
  
Press Enter to Exit_
```

CS D:_1Code\C\001\x64\Debug\001.exe

```
Enter your name: Chris  
Hi Chris.  
It is good that you are visiting Earth.  
  
Press Enter to Exit_
```

```
// if else - strcmp, null terminating the string  
// main.c
```

```
#include <stdio.h>  
#include <string.h>
```

```
int main()  
{  
    char name[101]; // buffer to store the name  
  
    printf("Enter your name: ");  
  
    // read input from user and prevent buffer  
    overflow  
    scanf_s("%100s", name, (unsigned  
int)sizeof(name) - 1);  
  
    // explicitly null terminate the string  
    name[sizeof(name) - 1] = '\0';  
  
    // remove newline char left in input buffer  
    getchar();  
  
    if (strcmp(name, "Chris") == 0)  
    {
```

```
    printf("Hi Chris.\nIt is good that you are  
visiting Earth.\n");  
}  
else  
{  
    printf("Howdy %s. Tell Chris to Sign in  
later.\n", name);  
}  
  
printf("\nPress Enter to Exit");  
  
// wait for user to press Enter  
getchar();  
  
return 0;  
}
```

// This version makes sure that we first null terminate the string before comparison.

```
// Open Browser to a URL  
// main.c
```

```
#include <windows.h>
```

```
int main()  
{  
    ShellExecuteA(NULL, "open",  
"https://www.google.com", NULL, NULL,  
SW_SHOWNORMAL);  
  
    return 0;  
}
```


// Open Browser to a URL using char url
// main.c

```
#include <windows.h>
```

```
int main()
```

```
{
```

```
    // URL to open
```

```
    const char* url = "https://www.google.com";
```

```
    // open web browser to specified URL
```

```
    ShellExecuteA(NULL, "open", url, NULL,  
NULL, SW_SHOWNORMAL);
```

```
    return 0;
```

```
}
```

// Custom Function - Open Browser to a URL

// main.c

```
#include <windows.h>
```

```
void openURL(const char *url)
{
    ShellExecuteA(NULL, "open", url, NULL,
    NULL, SW_SHOWNORMAL);
}
```

```
int main()
{
    const char *url = "https://www.google.com";

    openURL(url);

    return 0;
}
```

// Create Text File with Data

// main.c

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

```
int main()
```

```
{
```

```
    // declare FILE pointer
```

```
    FILE* outputFile;
```

```
    // open file for writing
```

```
    if (fopen_s(&outputFile, "ourTextFile.txt",  
"w") == 0)
```

```
    {
```

```
        // write data to file using fprintf_s
```

```
        if (fprintf_s(outputFile, "Hi Everyone\n") >  
0)
```

```
        {
```

```
            // close file
```

```
            fclose(outputFile);
```

```
            printf("Data written successfully.\n");
```

```
        }
```

```
    else
```

```
    {
```

```
    printf("Error writing data to file.\n");  
    // close file if an error occurs  
    fclose(outputFile);  
}  
}  
else  
{  
    printf("Failed to open file.\n");  
}  
  
return 0;  
}
```

// Custom Function - Create Text File with Data

// main.c

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

```
void writeToFile(const char* fileName, const  
char* content)  
{  
    // declare FILE pointer  
    FILE* outputFile;  
  
    // open file for writing  
    if (fopen_s(&outputFile, fileName, "w") == 0)  
    {  
        // check if file opened successfully  
        if (outputFile != NULL)  
        {  
            // write data to file  
            fprintf(outputFile, "%s\n", content);  
  
            // close file  
            fclose(outputFile);  
  
            printf("Data written to %s successfully.  
\n", fileName);  
        }  
    }  
}
```

```
    }  
    else  
    {  
        printf("Failed to open file %s.\n",  
fileName);  
    }  
}  
else  
{  
    printf("Failed to open file %s.\n",  
fileName);  
}  
}
```

```
int main()  
{  
    const char* fileName = "ourTextFile.txt";  
    const char* content = "Hi Everyone";  
  
    writeToFile(fileName, content);  
  
    return 0;  
}
```

// Read a Text File

// main.c

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

```
void displayFileContents(const char* fileName)
```

```
{
```

```
    // declare FILE pointer
```

```
    FILE* inputFile;
```

```
    // open file for reading
```

```
    if (fopen_s(&inputFile, fileName, "r") == 0)
```

```
    {
```

```
        // check if file opened successfully
```

```
        if (inputFile != NULL)
```

```
        {
```

```
            // max line length is 255 chars + 1 null  
terminator
```

```
            char line[256];
```

```
            printf("Contents of %s:\n", fileName);
```

```
            // read, display file contents line by line  
            while (fgets(line, sizeof(line), inputFile) !=  
= NULL)
```

```
    {
        printf("%s", line);
    }

    // close the file
    fclose(inputFile);
}
else
{
    printf("Error opening file: %s\n",
fileName);
}
}
else
{
    printf("Error opening file: %s\n",
fileName);
}
}

int main()
{
    const char* fileName = "ourTextFile.txt";

    displayFileContents(fileName);
}
```



```
printf("\nPress Enter to Exit");  
  
// wait for user to press Enter  
getchar();  
  
return 0;  
}
```

// Count Number of Lines in a Text File

// main.c

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

