A Structured Programming Approach Using C

PRESENTED BY
J.KRISHNA CHANDRIKA
ASSISTANT PROFESSOR
COMPUTER SCIENCE

Introduction to the C Language

Objectives

- ❑ To understand the structure of a C-language program.
- ❑ To write your first C program.
- ❑ To introduce the include preprocessor command.
- ❑ To be able to create good identifiers for objects in a program.
- ❑ To be able to list, describe, and use the C basic data types.
- ❑ To be able to create and use variables and constants.
- ❑ To understand input and output concepts.
- ❑ To be able to use simple input and output statements.

Background

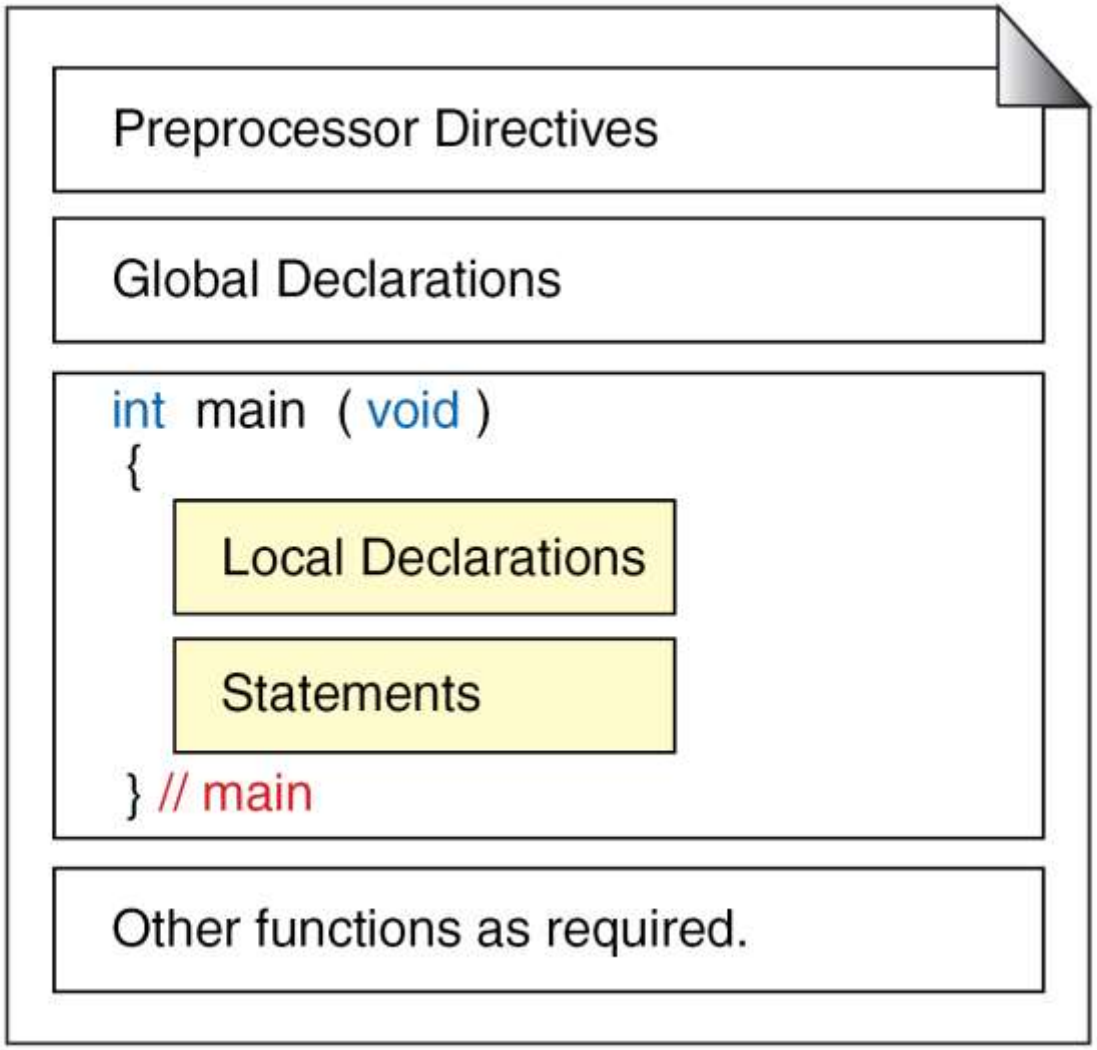
C is a structured programming language. It is considered a high-level language because it allows the programmer to concentrate on the problem at hand and not worry about the machine that the program will be using. That is another reason why it is used by software developers whose applications have to run on many different hardware platforms.

