Problem Solving and Programming using C (PSPC)

Dr. M. Raja Roy

Associate Professor

Anil Neerukonda Institute of Technology and Sciences

www.anits.edu.in

www.mrrtechnical.co.in

COMPUTER

COMPUTER Stands for Common Operating Machine Purposely Used for Technological and Educational Research.

Compute is an electronic device for storing and processing data, typically in binary form, according to instructions given to it in a variable program.

Charles Babbage, an English mechanical engineer and polymath, originated the concept of a programmable computer. Considered the "father of the computer", he conceptualized and invented the first mechanical computer in the early 19th century.

Basics: Computer Input/Output, Software and hardware

The computer mainly had two major components:

1. Hardware
2. Software

Hardware

Computer hardware includes the physical parts of a computer, such as a case, central processing unit (CPU), random access memory (RAM), monitor, and mouse which processes the input according to the set of instructions provided to it by the user and gives the desired output.

Hardware


portable
media player
(output device)
system unit
(processor, memory,
and storage devices)
screen
Web cam
(input device)
printer
(output device)
monitor
(output device)
speakers
(output device)
CD/DVD drive
(storage device)
hard disk drive
(storage device)
keyboard
(input device)
mouse
(input device)
microphone
(input device)
digital camera
(input device)
scanner
(input device)
card reader/writer
(storage device)
USB flash drive
(storage device)
external hard disk
(storage device)
modem
(communications device)
memory cards

Hardware

The **input device** is usually a keyboard where programs and data are entered into the computers. Examples of other input devices include a mouse, a pen or stylus, a touch screen, or an audio input unit.

Common Input Devices of Computer

Hardware

The central processing unit (CPU) is responsible for executing instructions such as arithmetic calculations, comparisons among data, and movement of data inside the system.

Hardware

The output device is usually a monitor or a printer to show output. If the output is shown on the monitor, we say we have a soft copy. If it is printed on the printer, we say we have a hard copy.

Hardware

Auxiliary storage, also known as secondary storage. It is the place where the programs and data are stored permanently. When we turn off the computer, or programs and data remain in the secondary storage.

Computer Software

Software is a set of instructions, data or programs used to operate computers and execute specific tasks.

The two main categories of software are

1. Application software
2. System software

System Software

System software is designed to run a computer's hardware and provides a platform for applications.

Ex : Window, Linex, Mac OS, Android for Mobiles etc.

Application Software

An application is software that fulfills a specific need or performs tasks.

Ex : MS Office, Auto CAD, Photoshop, Adobe Premiere Pro, After Effects etc.

Computer Languages

To write a program for a computer, we must use a computer language. Over the years computer languages have evolved from machine languages to natural languages.

1940’s	Machine level Languages
1950’s	Symbolic Languages
1960’s	High-Level Languages

Machine Languages

In the earliest days of computers, the only programming languages available were machine languages. Each computer has its own machine language, which is made of streams of 0's and 1's.

Internal circuits of a computer are made of switches transistors and other electronic devices that can be in one of two states: **off** or **on**.

Off state is represented by 0.

On state is represented by 1.

The only language understood by computer hardware is machine language.

Symbolic Languages

In early 1950's Admiral Grace Hopper, A mathematician and naval officer developed the concept of a special computer program that would convert programs into machine language.

Computer does not understand symbolic language it must be translated to the machine language.

A special program called ***assembler***, translates symbolic code into machine language.

High Level Languages

Symbolic languages greatly improved programming efficiency; they still required programmers to concentrate on the hardware that they were using. Working with symbolic languages was also very tedious because each machine instruction has to be individually coded. The desire to improve programmer efficiency and to change the focus from the computer to the problem being solved led to the development of high-level language.

Introduction to Problem Solving

Problem solving is a scientific technique to discover and implement the answer to a problem, and the computer is a manipulating device that follows a set of commands known as a program.

Problem solving Aspect: Procedure

(Steps Involved in Problem Solving)

In fact, the task of problem solving is not that of the computer.

It is the programmer who has to write down the solution to the problem in terms of simple operations which the computer can understand and execute.

In order to solve a problem by the computer, one has to pass though certain stages or steps. They are

Problem solving Aspect: Procedure

(Steps Involved in Problem Solving)

1. Problem Identification
2. Understanding the problem
3. Analyzing the problem
4. Developing the solution
5. Coding and implementation.

Problem solving Aspect: Procedure

(Steps Involved in Problem Solving)

Understanding the problem:

Here we try to understand the problem to be solved in totally. Before with the next stage or step, we should be absolutely sure about the objectives of the given problem.

Analyzing the problem:

After understanding thoroughly the problem to be solved, we look different ways of solving the problem and evaluate each of these methods. The idea here is to search an appropriate solution to the problem under consideration. The end result of this stage is a broad overview of the sequence of operations that are to be carries out to solve the given problem.

Developing the solution:

Here the overview of the sequence of operations that was the result of analysis stage is expanded to form a detailed step by step solution to the problem under consideration.

Coding and implementation:

The last stage of the problem solving is the conversion of the detailed sequence of operations in to a language that the computer can understand. Here each step is converted to its equivalent instruction or instructions in the computer language that has been chosen for the implantation.

Algorithm:

- A set of sequential steps usually written in Ordinary Language to solve a given problem is called Algorithm.

- An algorithm can be defined as “a complete, unambiguous, finite number of logical steps for solving a specific problem “

Algorithm:

- It may be possible to solve to problem in more than one ways, resulting in more than one algorithm.
- The choice of various algorithms depends on the factors like reliability, accuracy and easy to modify.
- The most important factor in the choice of algorithm is the time requirement to execute it, after writing code in High-level language with the help of a computer.
- The algorithm which will need the least time when executed is considered the best

Steps involved in algorithm development:

Step1: Identification of input: For an algorithm, there are quantities to be supplied called input and these are fed externally. The input is to be identified first for any specified problem.

Step2: Identification of output: From an algorithm, at least one quantity is produced, called for any specified problem.

Step3: Identification the processing operations: All the calculations to be performed in order to lead to output from the input are to be identified in an orderly manner.

Step4: Processing Definiteness: The instructions composing the

Steps involved in algorithm development:

Step4: Processing Definiteness: The instructions composing the algorithm must be clear and there should not be any ambiguity in them.

Step5 : Processing Finiteness : If we go through the algorithm, then for all cases, the algorithm should terminate after a finite number of steps.

Step6 : Possessing Effectiveness : The instructions in the algorithm must be sufficiently basic and in practice they can be carries out easily.

Flowchart:

- A flowchart is visual or graphical representation of an algorithm.
- A flowchart is a diagram made up of boxes, diamonds and other shapes, connected by arrows. Each shape represents a step of the solution process and the arrow represents the order or link among the steps.
- A flow chart is a step by step diagrammatic representation of the logic paths to solve a given problem.

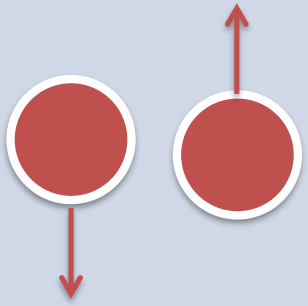
Advantages of Flowcharts:

- The flowchart shows the logic of a problem displayed in pictorial fashion which felicitates easier checking of an algorithm.
- The Flowchart is good means of communication to other users. It is also a compact means of recording an algorithm solution to a problem.
- The flowchart allows the problem solver to break the problem into parts. These parts can be connected to make master chart.
- The flowchart is a permanent record of the solution which can be consulted at a later time.

The symbols that we make use while drawing flowcharts as given below are as per conventions followed by **International Standard Organization (ISO).**

Flowchart symbol	Symbol name	Function	Description
	Oval	START/ STOP	Also called "**Terminator**" symbol. It indicates where the flow starts and ends.
	Parallelogram	INPUT / OUTPUT/ READ / PRINT	Also called **data symbol**, this parallelogram shape is used to input or output data indicators
	Rectangle	Process Indicators	Also called "**Action Symbol**," it represents a process, action, or a single step and used to indicate any set of processing operation such as for storing arithmetic operations.

Flowchart symbol	Symbol name	Function	Description
	Diamond	Decision Makers	The diamond is used for indicating the step of **decision making** and therefore known as decision box. A decision or branching point, usually a yes/no or true/ false question is asked, and based on the answer, the path gets split into two branches.
	Arrows or Flow Lines	Connectors	**Connector to show order of flow between shapes**. Flow lines indicate the direction being followed in the flowchart. In a Flowchart, every line must have an arrow on it to indicate the direction. The arrows may be in any direction

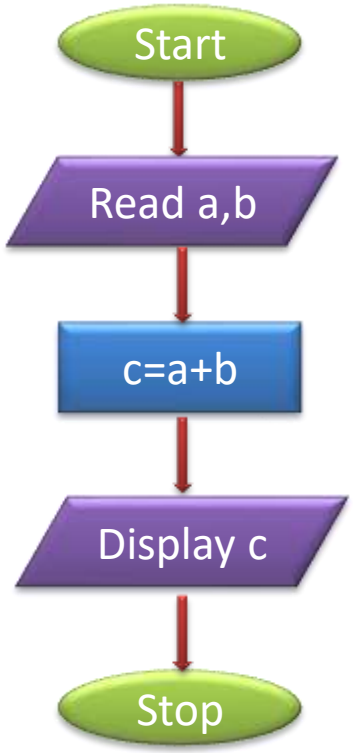
Flowchart symbol	Symbol name	Function	Description
	Circles with arrows	On- Page connectors	Circles are used to join the different parts of a flowchart and these circles are called on-page connectors. A flowchart may run in to several pages. The parts of the flowchart on different pages are to be joined with each other. The parts to be joined are indicated by the circle.
	Square Pentagon	Off-page connectors	This connector represents a break in the path of flowchart which is too large to fit on a single page. It is similar to on-page connector. The connector symbol marks where the algorithm ends on the first page and where it continues on the second.