```
int main()
```

```
{
```

```
    // declare FILE pointer
```

```
    FILE* inputFile;
```

```
    // open text file for reading
```

```
    if (fopen_s(&inputFile, "ourTextFile.txt", "r")  
== 0)
```

```
    {
```

```
        // check if the file opened successfully
```

```
        if (inputFile == NULL)
```

```
        {
```

```
            fprintf(stderr, "File won't open.\n");
```

```
            return 1; // return an error code
```

```
        }
```

```
    // variable to store the count of lines
```

```
    int lineCount = 0;
```

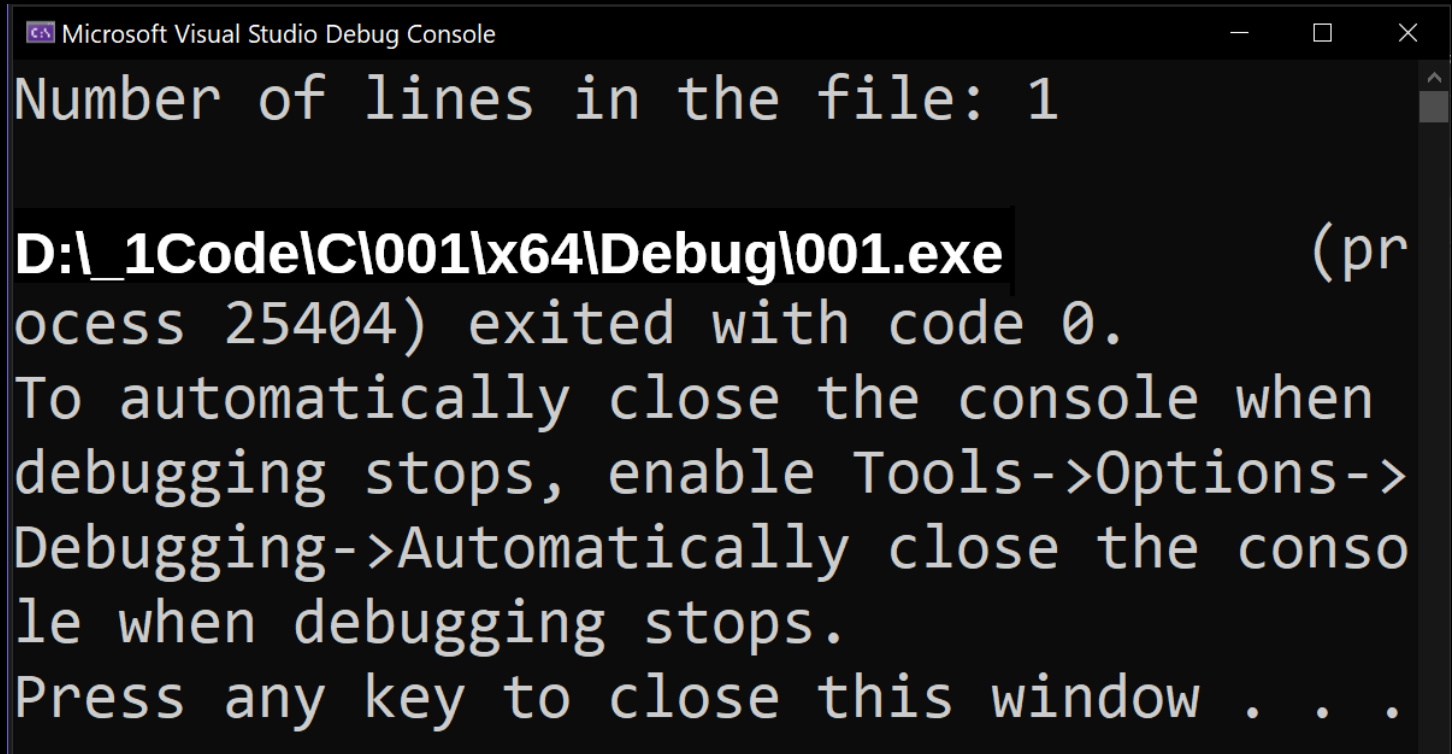
```
// temp buffer stores each line read from
file
// max line length is 255 chars + 1 null
terminator
char line[256];

// read file line by line and count lines
while (fgets(line, sizeof(line), inputFile) !=
NULL)
{
    lineCount++;
}

// close the file
fclose(inputFile);

// display the total number of lines
printf("Number of lines in the file: %d\n",
lineCount);
}
else
{
    fprintf(stderr, "File won't open.\n");
    return 1; // return an error code
}
```

```
    return 0;  
}
```

A screenshot of the Microsoft Visual Studio Debug Console window. The window has a title bar with the text "Microsoft Visual Studio Debug Console" and standard window controls (minimize, maximize, close). The console output shows the text "Number of lines in the file: 1" followed by a line separator. Below that, the text "D:_1Code\C\001\x64\Debug\001.exe (process 25404) exited with code 0." is displayed. This is followed by a multi-line instruction: "To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops." and finally "Press any key to close this window . . .".

```
Microsoft Visual Studio Debug Console  
Number of lines in the file: 1  
  
D:\_1Code\C\001\x64\Debug\001.exe (process 25404) exited with code 0.  
To automatically close the console when debugging stops, enable Tools->Options->  
Debugging->Automatically close the console when debugging stops.  
Press any key to close this window . . .
```

// Calculate Hard Drive Memory Statistics

```
#include <stdio.h>
#include <windows.h>

int main()
{
    ULARGE_INTEGER freeBytesAvailable;
    ULARGE_INTEGER totalBytes;
    ULARGE_INTEGER totalFreeBytes;

    if (GetDiskFreeSpaceEx(NULL,
        &freeBytesAvailable, &totalBytes,
        &totalFreeBytes))
    {
        printf("Total space: %llu bytes\n",
            totalBytes.QuadPart);