C Programs

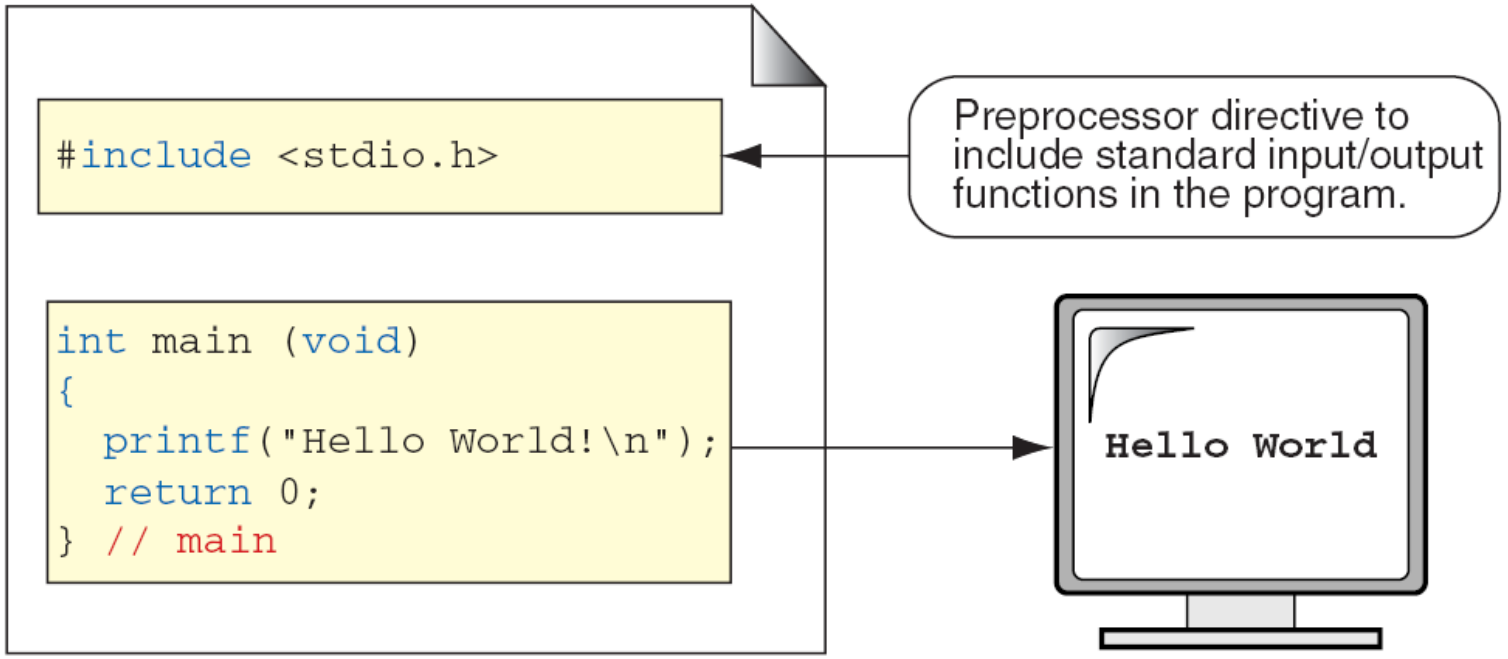
It's time to write your first C program.

