V-Ex Tech

V- EX TECH

Web Development

Java / Node.js / PHP / .Net / Python

Certification Course

Assured Placement Program With

International Certificate

About V-Ex Tech….

V-Ex Tech is an elevated education platform providing rigorous industry-relevant programs Designed and delivered on collaboration with industry professionals. It has been constantly Into process of creating an immersive learning experience binding latest technologies, pedagogy and services with enormous job placement opportunities too.

V-Ex Tech

C language

C is a general-purpose programming language, developed in 1972

C Introduction

What is C?

C is a general-purpose programming language created by Dennis Ritchie at the Bell Laboratories in 1972.

It is a very popular language

Why Learn C?

- It is one of the most popular programming language in the world
- If you know C, you will have no problem learning other popular programming languages such as Java, Python, C++, C#, etc, as the syntax is similar
- C is very fast, compared to other programming languages, like Java and Python

V-Ex Tech

Difference between C and C++

- C++ was developed as an extension of C, and both languages have almost the same syntax
-
- The main difference between C and C++ is that C++ support classes and objects, while C does not

Get Started With C

To start using C, you need two things:

- A text editor,vs code like Notepad, vs code, to write C code
-
- A compiler, like GCC, to translate the C code into a language that the computer will understand

There are many text editors and compilers to choose from. In this tutorial, we will use an ***IDE***

C Install IDE

An IDE (Integrated Development Environment) is used to edit AND compile the code.

Popular IDE's include Code::Blocks, Eclipse, and Visual Studio. These are all free, and they can be used to both edit and debug C code.

We will use **Code::Blocks** in our tutorial

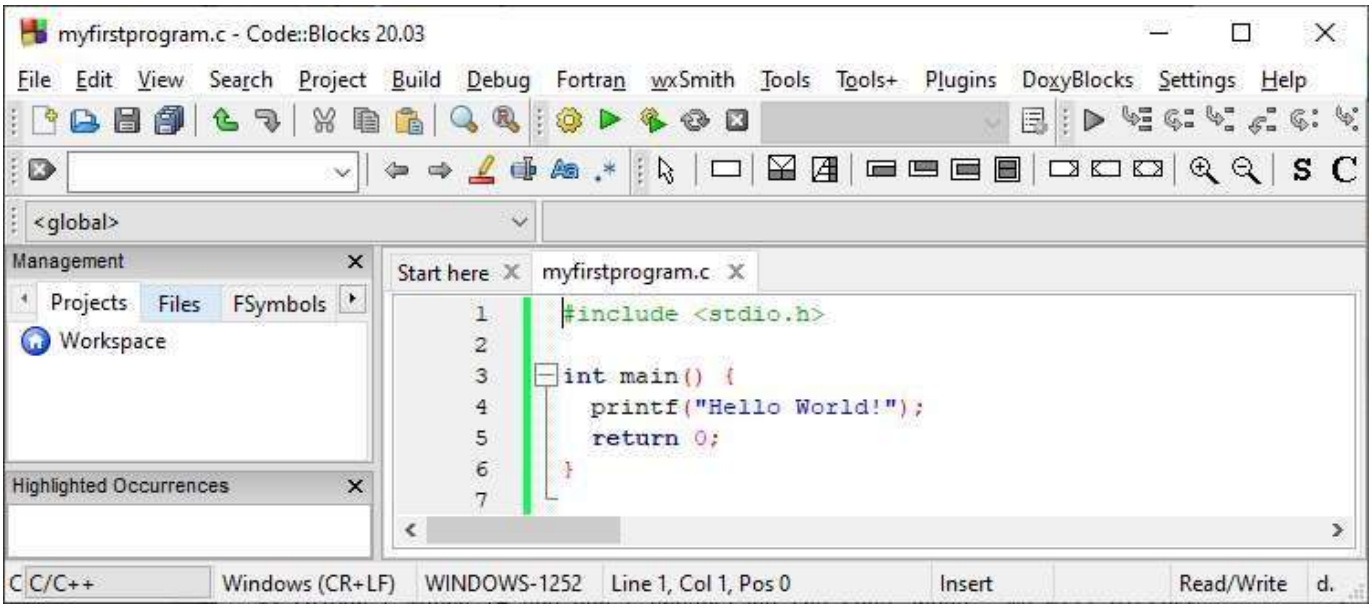
You can find the latest version of Codeblocks at http://www.codeblocks.org/. Download the `mingw-setup.exe` file, which will install the text editor with a compiler.