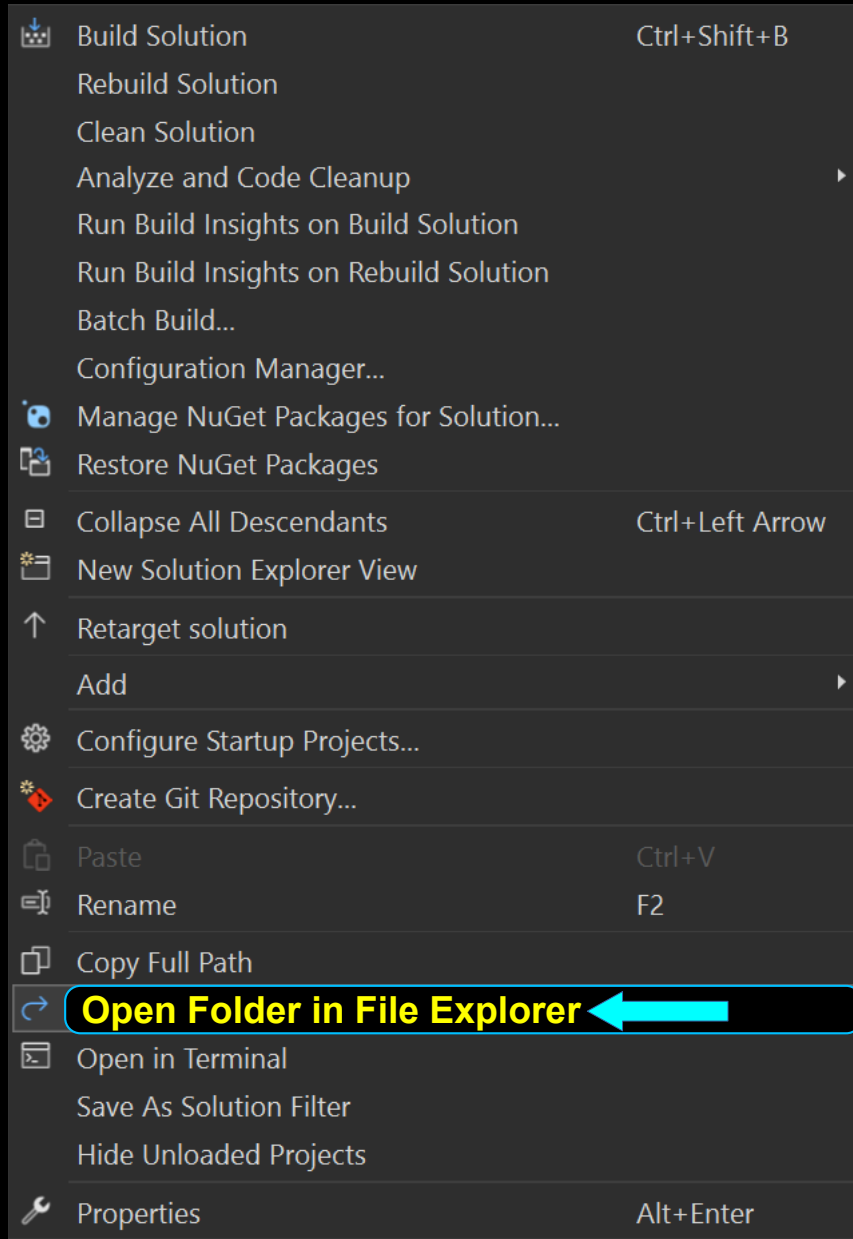
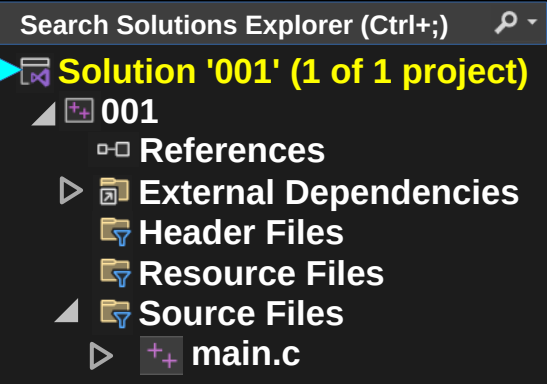
        printf("Free space: %llu bytes\n",
            totalFreeBytes.QuadPart);

        printf("Available space: %llu bytes\n",
            freeBytesAvailable.QuadPart);
    }
    else
    {
```

```
    perror("Error getting disk space  
information");  
    return 1; // return an error code  
}  
  
printf("\nPress Enter to Exit");  
  
// wait for user to press Enter  
getchar();  
  