Topics discussed in this section:

Structure of a C Program
Your First C Program
Comments
The Greeting Program



Structure of a C Program


#include <stdio.h>
Preprocessor directive to include standard input/output functions in the program.
int main (void)
{
printf("Hello World!\n");
return 0;
} // main
Hello World

The Greeting Program

```
1  /* The greeting program. This program demonstrates
2     some of the components of a simple C program.
3        Written by:  your name here
4        Date:        date program written
5  */
6  #include <stdio.h>
7
8  int main (void)
9  {
10 // Local Declarations
11
12 // Statements
13
14    printf("Hello World!\n");
15
16    return 0;
17 } // main
```

```
/* This is a block comment that
   covers two lines.                    */

/*
** It is a very common style to put the opening token
** on a line by itself, followed by the documentation
** and then the closing token on a separate line. Some
** programmers also like to put asterisks at the beginning
** of each line to clearly mark the comment.
*/
```

Examples of Block Comments

```
// This is a whole line comment

a = 5;              // This is a partial line comment
```

Examples of Line Comments

Identifiers

One feature present in all computer languages is the identifier. Identifiers allow us to name data and other objects in the program. Each identified object in the computer is stored at a unique address.

1. First character must be alphabetic character or underscore.
2. Must consist only of alphabetic characters, digits, or underscores.
3. First 63 characters of an identifier are significant.
4. Cannot duplicate a keyword.