To find sum of two numbers

Algorithm

1. Start
2. Read a, b
3. c=a+b
4. Display c
5. Stop

Flow Chart



Program

onlinegdb.com/online_c_compiler

Run | Debug | Stop | Share | Save | { } Beautify

main.c

```
/*************************************************************
Program to add two numbers
Regd No. :
Name :
Date :
*************************************************************/

#include <stdio.h>

int main()
{
    int a,b,c;

    printf("Enter value of a :");
    scanf("%d",&a);

    printf("\nEnter value of b :");
    scanf("%d",&b);

    c=a+b;

    printf("\nSum of the given two numbers is : %d",c);

    return 0;
}
```

```
Enter value of a :5

Enter value of b :3

Sum of the given two numbers is : 8

...Program finished with exit code 0
Press ENTER to exit console.
```

History of C

Origins at Bell Labs (1970-1972):

C was created by Dennis Ritchie at Bell Labs (AT&T's Murray Hill, New Jersey, research and development facility) between 1970 and 1972.

Ritchie developed C as an evolution of the B programming language, which itself was based on BCPL.

History of C

Subsequent Revisions (1999, 2011, 2018):

The C language underwent further revisions, with the ISO/IEC 9899 standard being revised in 1999 (commonly referred to as C99), 2011 (C11), and 2018 (C18). These revisions introduced new features and improvements.

Token in C Program : Token are the smallest individual units in a program.

1. Keywords : if, else, for etc
2. Constants : #define PI 3.14, MAX 200 etc
3. Variables : x,y,z,b etc
4. Data types : int, float etc
5. Operators : +,-,*,/ etc

C Language Elements

The C programming language consists of various elements that are used to write and structure programs. Here are some essential C language elements:

Keywords: Reserved words with specific meanings in C. Examples include int, float, char, if, while, and return.

Identifiers: Names given to variables, functions, arrays, etc., by the programmer. Identifiers must follow certain rules, such as starting with a letter or underscore.

Variables: Containers used to store data values. They must be declared with a data type (e.g., int, float, char) before use.

Data Types: Defines the type of data a variable can hold. Common data types include int (integer), float (floating-point), char (character), and double (double-precision floating-point).

Variables: Containers used to store data values. They must be declared with a data type (e.g., int, float, char) before use.

Data Types: Defines the type of data a variable can hold. Common data types include int (integer), float (floating-point), char (character), and double (double-precision floating-point).

Operators:

Symbols that perform operations on variables and values. Examples include arithmetic operators (+, -, *, /), relational operators (==, !=, <, >), and logical operators (&&, ||, !).

Control Flow Statements:

Statements that control the flow of program execution. Common control flow statements include if, else, switch, while, for, and do-while.

Functions:

Blocks of code that perform a specific task. Functions are essential for organizing code into modular and reusable components. Every C program must have a main function, which serves as the entry point.

Arrays:

Collections of similar data items stored in contiguous memory locations. Array elements are accessed using an index.

Pointers: Variables that store memory addresses. Pointers are powerful tools in C for dynamic memory allocation and manipulation.

Structures: User-defined data types that allow bundling different types of data under a single name. Each data member in a structure can have a different data type.

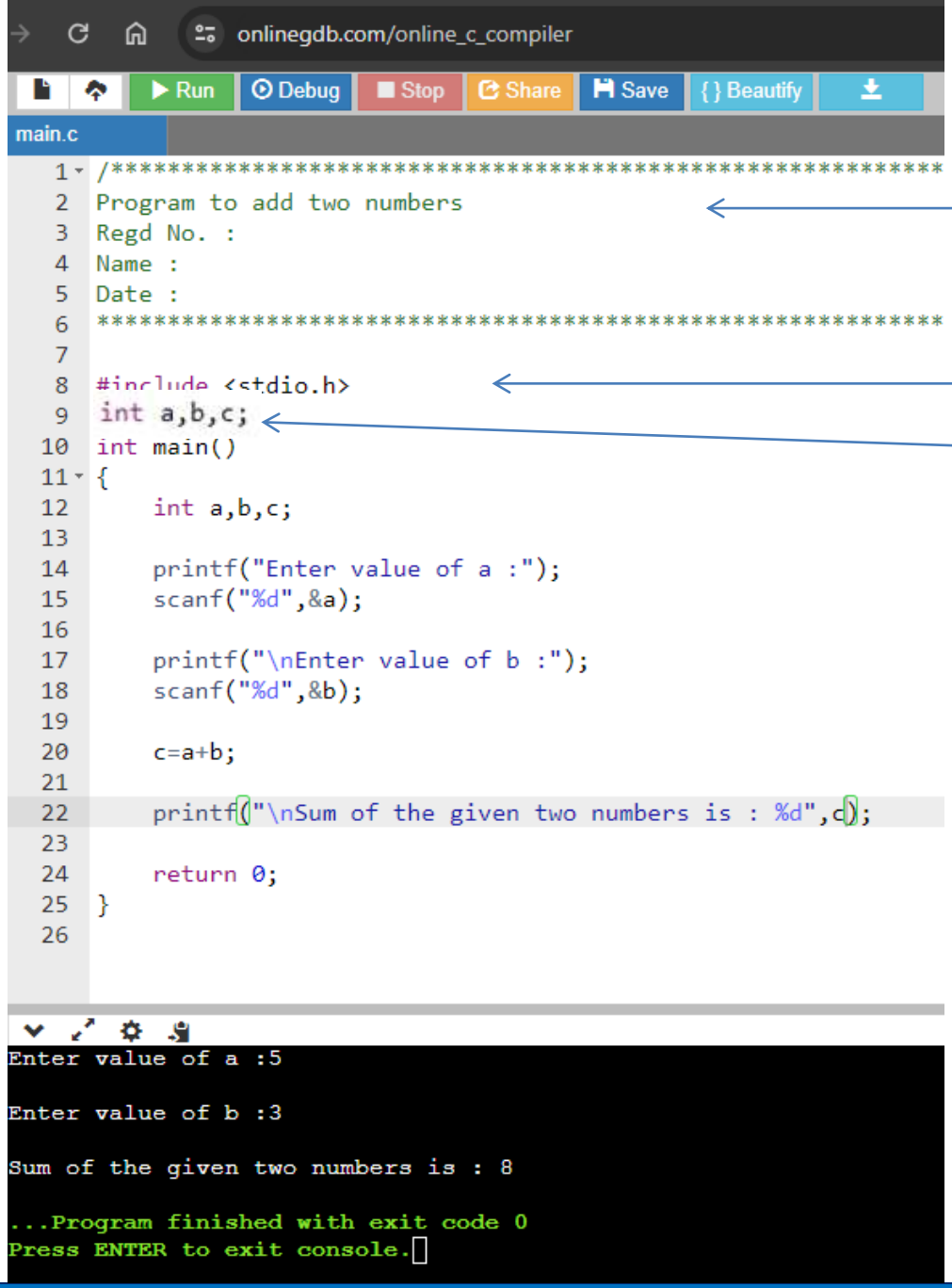
File Handling: C provides functions and structures for performing input and output operations on files. Common file-related functions include **fopen**, **fclose**, **fprintf**, and **fscanf**.

Structure of C Program

1. Documentation
2. Link Section
3. Definition section : #define PI 3.14

   #define MAX 1000

1. Global declaration section
2. Main Section
3. Sub Programs



Documentation

Link Section

Global declaration section

Header Files : Standard Library Header Files In C

Included with the C compiler, these header files provide declarations and definitions for all variables and predefined functions or methods found in the C standard library.

Examples include the header files stdio.h, string.h, math.h, etc., each of which caters to specific functions/ operations. The header file names are enclosed in angular brackets inside the code to indicate that the file is located in the standard folder for header files in C.

#include <stdio.h>

The name stands for standard input/output header, and it contains functions for standard input and output operations such as printf() and scanf() etc.

Some standard functions that form a part of this header file are:

1. printf(): Used to print formatted output to the console or a file.

2. scanf(): Used to read formatted input from the console or a file.

#include <stdio.h>

3. fgets(): Used to read a line of text from a file or the console.

4. fopen(): Used to open a file.

5. fclose(): Used to close a file.

6. fseek(): Used to set the file position indicator for a file.

7. fread(): Used to read data from a file.