return 0;  
}
```

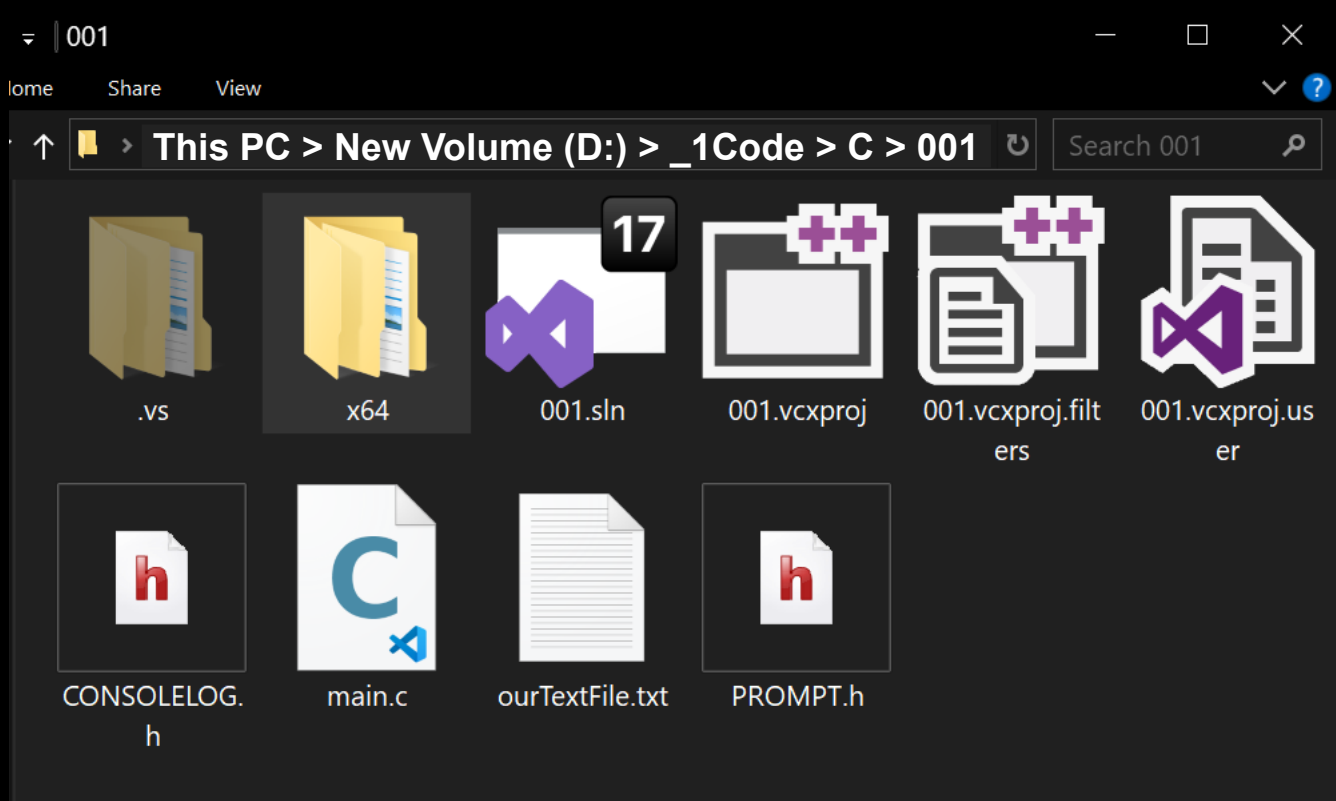
// How to Find Our Application .exe File

We put mouse arrow on:
Solution '001' (1 of 1 project)

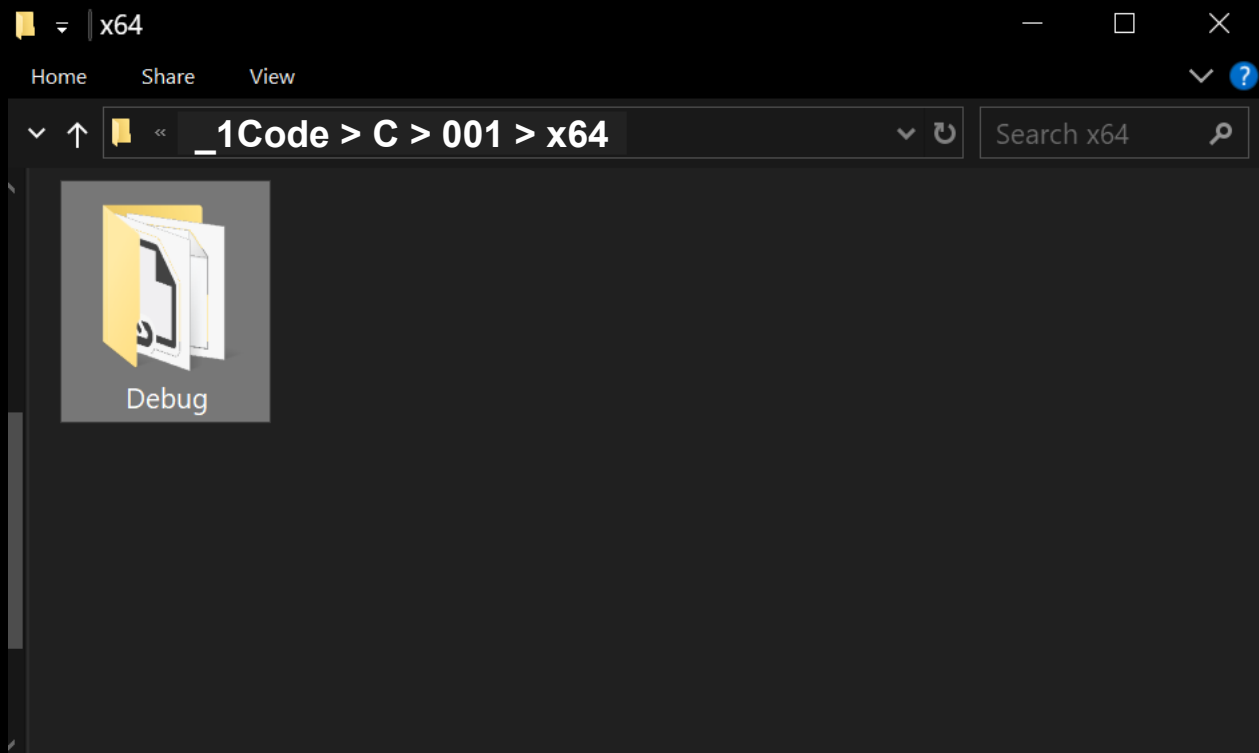


We Choose: **Open Folder in File Explorer**

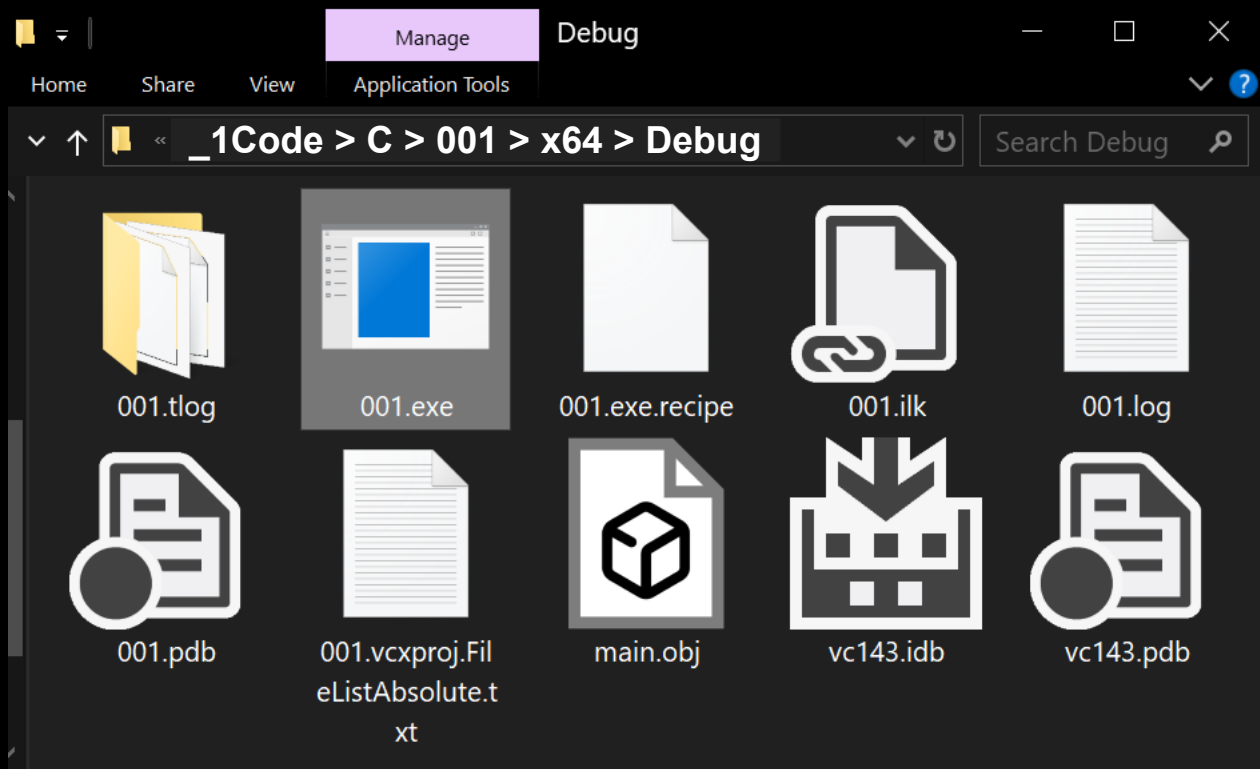
We see that our Project Folder has opened:



We Open: x64 Folder to find the Debug Folder



We Open: Debug Folder to find 001.exe



We Double Left Click: 001.exe

Our application should activate.

Happy Programming :-)