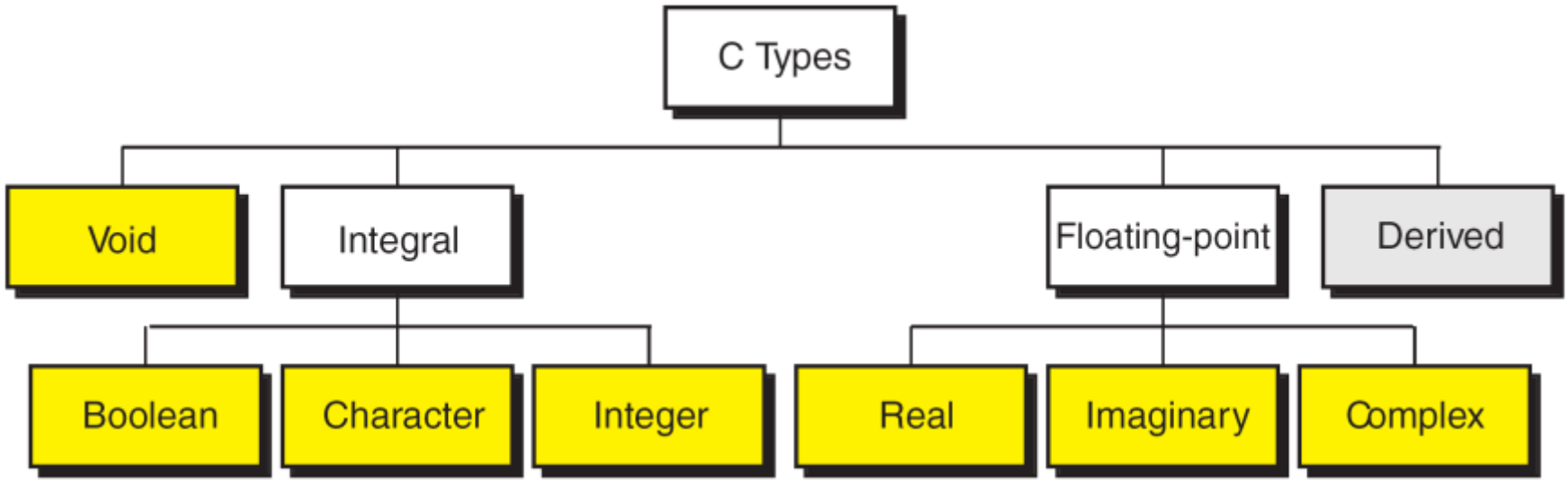
Rules for Identifiers

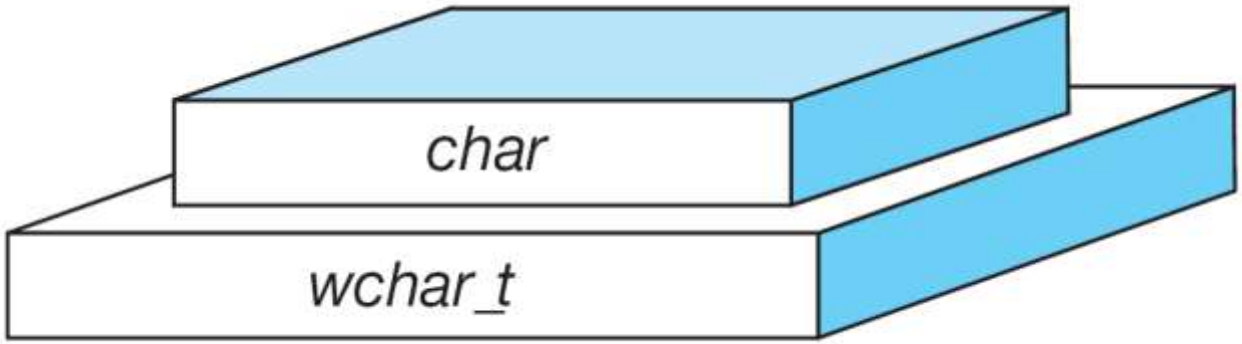
Note

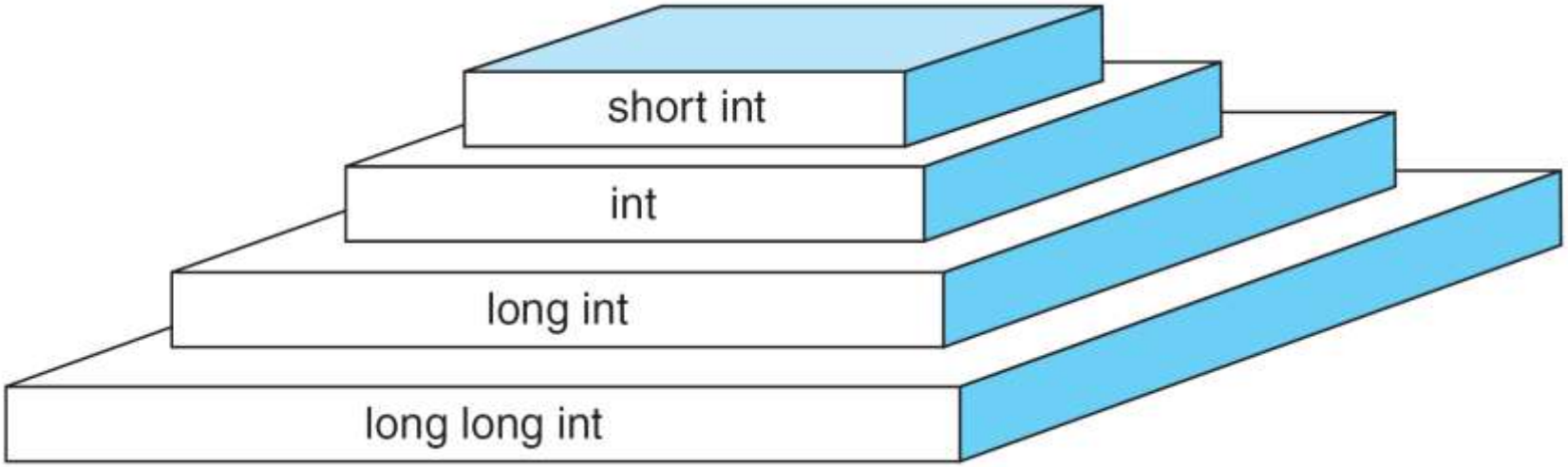
C is a case-sensitive language.

Valid Names		Invalid Name	
a	// Valid but poor style	$sum	// $ is illegal
student_name		2names	// First char digit
_aSystemName		sum-salary	// Contains hyphen
_Bool	// Boolean System id	stdnt Nmbr	// Contains spaces
INT_MIN	// System Defined Value	int	// Keyword