8. fwrite(): Used to write data to a file.

#include <stdlib.h>

The term stdlib.h stands for standard library header, which contains functions for general-purpose tasks such as memory allocation, process control, conversions, and searching.

Some standard functions that form a part of this header file are:

1. malloc(): Used to dynamically allocate memory.
2. free(): Used to free dynamically allocated memory.
3. atoi(): Used to convert a string to an integer.

#include <stdlib.h>

4. atof(): Used to convert a string to a floating-point number.

5. rand(): Used to generate a random number.

6. qsort(): Used to sort an array.

#include <string.h>

This header contains functions for string manipulation and memory manipulation.

Some standard functions that form a part of this header file are:

1. strlen(): Used to get the length of a string.
2. strcpy(): Used to copy one string to another.
3. strcat(): Used to concatenate two strings.
4. strstr(): Used to find a substring in a string.

#include <math.h>

This header file contains mathematical functions such as trigonometric functions, logarithmic functions, and exponential functions.

Some standard functions that form a part of this header file are:

1. sin(): Used to calculate the sine of an angle.
2. cos(): Used to calculate the cosine of an angle.
3. tan(): Used to calculate the tangent of an angle.
4. sqrt(): Used to calculate the square root of a number.

#include <math.h>

5. pow(): Used to raise a number to a power.

6. ceil(): Used to round a number up to the nearest integer.

7. floor(): Used to round a number down to the nearest integer.

Identifiers

Identifiers are user defined word used to name of entities like variables, arrays, functions, structures etc.

Rules for naming identifiers/ Variable are:

1) Name should only consists of alphabets (both upper and lower case), digits and underscore (_) sign.
2) First characters should be alphabet or underscore.
3) Name should not be a keyword.
4) Since **C** is a case sensitive, the upper case and lower case considered differently, for example code, Code, CODE etc.

Rules for naming identifiers/ Variable are:

5. Identifiers are generally given in some meaningful name such as value, net_salary, age, data etc.

Keyword

There are certain words reserved for doing specific task, these words are known as reserved word or keywords. These words are predefined and always written in lower case or small letter. These keywords cann't be used as a variable name as it assigned with fixed meaning.

Some examples are **int, short, signed, unsigned, default, volatile, float, long, double, break, continue, typedef, static, do, for, union, return, while, do, extern, register, enum, case, goto, struct, char, auto, const** etc.

Constants

Constant is a any value that cannot be changed during program execution.

In C, any number, single character, or character string is known as a constant.

A constant is an entity that doesn't change whereas a variable is an entity that may change.

Numeric constant: Numeric constant consists of digits. It required minimum size of 2 bytes and max 4 bytes. It may be positive or negative but by default sign is always positive. No comma or space is allowed within the numeric constant and it must have at least 1 digit. The allowable range for integer constants is -32768 to 32767.

Truly speaking the range of an Integer constant depends upon the compiler.

For a 16-bit compiler like Turbo C or Turbo C++ the range is –32768 to 32767. For a 32-bit compiler the range would be even greater.

Real constant : Real constant is also called floating point constant.

-real constant must have at least one digit.

-It must have a decimal point.

-It could be either positive or negative.

-Default sign is positive.

-No commas or blanks are allowed within a real constant. Ex.: +325.34

Character constant : Character constant represented as a single character enclosed within a single quote. These can be single digit, single special symbol or white spaces such as ‘9’,’c’,’$’, ‘ ’ etc.

Every character constant has a unique integer like value in machine’s character code.

Variables

Variable is a data name which is used to store some data value.

Syntax:

```
int a;

char c;

float f;
```

Data Types

Data types refer to an extensive system used for declaring variables or functions of different types before its use.

The type of a variable determines how much space it occupies in storage.

There are two types of type qualifier in C

Size qualifier: short, long

Sign qualifier: signed, unsigned

Data Types

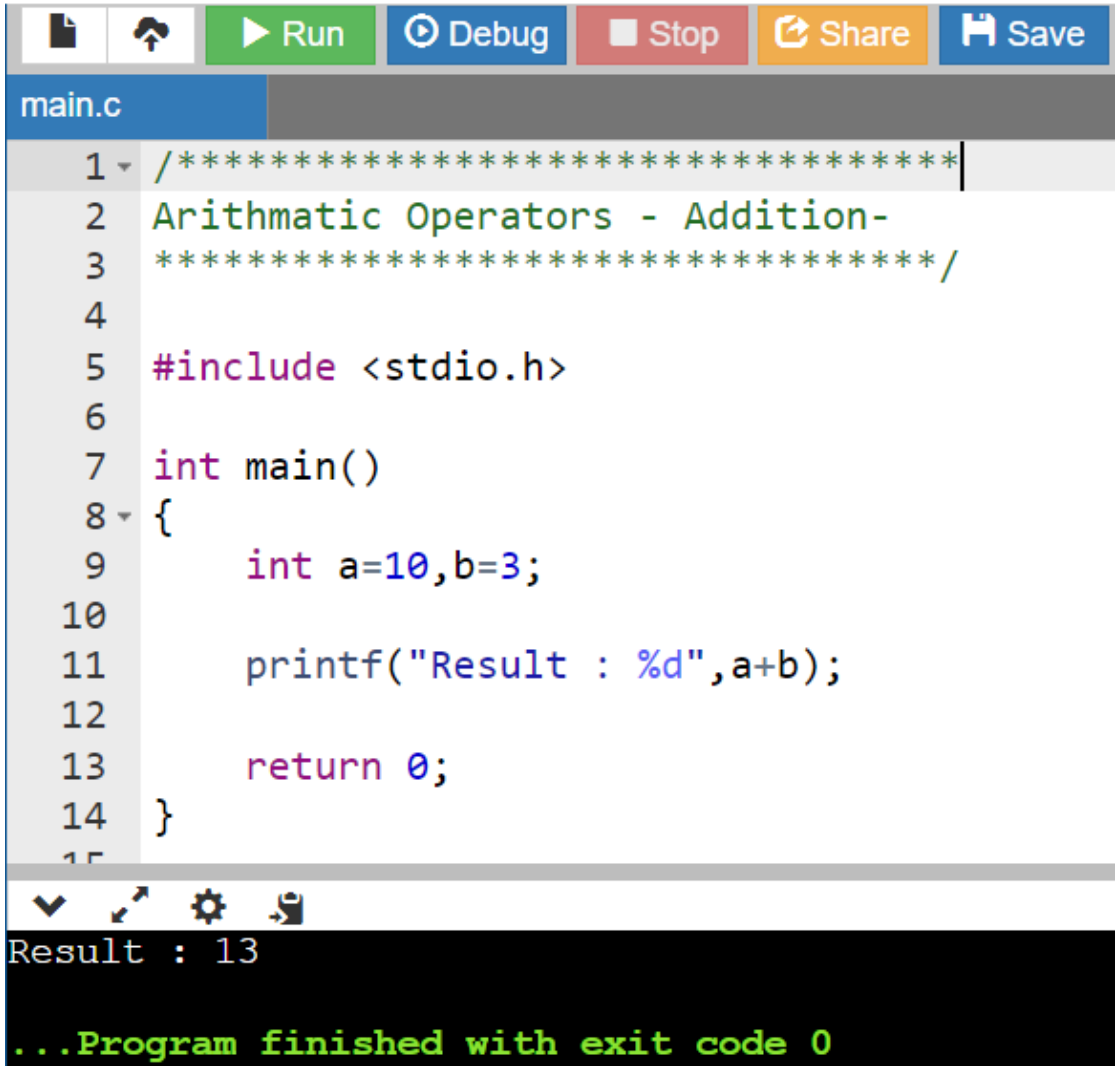
Data Type	Size of Data Type (in Bytes)	Range	Format Specifier
char	1	-128 to 127	%c
int	2	-2^{15} to $2^{15}-1$	%d
float	4	-2^{31} to $2^{31}-1$	%f
double	8	-2^{63} to $2^{63}-1$	%lf
MODIFIERS			
unsigned char	1	0 to 255	%c
unsigned int	2	0 to $2^{16}-1$	%u
short int	1	-128 to 127	%d
unsigned short int	1	0 to 255	%u
long int	4	-2^{31} to $2^{31}-1$	%ld
unsigned short int	4	0 to $2^{32}-1$	%lu
long double	10	-2^{79} to $2^{79}-1$	%Lf

Operators

Arithmetic Operators	-	+ , - , * , / , %
Relational Operators	-	< , > , <= , >= , == , !=
Assignment Operators	-	= , += , -= , *= , /= , %=
Logical Operators	-	and , or , not
Increment Operators	-	++ , --