```
// get current working directory
```

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

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

```
#include <windows.h> // GetCurrentDirectory
```

```
int main()
```

```
{
```

```
    char buffer[MAX_PATH];
```

```
    // get the current working directory
```

```
    if (!GetCurrentDirectory(MAX_PATH, buffer))
```

```
    {
```

```
        perror("Error getting current working  
directory");
```

```
        return EXIT_FAILURE;
```

```
    }
```

```
    printf("Current working directory: %s\n",  
buffer);
```

```
    return EXIT_SUCCESS;
```

```
}
```

// random0to1

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

```
int main()
```

```
{
```

```
    // seed the random number generator with
    the current time
```

```
    srand((unsigned int)time(NULL));
```

```
    // generate a random float between 0.0 and
    1.0
```

```
    float randomFloat = (float)srand() /
    (float)RAND_MAX;
```

```
    // print the random float
```

```
    printf("Random float: %f\n", randomFloat);
```

```
    // prompt user to press Enter to exit
```

```
    printf("Press Enter to Exit\n");
```

```
    getchar();
```

```
    return 0;
```

```
}
```

```
//----//
```

```
/*
```

Random float: 0.767113

Press Enter to Exit

```
*/
```

```
// random_integer_0_to_10.c
```

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

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

```
#include <time.h>
```

```
int main()
```

```
{
```

```
    // seed the random number generator with  
the current time
```

```
    srand((unsigned int)time(NULL));
```

```
    // generate a random integer between 1 and  
10
```

```
    int randomInteger = (rand() % 10) + 1;
```

```
    // print the random integer
```

```
    printf("Random integer: %d\n",  
randomInteger);
```

```
    // prompt user to press Enter to exit
```

```
    printf("Press Enter to Exit\n");
```

```
    getchar();
```

```
    return 0;
```

```
}
```

```
//----//
```

```
/*
```

Random integer: 7

Press Enter to Exit

```
*/
```

```
// getCurrentDate.c
```

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

```
#include <time.h>
```

```
void getCurrentDate(char* buffer, size_t  
bufferSize)
```

```
{
```

```
    // get current time
```

```
    time_t now = time(NULL);
```

```
    // declare a tm structure to hold local time  
    struct tm localTime;
```