Examples of Valid and Invalid Names


C Types
Void
Integral
Floating-point
Derived
Boolean
Character
Integer
Real
Imaginary
Complex


char
wchar_t



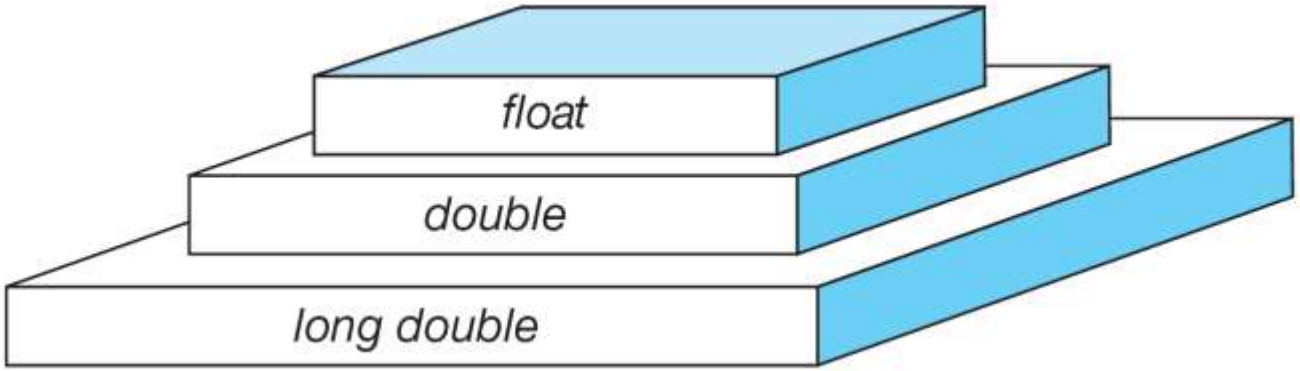
Integer Types

Note

sizeof (short) ≤ sizeof (int) ≤ sizeof (long) ≤ sizeof (long long)

Type	Byte Size	Minimum Value	Maximum Value
short int	2	–32,768	32,767
int	4	–2,147,483,648	2,147,483,647
long int	4	–2,147,483,648	2,147,483,647
long long int	8	–9,223,372,036,854,775,807	9,223,372,036,854,775,806

Typical Integer Sizes and Values for Signed Integers



Floating-point Types

Note

sizeof (float) ≤ sizeof (double) ≤ sizeof (long double)

Category	Type	C Implementation
Void	Void	*void*
Integral	Boolean	*bool*
	Character	*char, wchar_t*
	Integer	*short int, int, long int, long long int*
Floating-Point	Real	*float, double, long double*
	Imaginary	*float imaginary, double imaginary, long double imaginary*
	Complex	*float complex, double complex, long double complex*