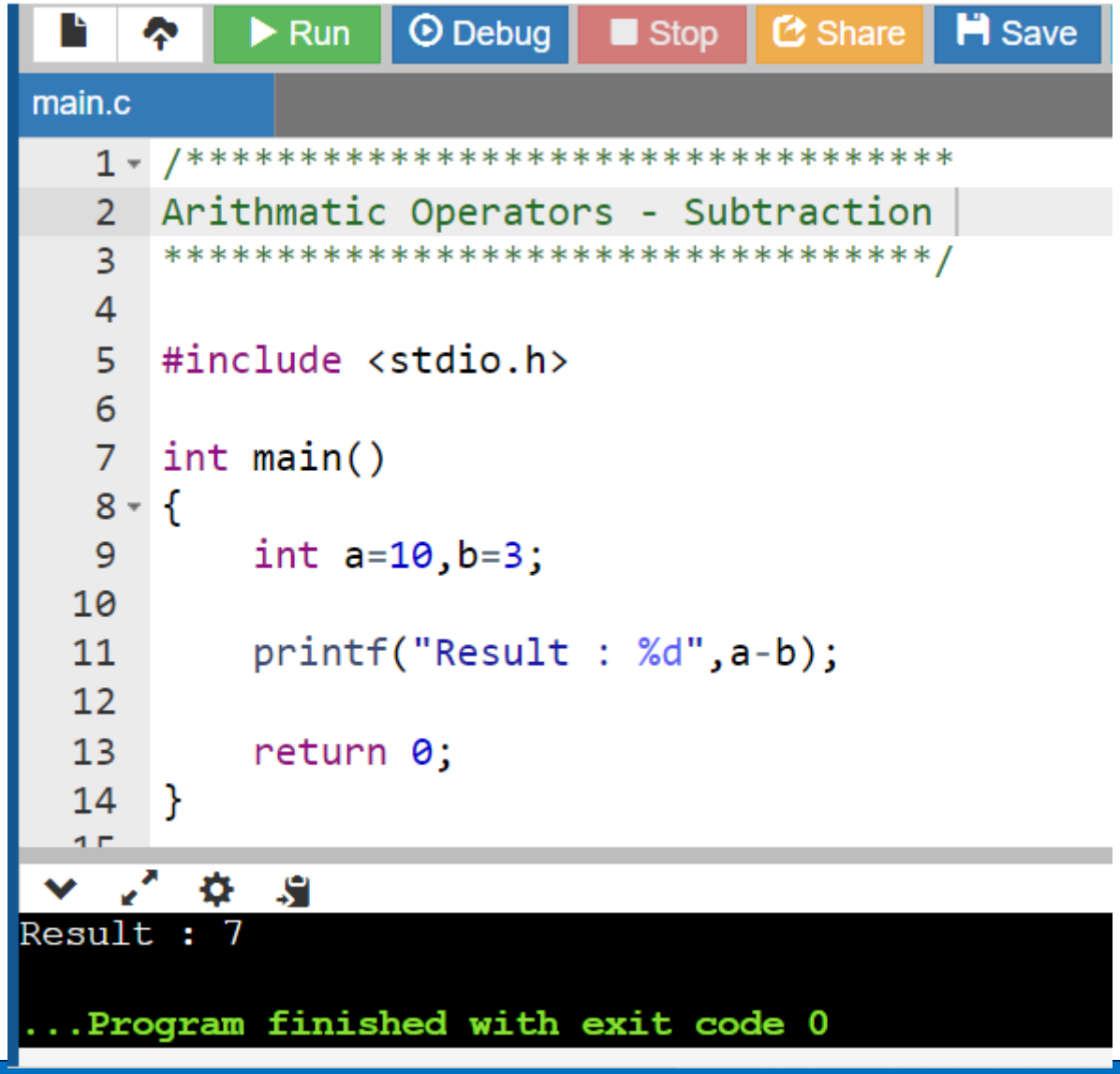
Arithmetic Operators

Addition : +



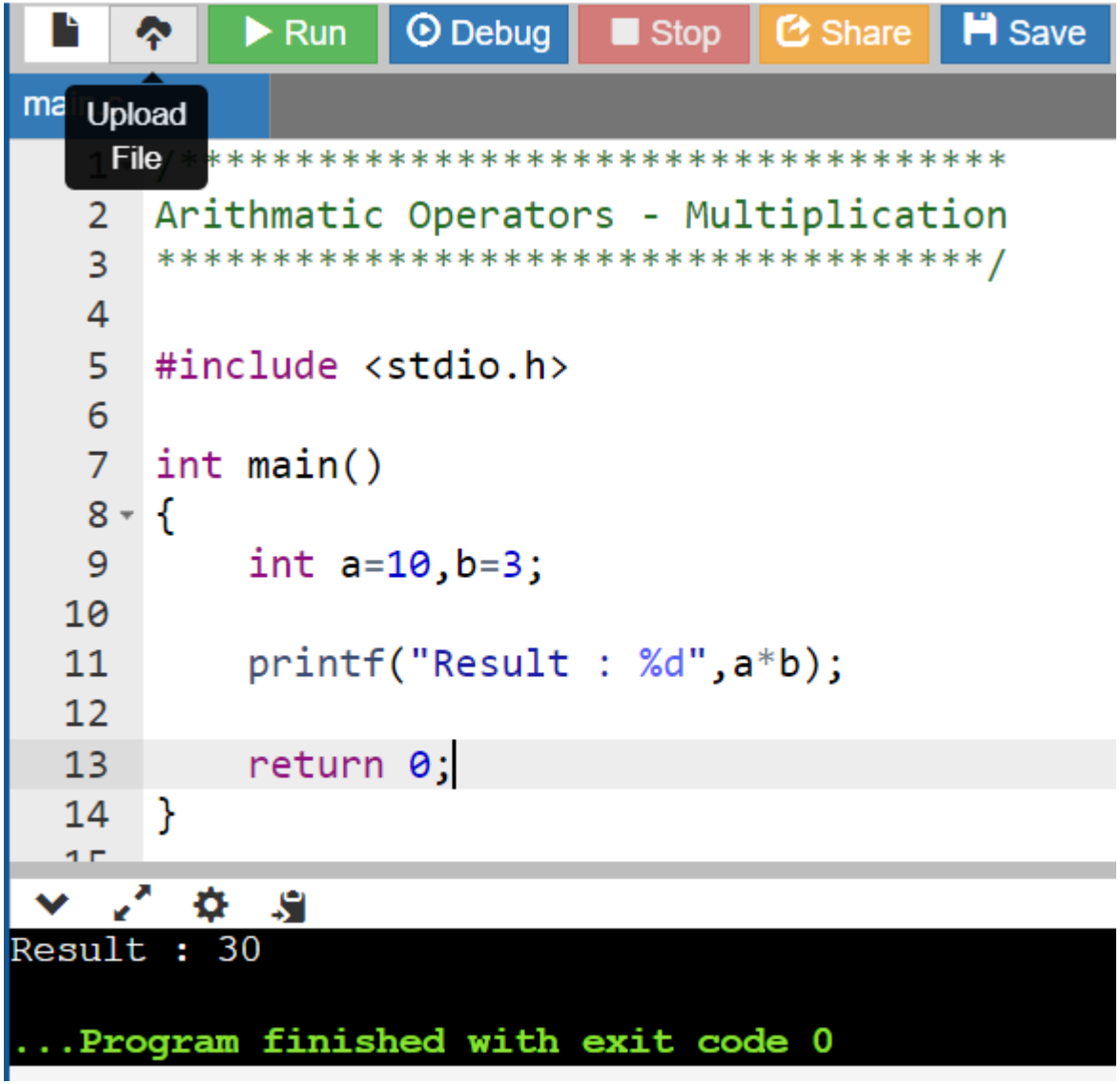
Arithmetic Operators

Subtraction : -



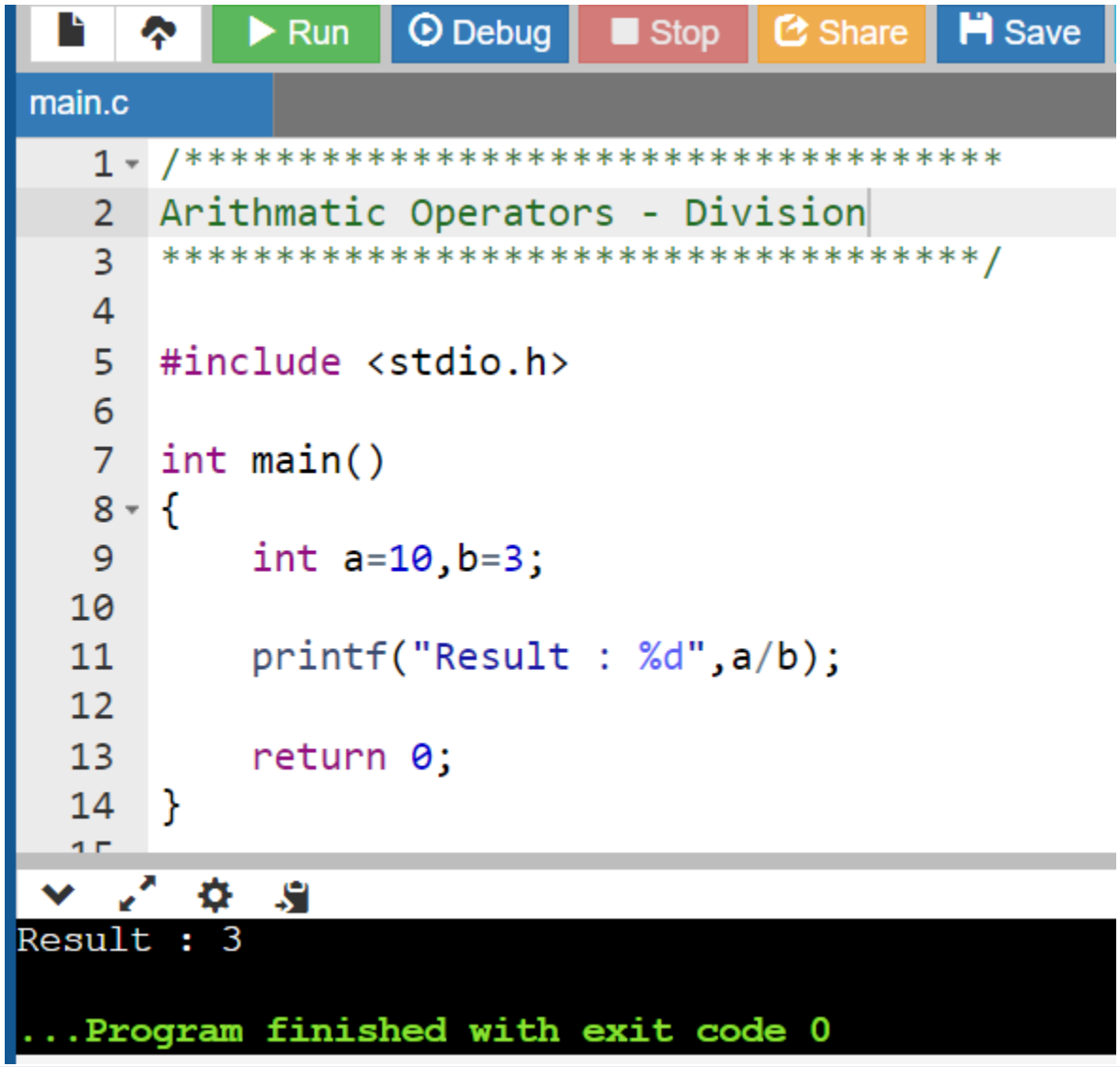