```
    // use localtime_s for safe local time  
conversion
```

```
    localtime_s(&localTime, &now);
```

```
    // format the date as "YYYY-MM-DD" and  
store it in buffer
```

```
    strftime(buffer, bufferSize, "%Y-%m-%d",  
&localTime);
```

```
}
```

```
int main()
```

```
{
```

```
char date[11]; // buffer to store date string
```

```
// get current date
```

```
getCurrentDate(date, sizeof(date));
```

```
// print current date
```

```
printf("Current date: %s\n", date);
```

```
return 0;
```

```
}
```

```
//----//
```

```
/*
```

```
Current date: 2024-10-31
```

```
*/
```



```
// getCurrentDate_if_date.c
```

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

```
#include <time.h>
```

```
#include <string.h>
```

```
void getCurrentDate(char* buffer, size_t  
bufferSize)
```

```
{
```

```
    // get current time
```

```
    time_t now = time(NULL);
```

```
    // declare a tm structure to hold local time  
    struct tm localTime;
```

```
    // use localtime_s for safe local time  
conversion
```

```
    localtime_s(&localTime, &now);
```

```
    // format the date as "YYYY-MM-DD" and  
store it in buffer
```

```
    strftime(buffer, bufferSize, "%Y-%m-%d",  
&localTime);
```

```
}
```

```
int main()
```

```
{  
    char date[11]; // buffer to store current date  
    string  
    const char* specifiedDate = "2024-10-31"; //  
    specify date to compare  
  
    // get current date  
    getCurrentDate(date, sizeof(date));  
  
    // print current date  
    printf("Current date: %s\n", date);  
  
    // check if the current date matches the  
    specified date  
    if (strcmp(date, specifiedDate) == 0)  
    {  
        printf("Today is the specified date: %s\n",  
specifiedDate);  
    }  
    else  
    {  
        printf("Today is not the specified date.\n");  
    }  
  
    return 0;  
}
```

//----//

/*

Current date: 2024-10-31

Today is the specified date: 2024-10-31

*/

```
// getDayOfMonth.c
```

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

```
#include <time.h>
```

```
int getDayOfMonth()
```

```
{
```

```
    // get current time
```

```
    time_t now = time(NULL);
```

```
    // declare a tm structure to hold local time
```

```
    struct tm localTime;
```

```
    // use localtime_s for safe local time  
conversion
```

```
    localtime_s(&localTime, &now);
```

```
    // return the day of the month
```

```
    return localTime.tm_mday;
```

```
}
```

```
int main()
```

```
{
```

```
    // get and print the current day of the month
```

```
    printf("Current day of the month: %d\n",
```

```
    getDayOfMonth());
```

```
    return 0;  
}
```

```
//----//
```

```
/*
```

```
Current day of the month: 31
```

```
*/
```

```
// getYear.c
```

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

```
#include <time.h>
```

```
int getYear()
```

```
{
```

```
    // get current time
```

```
    time_t now = time(NULL);
```

```
    // declare a tm structure to hold local time
```

```
    struct tm localTime;
```

```
    // use localtime_s for safe local time  
conversion
```

```
    localtime_s(&localTime, &now);
```

```
    // return the year
```

```
    return localTime.tm_year + 1900; // tm_year  
is years since 1900  
}
```

```
int main()
```

```
{
```

```
    // get and print the current year
```

```
    printf("Current year: %d\n", getYear());
```

```
    return 0;  
}
```

```
//----//
```

```
/*  
Current year: 2024  
*/
```

```
// sleep_print.c
```

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

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

```
#include <time.h>
```

```
int main()
```

```
{
```

```
    printf("Count to 4\n");
```

```
    // sleep for 4 seconds
```

```
    #ifdef _WIN32
```

```
        Sleep(4000); // Sleep function in  
milliseconds on Windows
```

```
    #else
```

```
        sleep(4); // sleep function in seconds on  
other systems
```

```
    #endif
```

```
    printf("4 seconds passed\n");
```

```
    // prompt user to press Enter to exit
```

```
    printf("Press Enter to Exit");
```

```
    getchar(); // waits for Enter
```

```
    return 0;
```


}

//----//

/*

Count to 4

4 seconds passed

Press Enter to Exit

*/

```
// reverse_string.c
```

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

```
#include <string.h>
```

```
void reverseString(char* str)
```

```
{
```

```
    int length = strlen(str);
```

```
    char temp;
```

```
    // reverse the string in place
```

```
    for (int i = 0; i < length / 2; i++)
```

```
    {
```

```
        temp = str[i];
```

```
        str[i] = str[length - i - 1];
```

```
        str[length - i - 1] = temp;
```

```
    }
```

```
}
```

```
int main()
```

```
{
```

```
    char word[] = "kingdom";
```

```
    // reverse the word
```

```
    reverseString(word);
```

```
// print the reversed word
printf("Reversed word: %s\n", word);

// prompt user to press Enter to exit
printf("Press Enter to Exit");
getchar(); // waits for Enter

return 0;
}

//----//

/*
Reversed word: modgnik
Press Enter to Exit
*/
```

```
// split_words.c
```

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

```
#include <string.h>
```

```
#define MAX_WORDS 10
```

```
#define MAX_WORD_LENGTH 50
```

```
int main()
```

```
{
```

```
    char words[] = "Tabitha Lee";
```

```
    char splitWords[MAX_WORDS]  
[MAX_WORD_LENGTH]; // array to hold split  
words
```