Start

Let's create our first C file.

Open Codeblocks and go to **File > New > Empty File**.

Write the following C code and save the file as `myfirstprogram.c`



Then, go to **Build > Build and Run** to run (execute) the program. The result will look something to this:

```
Hello World!
Process returned 0 (0x0) execution time : 0.011 s
Press any key to continue.
```

C Syntax

```c
#include <stdio.h>

int main() {
  printf(" World");
  return 0;
}
```

Header file

Main function

{

Variablr declare

Body..

Return value

}

V-Ex Tech

Explaination

Line 1: `#include <stdio.h>` is a **header file library** that lets us work with input and output functions, such as `printf()` (used in line 4). Header files add functionality to C programs.

Studio.h -defines standered input and output function

include-declaration syntex

Line 2: A blank line. C ignores white space. But we use it to make the code more readable.

Line 3: Another thing that always appear in a C program, is `main()`. This is called a **function**. Any code inside its curly brackets `{}` will be executed.

Line 4: `printf()` is a **function** used to output/print text to the screen.

Line 5: `return 0` ends the `main()` function.

Line 6: Do not forget to add the closing curly bracket `}` to actually end the main function.

Notes:

Every C statement ends with a semicolon ;

C Output (Print Text)

```c
#include <stdio.h>

int main() {
  printf(" World");
  return 0;
}
```

```c
#include <stdio.h>

int main() {
  printf("Hello World!");
  printf("I am learning C.");
  return 0;
}
```

New Lines

```
#include <stdio.h>

int main() {
  printf("Hello World!\n");
  printf("I am learning C.");
  return 0;
}
```

The newline character (\n) is called an escape sequence

Space consider

```
#include <stdio.h>

int main() {
  printf("Hello World!\n");
  printf("I am learning C.");
  return 0;
}
```

Two \n characters after each other will create a blank line:

```
#include <stdio.h>

int main() {
  printf("Hello World!\n\n");
  printf("I am learning C.");
  return 0;
}
```

Horizontal space

```
#include <stdio.h>

int main() {
  printf("Hello World!\t");
  printf("I am learning C.");
  return 0;
}
```

V-Ex Tech

Inserts a backslash character (\)

```
#include <stdio.h>

int main() {
  printf("Hello World!\\");
  printf("I am learning C.");
  return 0;
}
```

Inserts a double quote character

```
#include <stdio.h>

int main() {
  printf("They call him \"Johnny\".");
  return 0;
}
```

Task-1)

Print Your name,dob,age,pincode,state,city,country,bloodgroup,roll no.

V-Ex Tech

C Comments

Comments can be **singled-lined** or **multi-lined**.

Single-line Comments

C Multi-line Comments

```
// This is a comment
printf("Hello World!");
```

C Variables

In C, there are different types **of variables**

- `int` - stores integer
- `float` - stores floating point numbers, with decimals, such as `19.99` or `-19.99`
- `char` - stores single characters, such as `'a'` or `'B'`.