Arithmetic Operators

Multiplication : *



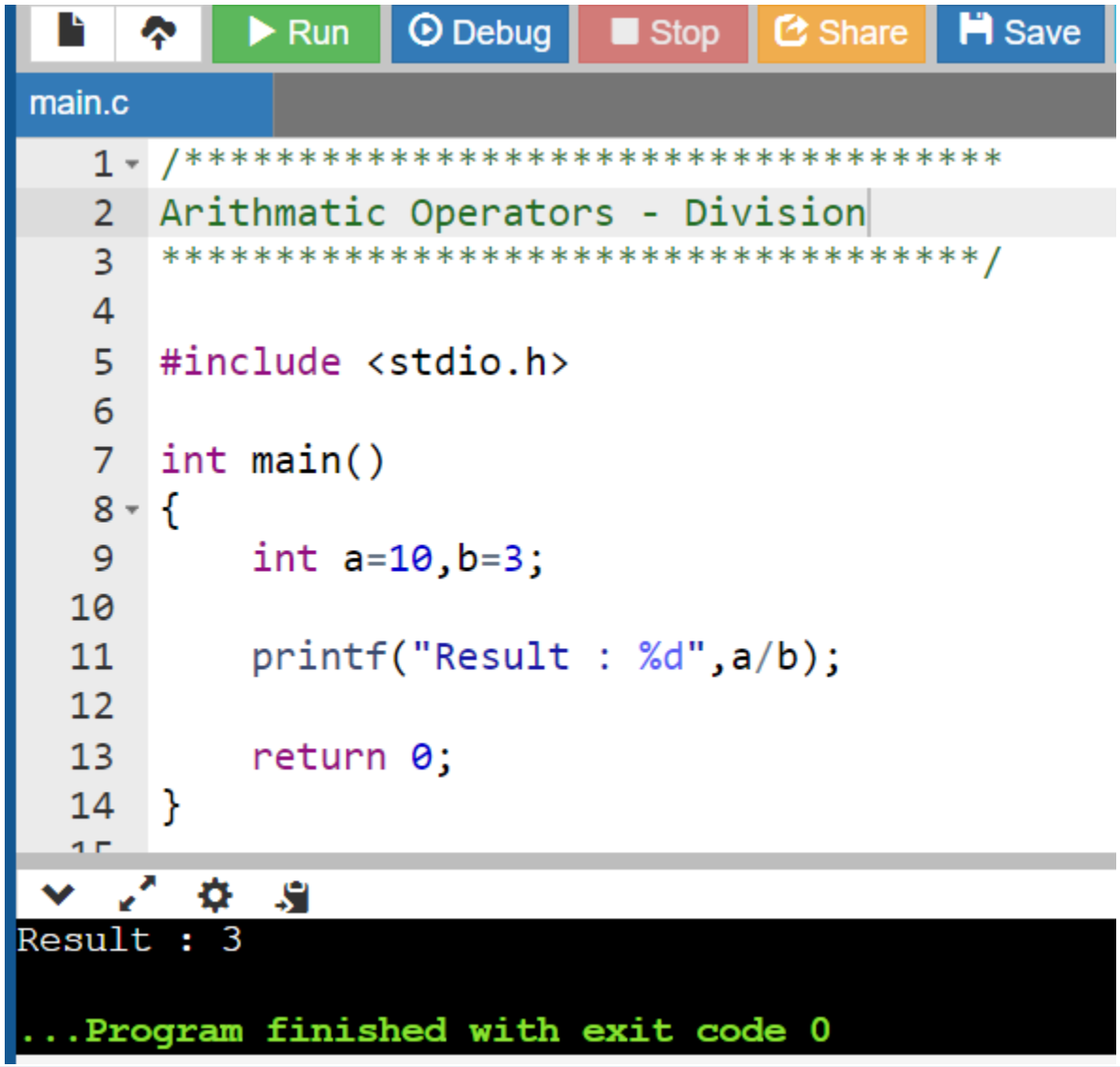
Arithmetic Operators

Division : /



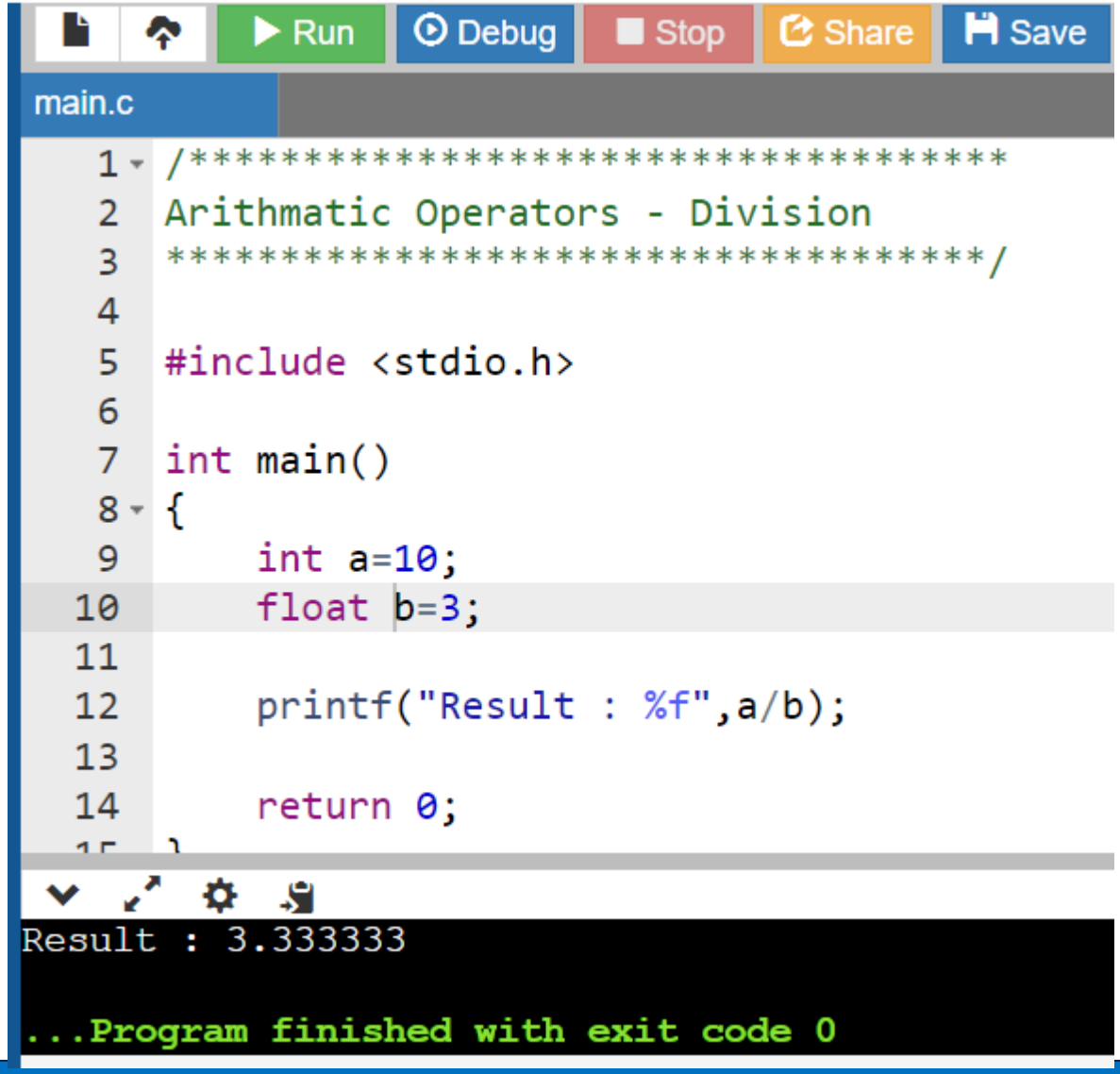
Arithmetic Operators