```
    int count = 0;
```

```
    // use strtok_s to split the string
```

```
    char* context; // for strtok_s
```

```
    char* token = strtok_s(words, " ",  
&context); // split by space
```

```
    while (token != NULL && count <  
MAX_WORDS)
```

```
{
    // copy the token to the array safely
    strncpy_s(splitWords[count],
MAX_WORD_LENGTH, token,
MAX_WORD_LENGTH - 1);

    splitWords[count][MAX_WORD_LENGTH -
1] = '\0'; // ensure null-termination

    count++;

    // get the next token
    token = strtok_s(NULL, " ", &context);
}

// print the split words
for (int i = 0; i < count; i++)
{
    printf("%s\n", splitWords[i]);
}

// prompt user to press Enter to exit
printf("Press Enter to Exit");
getchar(); // waits for Enter

return 0;
```

}

//----//

/*

Tabitha

Lee

Press Enter to Exit

*/

```
// get_os_name.c

#include <stdio.h>

int main()
{
    // determine the operating system
#ifdef _WIN32
    printf("Operating System: Windows\n");
#elif __linux__
    printf("Operating System: Linux\n");
#elif __APPLE__ || __MACH__
    printf("Operating System: macOS\n");
#else
    printf("Operating System: Unknown\n");
#endif

    // prompt user to press Enter to exit
    printf("Press Enter to Exit");
    getchar(); // waits for Enter

    return 0;
}

//----//
```

/*

Operating System: Windows

Press Enter to Exit

*/


```
// mph_to_kph.c
```

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

```
int main()
```

```
{
```

```
    double mph, kph;
```

```
    // prompt user for input
```

```
    printf("Enter speed in miles per hour (mph):  
");
```

```
    if (scanf_s("%lf", &mph) != 1)
```

```
    {
```

```
        printf("Invalid input.\n");
```

```
        return 1; // exit if input is invalid
```

```
    }
```

```
    // convert mph to kph
```

```
    kph = mph * 1.60934;
```

```
    // display result
```

```
    printf("Speed in kph: %.2f\n", kph);
```

```
    return 0;
```

```
}
```

//----//

/*

Enter speed in miles per hour (mph): 40

Speed in kilometers per hour (kph): 64.37

*/

// pointersTutorial

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

```
int main()
```

```
{
```

```
    int num = 10; // declare an integer variable
```

```
    int* pointerToNum = &num; // declare a  
    pointer variable and assign it the address of  
    'num'
```

```
    printf("Value of num: %d\n", num); // prints  
    the value of num (10)
```

```
    printf("Address of num: %p\n", &num); //  
    prints the memory address of num
```

```
    printf("Pointer pointerToNum points to  
    address: %p\n", pointerToNum); // prints the  
    address stored in pointerToNum (address of  
    num)
```

```
    printf("Value stored at the address  
    pointerToNum is pointing to: %d\n",  
    *pointerToNum); // dereferencing  
    pointerToNum, prints the value of num (10)
```

```
printf("\nPress Enter to Exit");  
getchar();
```

```
return 0;  
}
```

```
/*  
Value of num: 10  
Address of num: 0000009DB02FF764  
Pointer pointerToNum points to address:  
0000009DB02FF764  
Value stored at the address pointerToNum is  
pointing to: 10
```

```
Press Enter to Exit  
*/
```

// arrayOfObjects

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

```
#include <string.h>
```

```
// define the Person class (struct)
```

```
struct Person {  
    char name[50];  
    char dob[20];  
    int score;  
};
```

```
int main()  
{
```

```
    // create an array of 3 Person objects
```

```
    struct Person people[3] = {  
        {"John Doe", "1995-03-25", 85},  
        {"Jane Smith", "1993-07-19", 92},  
        {"Alice Brown", "1996-12-12", 78}  
    };
```

```
    // use a for loop to display the information of  
    each person
```

```
    for (int i = 0; i < 3; i++) {  
        printf("Name: %s\n", people[i].name);  
        printf("Date of Birth: %s\n", people[i].dob);  
    }
```

```
printf("Score: %d\n", people[i].score);  
printf("\n");  
}
```

```
printf("\nPress Enter to Exit");  
getchar();
```

```
return 0;  
}
```

```
/*
```

```
Name: John Doe  
Date of Birth: 1995-03-25  
Score: 85
```

```
Name: Jane Smith  
Date of Birth: 1993-07-19  
Score: 92
```

```
Name: Alice Brown  
Date of Birth: 1996-12-12  
Score: 78
```

```
Press Enter to Exit  
*/
```

// arrayOfObjectsFilterByScore

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

```
#include <string.h>
```

```
// define the Person class (struct)
```

```
struct Person {  
    char name[50];  
    char dob[20];  
    int score;  
};
```

```
int main()  
{
```

```
    // create an array of 3 Person objects
```

```
    struct Person people[3] = {  
        {"John Doe", "1995-03-25", 85},  
        {"Jane Smith", "1993-07-19", 92},  
        {"Alice Brown", "1996-12-12", 78}  
    };
```

```
    // define the threshold score for filtering
```

```
    int scoreThreshold = 80;
```

```
    // use a for loop to display the information of  
    people who meet the score condition
```