Declaring (Creating) Variables

Syntax

```
type variableName = value;
```

```
int myNum = 15;
```

You can also declare a variable without assigning the value, and assign the value later:

```
int myNum;

// Assign a value to the variable
myNum = 15;
```

```
#include <stdio.h>

int main() {
  int myNum = 15;
  printf(myNum); // Nothing happens
  return 0;
}
```

V-Ex Tech

```c
#include <stdio.h>

int main() {
  int myNum = 15;
  printf("%d", myNum);
  return 0;
}
```

```c
#include <stdio.h>

int main() {
  int myNum = 15;
  printf("My favorite number is: %d", myNum);
  return 0;
}
```

```c
#include <stdio.h>

int main() {
  int myNum = 15;
  char myLetter = 'D';
  printf("My number is %d and my letter is %c", myNum,
myLetter);
  return 0;
}
```

Change Variable Values

```c
#include <stdio.h>

int main() {
  int myNum = 15; // myNum is 15
  myNum = 10; // Now myNum is 10

  printf("%d", myNum);
  return 0;
}
```

```c
#include <stdio.h>

int main() {
  int myNum = 15;

  int myOtherNum = 23;

  // Assign the value of myOtherNum (23) to myNum
  myNum = myOtherNum;

  // myNum is now 23, instead of 15
  printf("%d", myNum);

  return 0;
}
```

```c
#include <stdio.h>

int main() {
  // Create a myNum variable and assign the value 15 to it
  int myNum = 15;

  // Declare a myOtherNum variable without assigning it a value
  int myOtherNum;

  // Assign value of myNum to myOtherNum
  myOtherNum = myNum;

  // myOtherNum now has 15 as a value
  printf("%d", myOtherNum);

  return 0;
}
```

Add Variables Together

```c
#include <stdio.h>

int main() {
  int x = 5;
  int y = 6;
  int sum = x + y;
  printf("%d", sum);
  return 0;
}
```

Declare Multiple Variables

```
#include <stdio.h>

int main() {
  int x = 5, y = 6, z = 50;
  printf("%d", x + y + z);
  return 0;
}
```

The **general rules** for naming variables are:

- Names can contain letters, digits and underscores
- Names must begin with a letter or an underscore (_)
- Names are case sensitive (`myVar` and `myvar` are different variables)
- Names cannot contain whitespaces or special characters like !, #, %, etc.
- Reserved words (such as `int`) cannot be used as names

V-Ex Tech

```c
#include <stdio.h>

int main() {
  // Student data
  int studentID = 15;
  int studentAge = 23;
  float studentFee = 75.25;
  char studentGrade = 'B';

  // Print variables
  printf("Student id: %d\n", studentID);
  printf("Student age: %d\n", studentAge);
  printf("Student fee: %f\n", studentFee);
  printf("Student grade: %c", studentGrade);

  return 0;
}
```

V-Ex Tech

C Data Types

```
#include <stdio.h>

int main() {
  // Create variables
  int myNum = 5;               // Integer (whole number)
  float myFloatNum = 5.99;     // Floating point number
  char myLetter = 'D';         // Character

  // Print variables
  printf("%d\n", myNum);
  printf("%f\n", myFloatNum);
  printf("%c\n", myLetter);
  return 0;
}
```

Basic Data Type

Format Specifier	Data Type	T
%d or %i	int	Tr
%f	float	
%lf	double	Tr
%c	char	
%s	Used for **strings** **(text)**, which you will learn more about in a later chapter	Tr

```
#include <stdio.h>

int main() {
  float myFloatNum = 3.5;

  printf("%f\n", myFloatNum); // Default will show 6 digits
after the decimal point
  printf("%.1f\n", myFloatNum); // Only show 1 digit
  printf("%.2f\n", myFloatNum); // Only show 2 digits
  printf("%.4f", myFloatNum);   // Only show 4 digits
  return 0;
}
```

Type Conversion

```c
#include <stdio.h>

int main() {
  int x = 5;
  int y = 2;
  int sum = 5 / 2;

  printf("%d", sum);
  return 0;
}
```

Implicit Conversion

```c
#include <stdio.h>

int main() {
  // Automatic conversion: int to float
  float myFloat = 9;

  printf("%f", myFloat);
  return 0;
}
```

```c
#include <stdio.h>

int main() {
  // Automatic conversion: float to int
  int myInt = 9.99;

  printf("%d", myInt);
  return 0;
}
```