Division : /



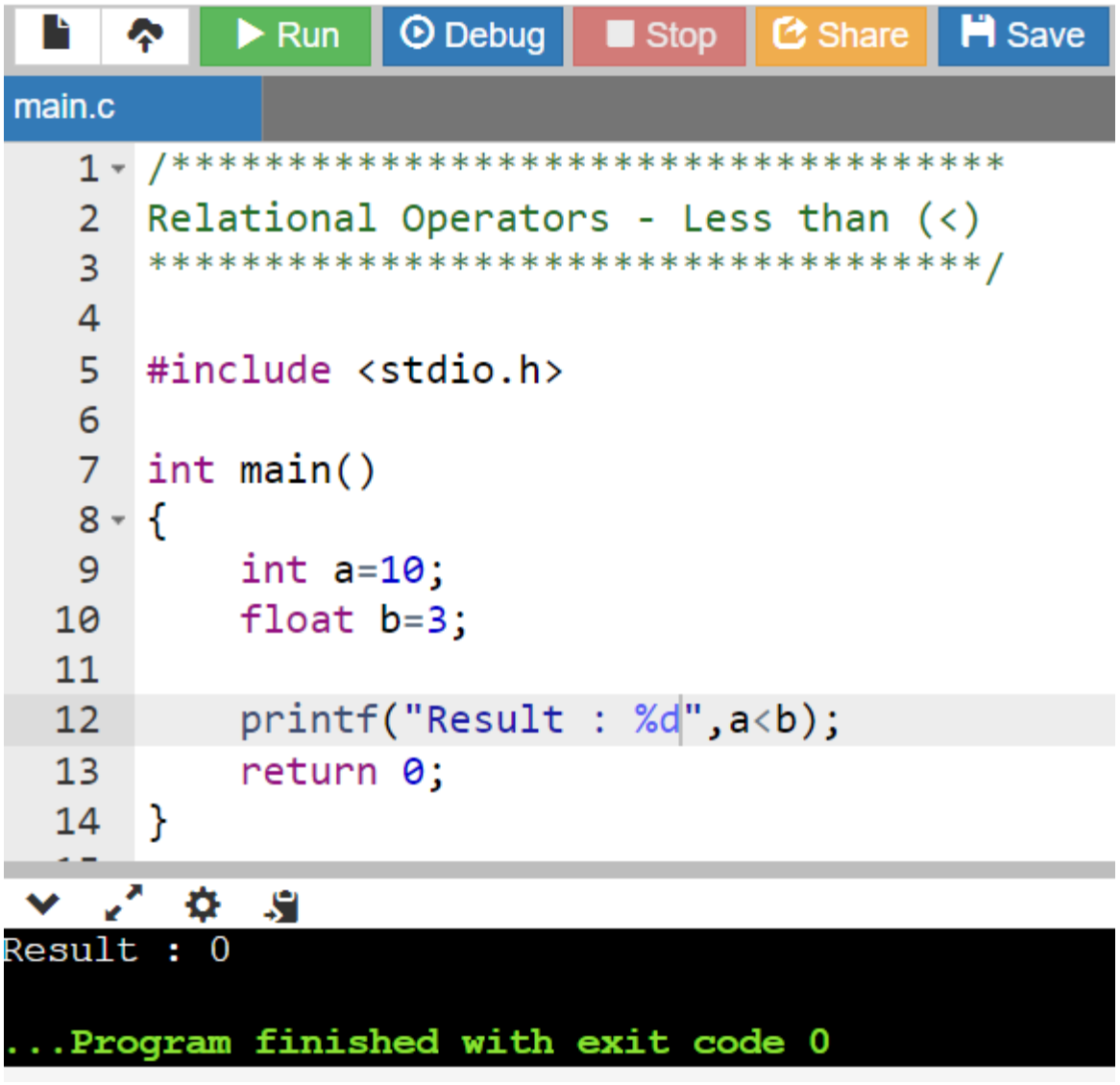
Arithmetic Operators

Division : /



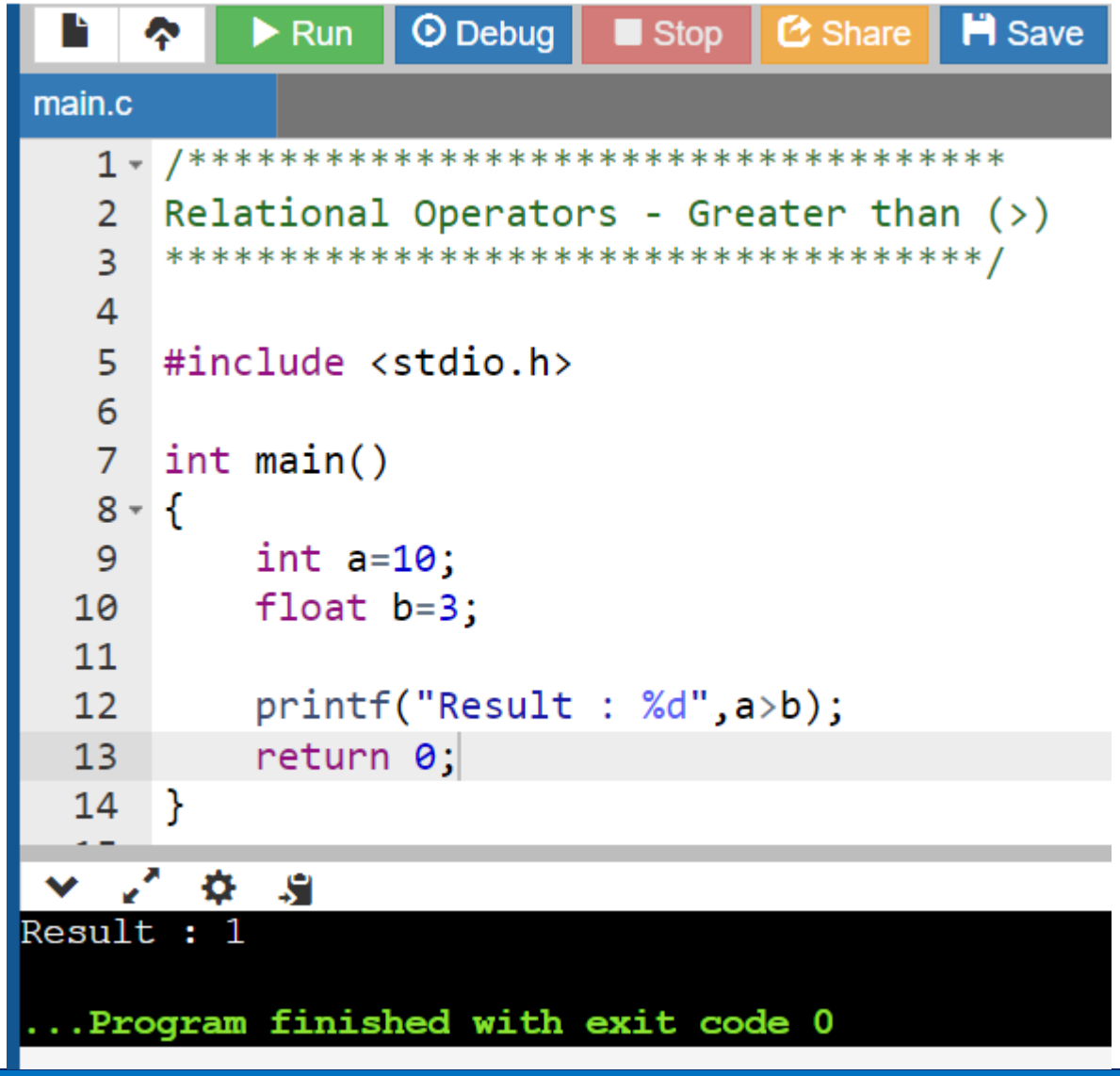
Relational Operators : Returns Boolean value (1-TRUE,0-False)