Type Summary

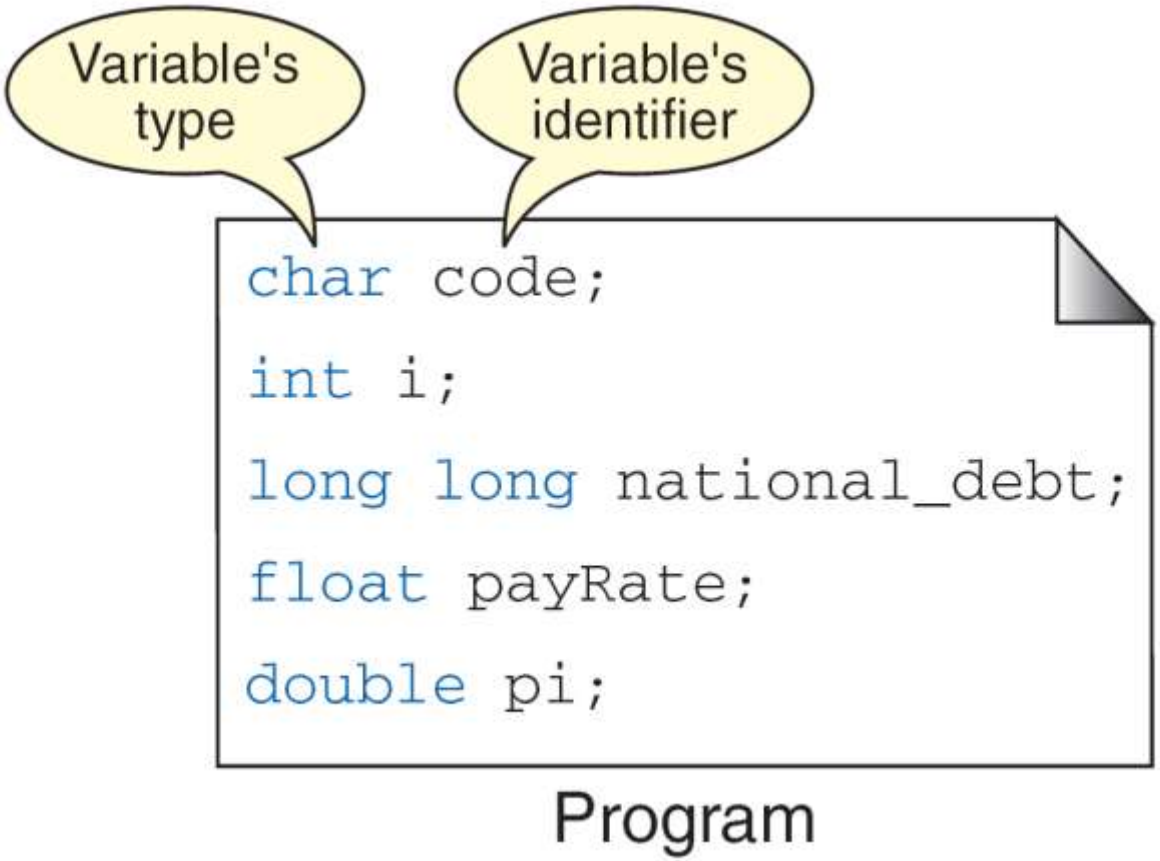
Variables

Variables are named memory locations that have a type, such as integer or character, which is inherited from their type. The type determines the values that a variable may contain and the operations that may be used with its values.

Topics discussed in this section:

Variable Declaration

Variable Initialization

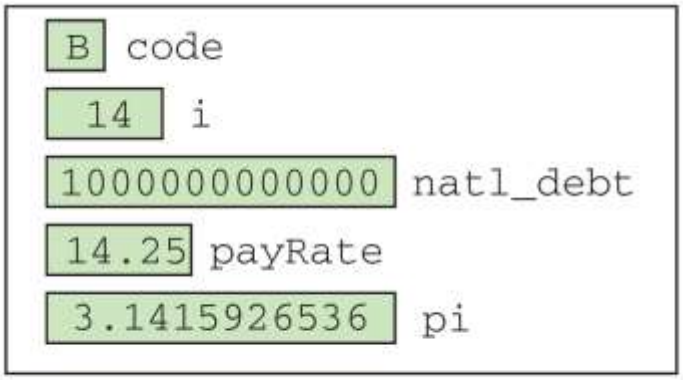

Variable's type
Variable's identifier
char code;
int i;
long long national_debt;
float payRate;
double pi;
Program

```
bool   fact;
short  maxItems;                 // Word separator: Capital
long   long national_debt;       // Word separator: underscore
float  payRate;                  // Word separator: Capital
double tax;
float  complex voltage;
char   code, kind;               // Poor style—see text
int    a, b;                     // Poor style—see text
```

Examples of Variable Declarations and Definitions

```
char code = 'B';
int  i    = 14;
long long natl_debt = 1000000000000;
float     payRate   = 14.25;
double    pi        = 3.1415926536;
```

Program



Memory

Variable Initialization

Constants

Constants are data values that cannot be changed during the execution of a program. Like variables, constants have a type. In this section, we discuss Boolean, character, integer, real, complex, and string constants.

Topics discussed in this section:

Constant Representation
Coding Constants

ASCII Character	Symbolic Name
null character	'\0'
alert (bell)	'\a'
backspace	'\b'
horizontal tab	'\t'
newline	'\n'
vertical tab	'\v'
form feed	'\f'
carriage return	'\r'
single quote	'\''
double quote	'\"'
backslash	'\\'

Symbolic Names for Control Characters

Representation	Value	Type
+123	123	int
–378	–378	int
–32271L	–32,271	long int
76542LU	76,542	unsigned long int
12789845LL	12,789,845	long long int

Examples of Integer Constants

Representation	Value	Type
0.	0.0	double
.0	0.0	double
2.0	2.0	double
3.1416	3.1416	double
–2.0f	–2.0	float
3.1415926536L	3.1415926536	long double

Examples of Real Constants

THANK YOU…..