Explicit Conversion

```c
#include <stdio.h>

int main() {
  // Manual conversion: int to float
  float sum = (float) 5 / 2;

  printf("%f", sum);
  return 0;
}
```

```c
#include <stdio.h>

int main() {
  int num1 = 5;
  int num2 = 2;
  float sum = (float) num1 / num2;

  printf("%.1f", sum);
  return 0;
}
```

Constants

```c
#include <stdio.h>

int main() {
  const int myNum = 15;
  myNum = 10;

  printf("%d", myNum);
  return 0;
}
```

```c
#include <stdio.h>

int main() {
  const int minutesPerHour = 60;
  const float PI = 3.14;

  printf("%d\n", minutesPerHour);
  printf("%f\n", PI);
  return 0;
}
```

Operators

Operators are used to perform operations on variables and values.

```c
#include <stdio.h>

int main() {
  int myNum = 100 + 50;
  printf("%d", myNum);
  return 0;
}
```

```c
#include <stdio.h>

int main() {
  int sum1 = 100 + 50;        //150 (100 + 50)
  int sum2 = sum1 + 250;      // 400 (150 +250)
  int sum3 = sum2 + sum2;     // 800 (400 +400)
  printf("%d\n", sum1);
  printf("%d\n", sum2);
  printf("%d\n", sum3);
  return 0;
}
```

C divides the operators into the following groups:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators
- Bitwise operators

V-Ex Tech

Arithmetic Operators

Operator	Name	Description	Example
+	Addition	Adds together two values	x + y
-	Subtraction	Subtracts one value from another	x - y
*	Multiplication	Multiplies two values	x * y
/	Division	Divides one value by another	x / y
%	Modulus	Returns the division remainder	x % y
++	Increment	Increases the value of a variable by 1	++x
--	Decrement	Decreases the value of a variable by 1	--x

Assignment Operators

```
#include <stdio.h>

int main() {
  int x = 10;
  x += 5;
  printf("%d", x);
  return 0;
}
```

Operator	Example	Same As	Try it
=	x = 5	x = 5	Try it »
+=	x += 3	x = x + 3	Try it »
-=	x -= 3	x = x - 3	Try it »
*=	x *= 3	x = x * 3	Try it »
/=	x /= 3	x = x / 3	Try it »
%=	x %= 3	x = x % 3	Try it »
&=	x &= 3	x = x & 3	Try it »
|=	x |= 3	x = x | 3	Try it »
^=	x ^= 3	x = x ^ 3	Try it »
>>=	x >>= 3	x = x >> 3	Try it »
<<=	x <<= 3	x = x << 3	Try it »

Comparison Operators

```
#include <stdio.h>

int main() {
  int x = 5;
  int y = 3;
  printf("%d", x > y); // returns 1 (true) because 5 is greater
than 3
  return 0;
}
```

Operator	Name	Example
==	Equal to	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Logical Operators

You can also test for true or false values with logical operators.

Logical operators are used to determine the logic between variables or values:

Operator	Name	Description	Example
&&	Logical and	Returns true if both statements are true	x < 5 && x < 10
||	Logical or	Returns true if one of the statements is true	x < 5 || x < 4
!	Logical not	Reverse the result, returns false if the result is true	!(x < 5 && x < 10)

Booleans

- YES / NO
- ON / OFF
- TRUE / FALSE

- 1 (or any other number that is not 0) represents true
- 0 represents false

you must use the %d format specifier to print a boolean value:

```
#include <stdio.h>
#include <stdbool.h>  // Import the boolean header file

int main() {
  bool isProgrammingFun = true;
  bool isFishTasty = false;
  printf("%d\n", isProgrammingFun);  // Returns 1 (true)
  printf("%d", isFishTasty);         // Returns 0 (false)

  return 0;
}
```