Less than : <



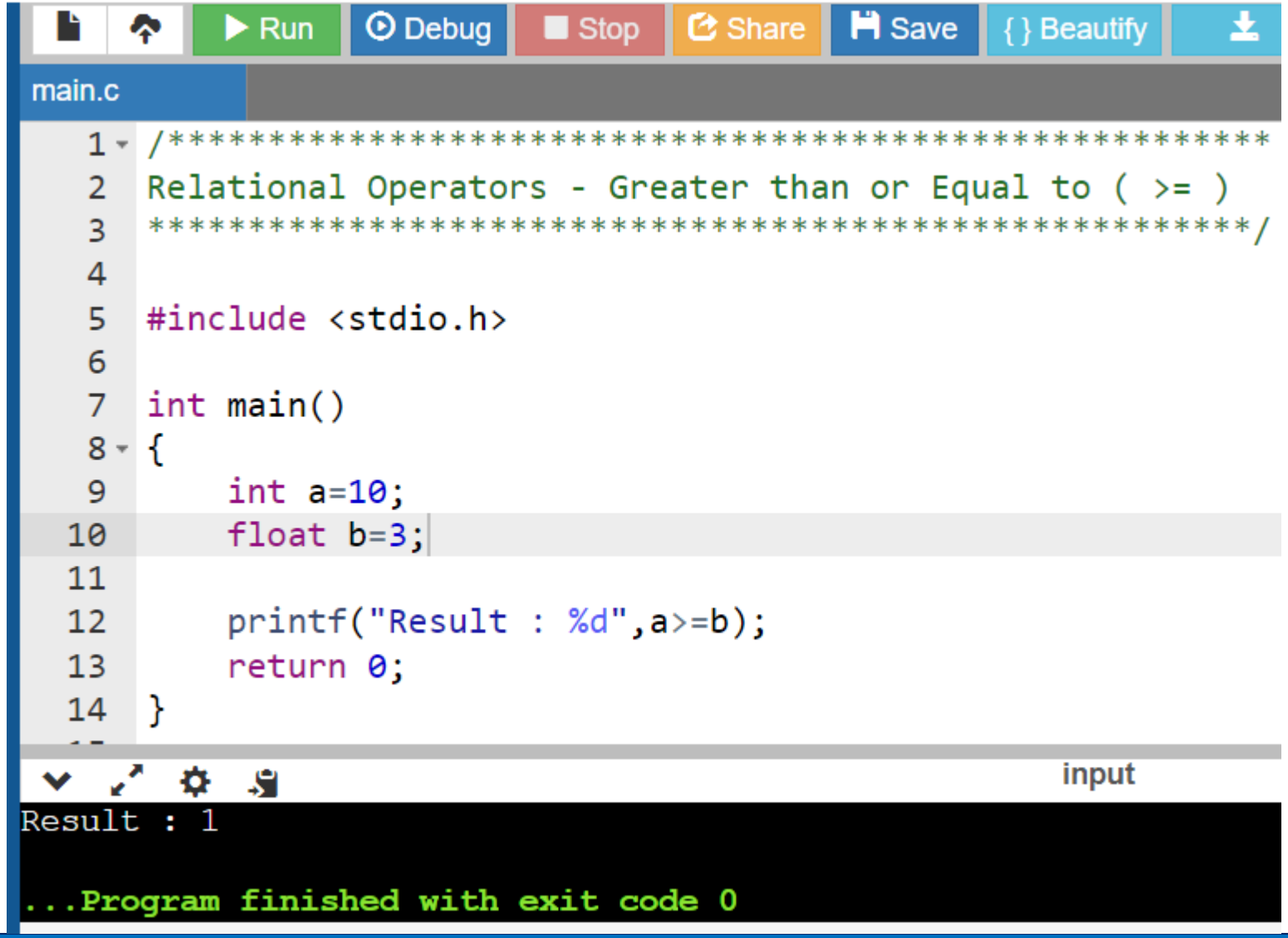
Relational Operators : Returns Boolean value (1-TRUE,0-False)

Greater than : >



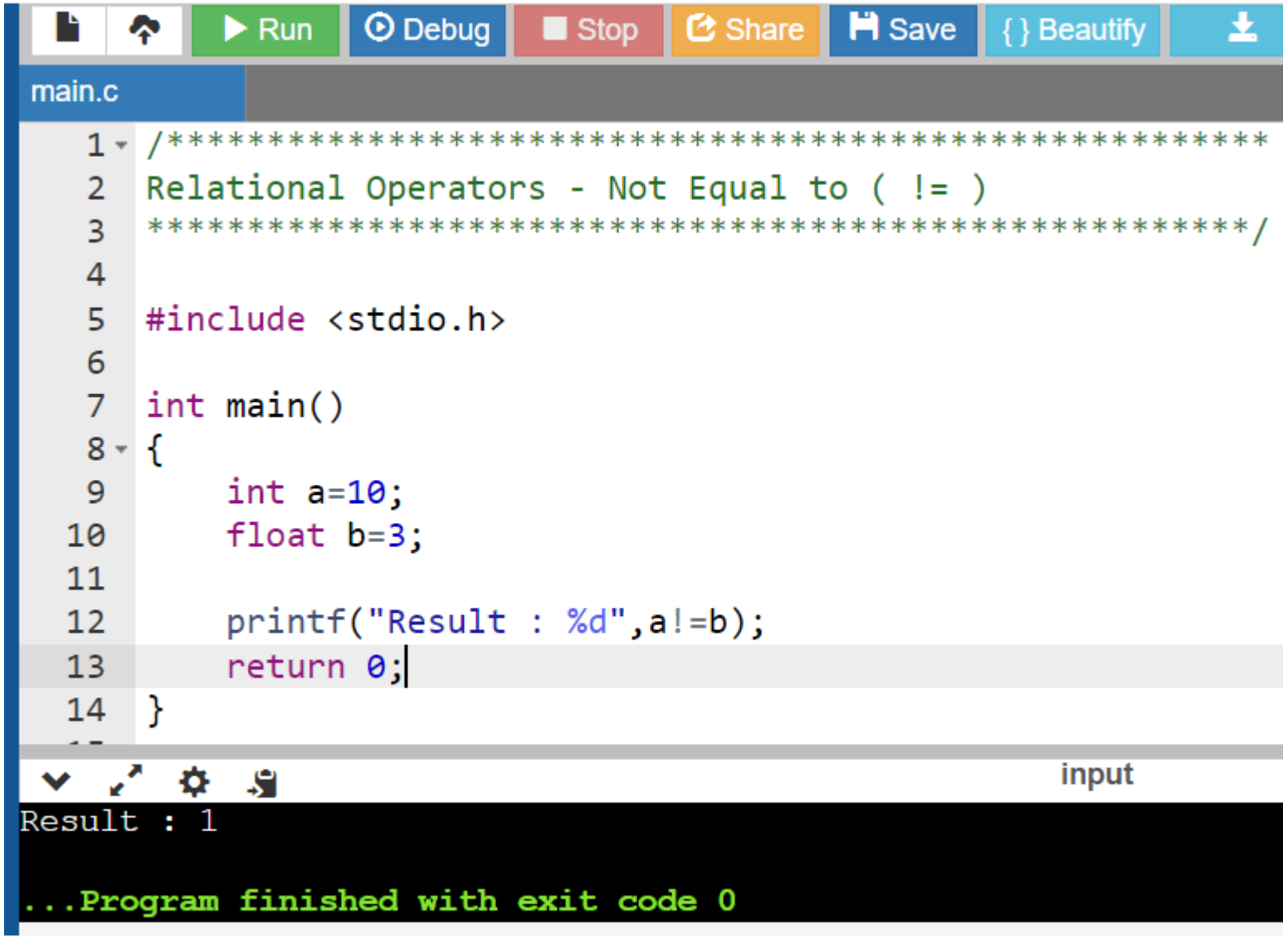
Relational Operators : Returns Boolean value (1-TRUE,0-False)

Greater than or Equal to : >=



Relational Operators : Returns Boolean value (1-TRUE,0-False)

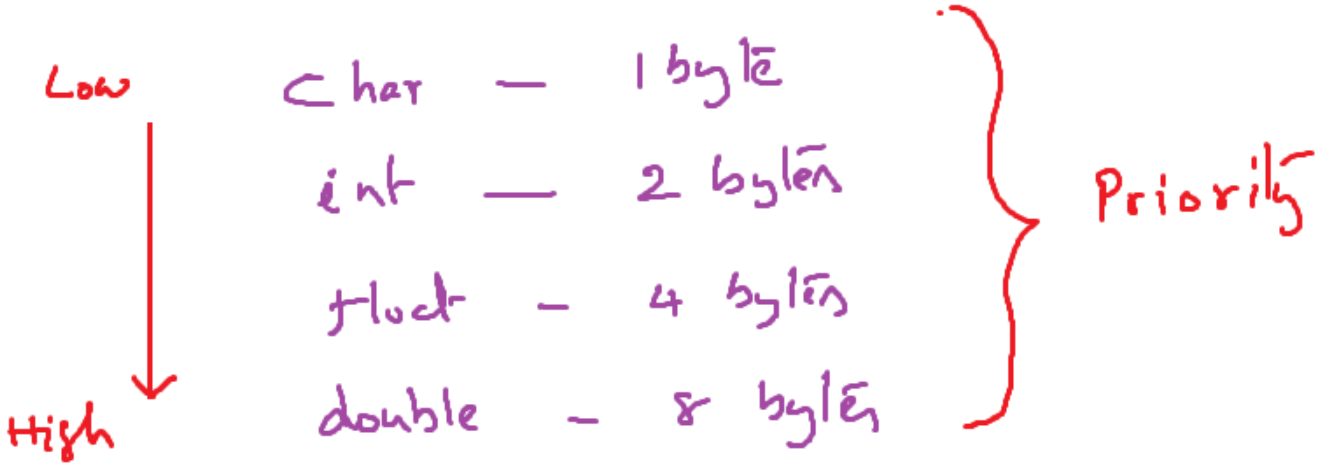
Not Equal to : !=



Assignment Operators : Assign value to the variables or sets an expression

Type Conversion

Converting one data to another



Implicit Type Conversion

This is automatic type conversion done by system

Ex: char ch='a';

int a=10, c;

c = a + ch;