```
for (int i = 0; i < 3; i++) {  
    if (people[i].score > scoreThreshold)  
    {  
        printf("Name: %s\n", people[i].name);  
        printf("Date of Birth: %s\n",  
people[i].dob);  
        printf("Score: %d\n", people[i].score);  
        printf("\n");  
    }  
}  
  
printf("\nPress Enter to Exit");  
getchar();  
  
return 0;  
}
```

```
/*
```

Name: John Doe

Date of Birth: 1995-03-25

Score: 85

Name: Jane Smith

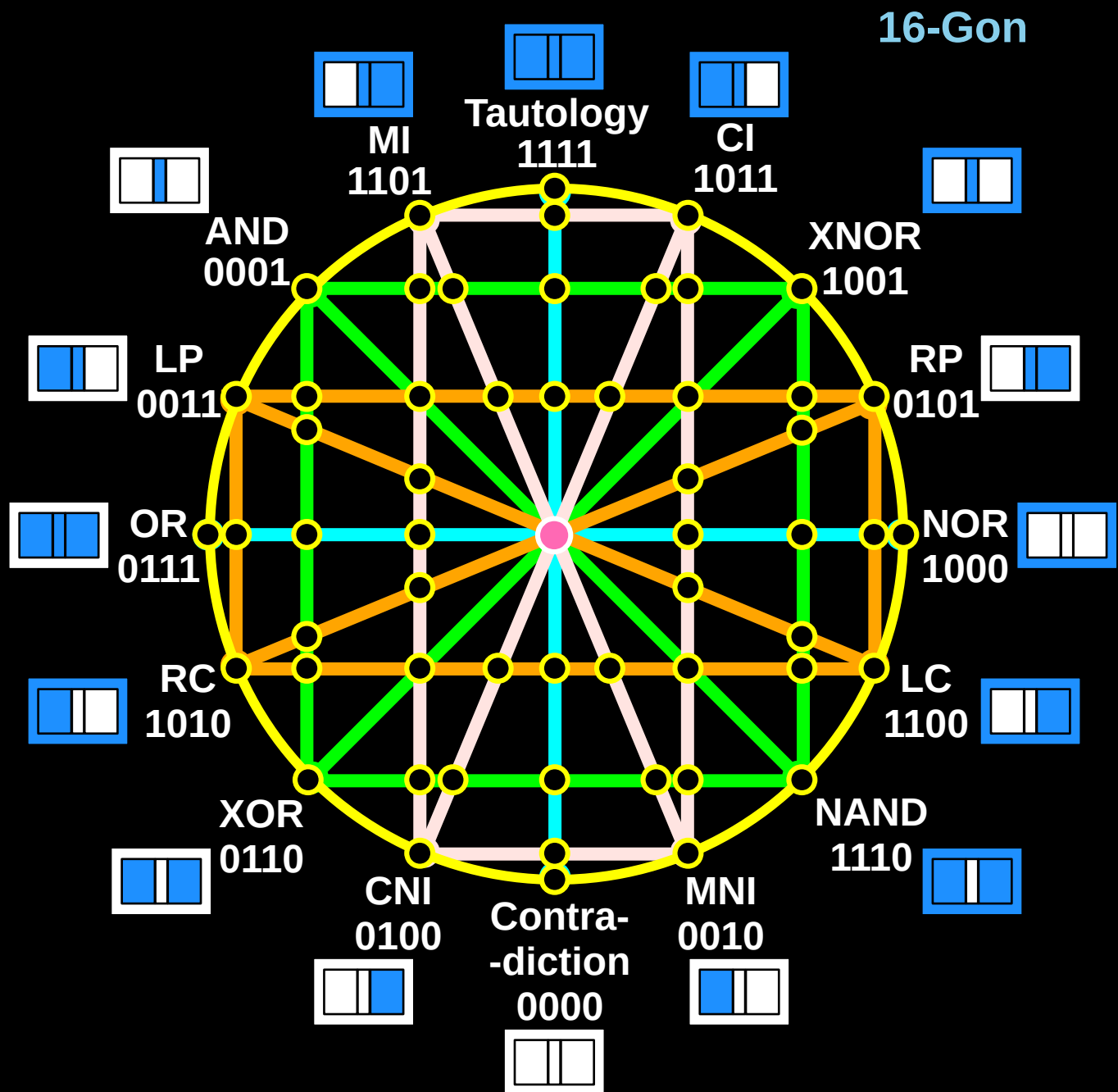
Date of Birth: 1993-07-19

Score: 92

Press Enter to Exit

***/**

True Artificial Intelligence System



For More Tutorials:

[**GitHub.com/ChristopherTopalian**](https://github.com/ChristopherTopalian)

[**GitHub.com/ChristopherAndrewTopalian**](https://github.com/ChristopherAndrewTopalian)

[**Youtube.com/ScriptingCollege**](https://youtube.com/ScriptingCollege)

[**Youtube.com/@CollegOfScripting**](https://youtube.com/@CollegOfScripting)

[**X.com/CollegeOfScript**](https://x.com/CollegeOfScript)

[**Rumble.com/user/CollegeofScripting**](https://rumble.com/user/CollegeofScripting)

[**Sites.google.com/view/CollegeOfScripting**](https://sites.google.com/view/CollegeOfScripting)

[**CollegeOfScripting.weebly.com**](https://CollegeOfScripting.weebly.com)

[**CollegeOfScripting.wordpress.com**](https://CollegeOfScripting.wordpress.com)

Dedicated to God the Father

This book is created by the
College of Scripting Music & Science.

Always remember, that each time you write a script with a pencil and paper, it becomes imprinted so deeply in memory that the material and methods are learned extremely well. When you Type the scripts, the same is true.

The more you type and write out the scripts by keyboard or pencil and paper, the more you will learn programming!

Write & Type EVERY example that you find. Keep all of your scripts organized. Every script that you create increases your programming abilities.

SEEING CODE, is one thing,
but WRITING CODE is another.
Write it, Type it, Speak it, See it, Dream it.

www.CollegeOfScripting.weebly.com