Comparing Values and Variables

```
#include <stdio.h>

int main() {
  printf("%d", 10 > 9);

  return 0;
}
```

```
#include <stdio.h>

int main() {
  printf("%d\n", 10 == 10);
  printf("%d\n", 10 == 15);
  printf("%d", 5 == 55);
  return 0;
}
```

```c
#include <stdio.h>
#include <stdbool.h>  // Import the boolean header file

int main() {
  bool isHamburgerTasty = true;
  bool isPizzaTasty = true;
  printf("%d", isHamburgerTasty == isPizzaTasty);

  return 0;
}
```

Real Life Example

```c
#include <stdio.h>

int main() {
  int myAge = 25;
  int votingAge = 18;

  printf("%d", myAge >= votingAge);
  return 0;
}
```

```c
#include <stdio.h>

int main() {
  int myAge = 25;
  int votingAge = 18;

  if (myAge >= votingAge) {
    printf("Old enough to vote!");
  } else {
    printf("Not old enough to vote.");
  }

  return 0;
}
```

If ... Else

- Less than: a < b
- Less than or equal to: a <= b
- Greater than: a > b
- Greater than or equal to: a >= b
- Equal to a == b
- Not Equal to: a != b

if Statement

Syntax

```
if (condition) {
  // block of code to be executed if the condition is true
}
```

```
#include <stdio.h>

int main() {
  if (20 > 18) {
    printf("20 is greater than 18");
  }
  return 0;
}
```

```c
#include <stdio.h>

int main() {
  int x = 20;
  int y = 18;
  if (x > y) {
    printf("x is greater than y");
  }
  return 0;
}
```

The else Statement

Syntax

```c
if (condition) {
  // block of code to be executed if the condition is true
} else {
  // block of code to be executed if the condition is false
}
```

```c
#include <stdio.h>

int main() {
  int time = 20;
  if (time < 18) {
    printf("Good day.");
  } else {
    printf("Good evening.");
  }
  return 0;
}
```

The else if Statement

Syntax

```c
if (condition1) {
  // block of code to be executed if condition1 is true
} else if (condition2) {
  // block of code to be executed if the condition1 is false and condition2 is true
} else {
  // block of code to be executed if the condition1 is false and condition2 is false
}
```

```c
#include <stdio.h>

int main() {
  int time = 22;
  if (time < 10) {
    printf("Good morning.");
  } else if (time < 20) {
    printf("Good day.");
  } else {
    printf("Good evening.");
  }
  return 0;
}
```

```c
#include <stdio.h>

int main() {
  int myNum = 10;

  if (myNum > 0) {
    printf("The value is a positive number.");
  } else if (myNum < 0) {
    printf("The value is a negative number.");
  } else {
    printf("The value is 0.");
  }

  return 0;
}
```

Short Hand If...Else (Ternary Operator)

Syntax

```
variable = (condition) ? expressionTrue : expressionFalse;
```

```c
#include <stdio.h>

int main() {
  int time = 20;
  if (time < 18) {
    printf("Good day.");
  } else {
    printf("Good evening.");
  }
  return 0;
}
```

V-Ex Tech

Switch Statement

Syntax

Syntax

- ```
  switch(expression) {
    case x:
      // code block
      break;
    case y:
      // code block
      break;
    default:
      // code block
  }
  ```
  ```

```c
#include <stdio.h>

int main() {
  int day = 6;

  switch (day) {
    case 1:
      printf("Monday");
      break;
    case 2:
      printf("Tuesday");
      break;
    case 3:
      printf("Wednesday");
      break;
    case 4:
      printf("Thursday");
      break;
    case 5:
      printf("Friday");
      break;
    case 6:
      printf("Saturday");
      break;
    case 7:
      printf("Sunday");
      break;
  }

  return 0;
}
```

The break Keyword

```c
#include <stdio.h>

int main() {
  int day = 4;

  switch (day) {
  case 6:
    printf("Today is Saturday");
    break;
  case 7:
    printf("Today is Sunday");
    break;
  default:
```