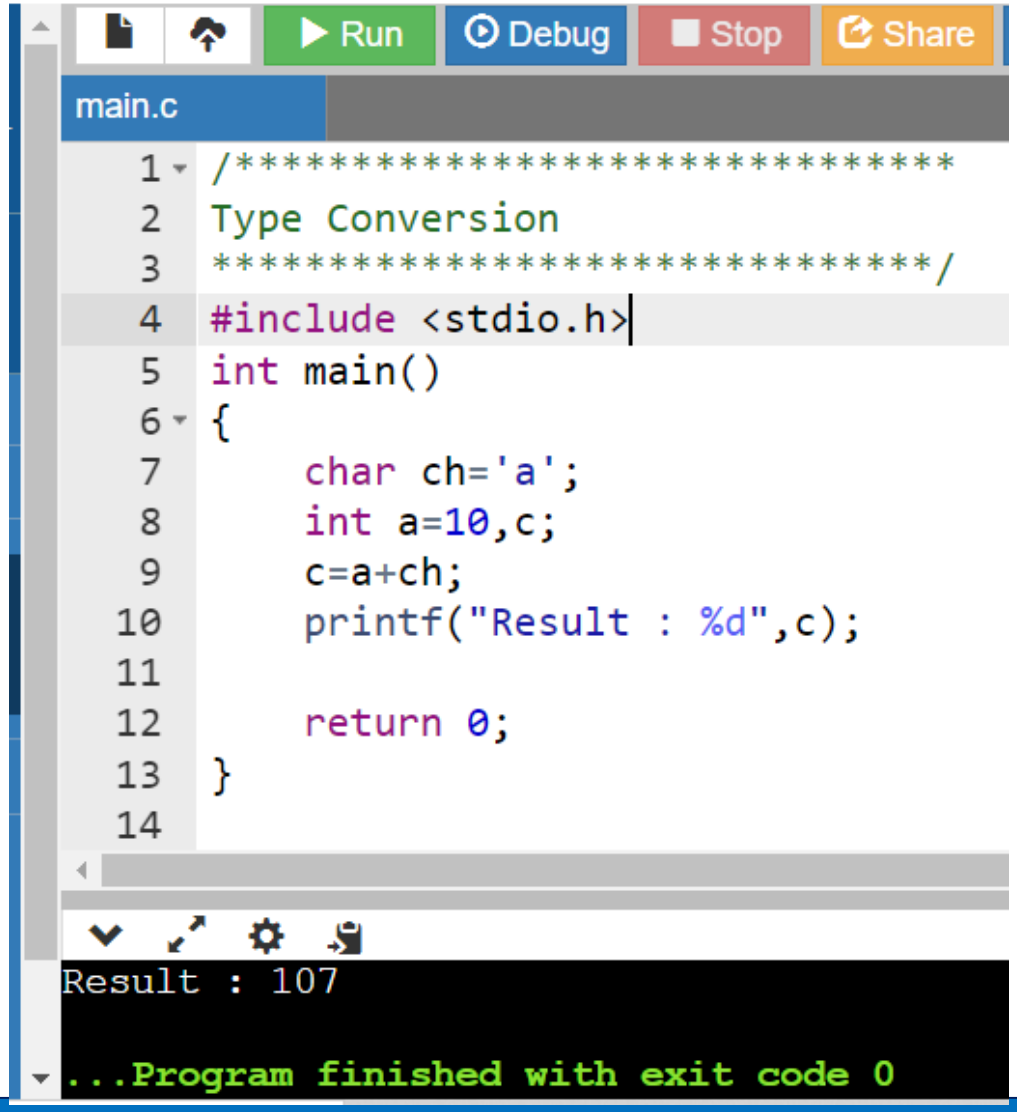
int char → Integer ASCII value of a is 97

10 + 97 → 107

int int int

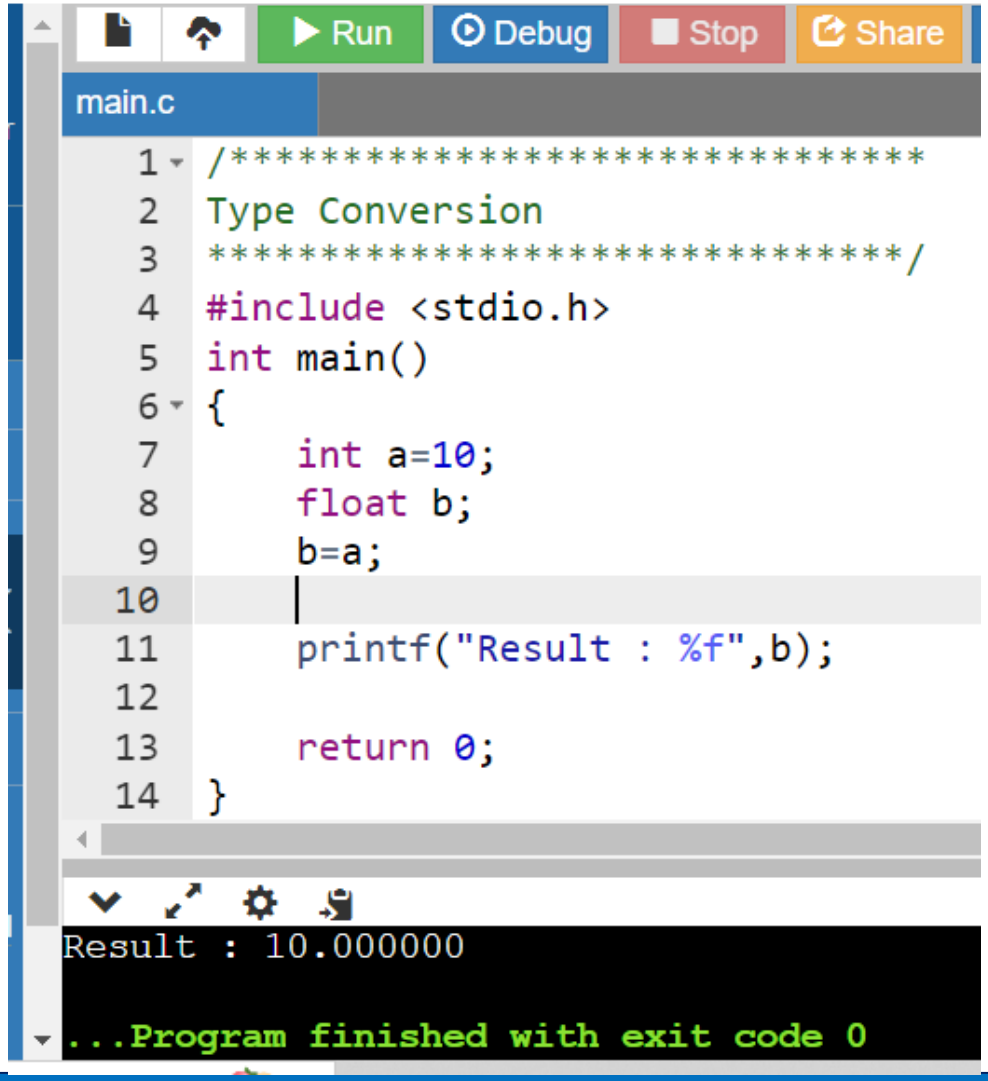
Implicit Type Conversion

This is automatic type conversion done by system



Implicit Type Conversion

Ex 2 : int to float conversion



Explicit Type Conversion

This is done by the user. This is called as type casting.

Syntax :

(datatype) variable ;

Ex: int a ,

float b = 10.5 ;

a = b ; ✗

a = (int) b ✓ 10.5 → 10 (loss of data)

Explicit Type Conversion

This is done by the user. This is called as type casting.

Ex 2 : int a = 10, b = 3;

a/b → 10/3 → 3

(float)a/b → 10.0/3 → 3.3333

Run | Debug | Stop | Share | Save

main.c

```
1  /*********************************
2  Type Conversion
3  *********************************/
4  #include <stdio.h>
5  int main()
6  {
7      int a=10,b=3;
8
9      printf("Result : %f",(float)a/b);
10
11     return 0;
12 }
13
```

input

```
Result : 3.333333

...Program finished with exit code 0
```