```
    printf("Looking forward to the Weekend");
  }

  return 0;
}
```

For Loop

```
for (statement 1; statement 2; statement 3) {
  // code block to be executed
}
```

```
#include <stdio.h>

int main() {
  int i;

  for (i = 0; i < 5; i++) {
    printf("%d\n", i);
  }

  return 0;
}
```

```
#include <stdio.h>

int main() {
  int i;

  for (i = 0; i <= 10; i = i + 2) {
    printf("%d\n", i);
  }

  return 0;
}
```

```c
#include <stdio.h>

int main() {
  int i, j;

  // Outer loop
  for (i = 1; i <= 2; ++i) {
    printf("Outer: %d\n", i);  // Executes 2 times

    // Inner loop
    for (j = 1; j <= 3; ++j) {
      printf(" Inner: %d\n", j);  // Executes 6 times (2 * 3)
    }
  }

  return 0;
}
```

C Break and Continue

```c
#include <stdio.h>

int main() {
  int i;

  for (i = 0; i < 10; i++) {
    if (i == 4) {
      break;
    }
    printf("%d\n", i);
  }

  return 0;
}
```

Continue

```c
#include <stdio.h>

int main() {
  int i;

  for (i = 0; i < 10; i++) {
    if (i == 4) {
      continue;
    }
    printf("%d\n", i);
  }

  return 0;
}
```

Arrays

```c
int myNumbers[] = {25, 50, 75, 100};
```

```c
#include <stdio.h>

  int main() {
    int myNumbers[] = {25, 50, 75, 100};
    printf("%d", myNumbers[0]);

    return 0;
  }
```

```c
#include <stdio.h>

  int main() {
    int myNumbers[] = {25, 50, 75, 100};
    int i;

    for (i = 0; i < 4; i++) {
      printf("%d\n", myNumbers[i]);
    }
```

```
    return 0;
  }
```

Strings

```
char greetings[] = "Hello World!";
```

```
#include <stdio.h>

  int main() {
    char greetings[] = "Hello World!";
    printf("%s", greetings);

    return 0;
  }
```

```
#include <stdio.h>

  int main() {
    char greetings[] = "Hello World!";
    printf("%c", greetings[0]);

    return 0;
  }
```

Special Characters

```
char txt[] = "We are the so-called "Vikings" from the north.";
```

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

```
int main() {
  char txt[] = "We are the so-called \"Vikings\" from the north.";
  printf("%s", txt);

  return 0;
}
```

String Functions

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

    int main() {
      char alphabet[] = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
      printf("%d", strlen(alphabet));
      return 0;
    }
```

User Input

To get user input**, you can use the** `scanf()` **function**

https://www.javatpoint.com/how-to-run-a-c-program-in-visual-studio-code

go to this link for setting

```
#include <stdio.h>

    int main() {
      // Create an integer variable that will store the number we get from the user
      int myNum;

      // Ask the user to type a number
      printf("Type a number and press enter: \n");

      // Get and save the number the user types
```

```
    scanf("%d", &myNum);

    // Print the number the user typed
    printf("Your number is: %d", myNum);

    return 0;
  }
```

```
#include <stdio.h>

  int main() {
    // Create an int and a char variable
    int myNum;
    char myChar;

    // Ask the user to type a number AND a character
    printf("Type a number AND a character and press enter: \n");

    // Get and save the number AND character the user types
    scanf("%d %c", &myNum, &myChar);

    // Print the number
    printf("Your number is: %d\n", myNum);

    // Print the character
    printf("Your character is: %c\n", myChar);

    return 0;
  }
```

V-Ex Tech

Functions

Syntax

```
void myFunction() {
// code to be executed
}
```

```
#include <stdio.h>

// Create a function
void myFunction() {
  printf("I just got executed!");
}

int main() {
  myFunction(); // call the function
  myFunction();
myFunction();
  return 0;
}
```

Function Parameters

Syntax

```
  returnType functionName(parameter1, parameter2, parameter3) {
 // code to be executed
}
```

```c
#include <stdio.h>

    void myFunction(char name[]) {
      printf("Hello %s\n", name);
    }

    int main() {
      myFunction("Liam");
      myFunction("Jenny");
      myFunction("Anja");
      return 0;
    }
```

Multiple Parameters

```c
#include <stdio.h>

    void myFunction(char name[], int age) {
      printf("Hello %s. You are %d years old\n", name, age);
    }

    int main() {
      myFunction("Liam", 3);
      myFunction("Jenny", 14);
      myFunction("Anja", 30);
      return 0;
    }
```

V-Ex Tech

```c
#include <stdio.h>

    int myFunction(int x) {
      return 5 + x;
    }

    int main() {
      printf("Result is: %d", myFunction(3));
      return 0;
    }
```

Function Declaration and Definition

```c
#include <stdio.h>

    // Create a function
    void myFunction() {
      printf("I just got executed!");
    }

    int main() {
      myFunction(); // call the function
      return 0;
    }
```

```c
#include <stdio.h>

    // Function declaration
    void myFunction();

    // The main method
    int main() {
      myFunction();  // call the function
```

```
    return 0;
  }

  // Function definition
  void myFunction() {
    printf("I just got executed!");
  }
```

Recursion

```
#include <stdio.h>

  int sum(int k);

  int main() {
    int result = sum(10);
    printf("%d", result);
    return 0;
  }

  int sum(int k) {
    if (k > 0) {
      return k + sum(k - 1);
    } else {
      return 0;
    }
  }
```

10 + sum(9)
10 + (9 + sum(8))
10 + (9 + (8 + sum(7)))
...
10 + 9 + 8 + 7 + 6 + 5 + 4 + 3 + 2 + 1 + sum(0)
10 + 9 + 8 + 7 + 6 + 5 + 4 + 3 + 2 + 1 + 0

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

  int main() {
```

```
    printf("%f", pow(4, 3));
    return 0;
  }
```

File Handling

In C, you can create, open, read, and write to files by declaring a pointer of type `FILE`, and use the `fopen()` function:

```
FILE *fptr
fptr = fopen(filename, mode);
```

`w` - Writes to a file
`a` - Appends new data to a file
`r` - Reads from a file

```
FILE *fptr;

// Create a file
fptr = fopen("filename.txt", "w");

// Close the file
fclose(fptr);
```

Write To a File

The `w` mode means that the file is opened for writing**. To insert content to it, you can use the `fprint()` function and add the pointer variable (`fptr` in our example) and some text:**

```
FILE *fptr;

// Open a file in writing mode
fptr = fopen("filename.txt", "w");
```

```
// Write some text to the file
fprintf(fptr, "Some text");

// Close the file
fclose(fptr);
```

Append Content To a File

```
FILE *fptr;

// Open a file in append mode
fptr = fopen("filename.txt", "a");

// Append some text to the file
fprintf(fptr, "\nHi everybody!");

// Close the file
fclose(fptr);
```

Read a File

```
FILE *fptr;

// Open a file in read mode
fptr = fopen("filename.txt", "r");
```

This will make the `filename.txt` opened for reading.

```
FILE *fptr;

// Open a file in read mode
fptr = fopen("filename.txt", "r");

// Store the content of the file
char myString[100];
```

V-Ex Tech

```c
FILE *fptr;

// Open a file in read mode
fptr = fopen("filename.txt", "r");

// Store the content of the file
char myString[100];

// Read the content and store it inside myString
fgets(myString, 100, fptr);

// Print the file content
printf("%s", myString);

// Close the file
fclose(fptr);
```

V-Ex Tech

```c
FILE *fptr;

// Open a file in read mode
fptr = fopen("filename.txt", "r");

// Store the content of the file
char myString[100];

// Read the content and print it
while(fgets(myString, 100, fptr)) {
  printf("%s", myString);
}

// Close the file
fclose(fptr);
```