C STANDARD LIBRARY

collection of built-in functions

tutorialspoint

SIMPLYEASYLEARNING

www.tutorialspoint.com

https://www.facebook.com/tutorialspointindia

https://twitter.com/tutorialspoint

About the Tutorial

C is a general-purpose, procedural, imperative computer programming language developed in 1972 by Dennis M. Ritchie at the Bell Telephone Laboratories to develop the Unix operating system.

C is the most widely used computer language that keeps fluctuating at number one scale of popularity along with Java programming language which is also equally popular and most widely used among modern software programmers.

The C Standard Library is a set of C built-in functions, constants and header files like <assert.h>, <ctype.h>, etc. This library will work as a reference manual for C programmers.

Audience

The C Standard Library is a reference for C programmers to help them in their projects related to system programming. All the C functions have been explained in a user-friendly way and they can be copied and pasted in your C projects.

Prerequisites

A basic understanding of the C Programming language will help you in understanding the C built-in functions covered in this library.

Copyright & Disclaimer

Table of Contents

1. C LIBRARY – <ASSERT.H>

Introduction

The **assert.h** header file of the C Standard Library provides a macro called **assert** which can be used to verify assumptions made by the program and print a diagnostic message if this assumption is false.

The defined macro **assert** refers to another macro **NDEBUG** which is not a part of <assert.h>. If NDEBUG is defined as a macro name in the source file, at the point where <assert.h> is included, the **assert** macro is defined as follows:

```
#define assert(ignore) ((void)0)
```

Library Macros

Following is the only function defined in the header assert.h:

S.N.	Function & Description
1	void assert(int expression) This is actually a macro and not a function, which can be used to add diagnostics in your C program.

void assert(int expression)

Description

The C library macro **void assert(int expression)** allows diagnostic information to be written to the standard error file. In other words, it can be used to add diagnostics in your C program.

Declaration

Following is the declaration for assert() Macro.

```
void assert(int expression);
```

Parameters

- **expression** -- This can be a variable or any C expression. If **expression** evaluates to TRUE, assert() does nothing. If **expression** evaluates to FALSE, assert() displays an error message on **stderr** (standard error stream to display error messages and diagnostics) and aborts program execution.

Return Value

This macro does not return any value.

Example

The following example shows the usage of assert() macro:

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

int main()
{
   int a;
   char str[50];

   printf("Enter an integer value: ");
   scanf("%d\n", &a);
   assert(a >= 10);
   printf("Integer entered is %d\n", a);

   printf("Enter string: ");
   scanf("%s\n", &str);
   assert(str != NULL);
   printf("String entered is: %s\n", str);

   return(0);
}
```

Let us compile and run the above program in the interactive mode as shown below:

```
Enter an integer value: 11
Integer entered is 11
Enter string: tutorialspoint
String entered is: tutorialspoint
```

2. C LIBRARY – <CTYPE.H>

Introduction

The **ctype.h** header file of the C Standard Library declares several functions that are useful for testing and mapping characters.

All the functions accepts **int** as a parameter, whose value must be EOF or representable as an unsigned char.

All the functions return non-zero (true) if the argument c satisfies the condition described, and zero (false) if not.

Library Functions

Following are the functions defined in the header ctype.h:

S.N.	Function & Description
1	int isalnum(int c) This function checks whether the passed character is alphanumeric.
2	int isalpha(int c) This function checks whether the passed character is alphabetic.
3	int iscntrl(int c) This function checks whether the passed character is control character.
4	int isdigit(int c) This function checks whether the passed character is decimal digit.
5	int isgraph(int c) This function checks whether the passed character has graphical representation using locale.
6	int islower(int c)

	This function checks whether the passed character is lowercase letter.
7	int isprint(int c) This function checks whether the passed character is printable.
8	int ispunct(int c) This function checks whether the passed character is a punctuation character.
9	int isspace(int c) This function checks whether the passed character is white-space.
10	int isupper(int c) This function checks whether the passed character is an uppercase letter.
11	int isxdigit(int c) This function checks whether the passed character is a hexadecimal digit.

int isalnum(int c)

Description

The C library function **void isalnum(int c)** checks if the passed character is alphanumeric.

Declaration

Following is the declaration for isalnum() function.

```
int isalnum(int c);
```

Parameters

- **c** -- This is the character to be checked.

Return Value

This function returns non-zero value if c is a digit or a letter, else it returns 0.

Example

The following example shows the usage of isalnum() function.

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

```
int main()
{
   int var1 = 'd';
   int var2 = '2';
   int var3 = '\t';
   int var4 = ' ';

   if( isalnum(var1) )
   {
      printf("var1 = |%c| is alphanumeric\n", var1 );
   }
   else
   {
      printf("var1 = |%c| is not alphanumeric\n", var1 );
   }
   if( isalnum(var2) )
   {
      printf("var2 = |%c| is alphanumeric\n", var2 );
   }
   else
   {
      printf("var2 = |%c| is not alphanumeric\n", var2 );
   }
   if( isalnum(var3) )
   {
      printf("var3 = |%c| is alphanumeric\n", var3 );
   }
   else
   {
      printf("var3 = |%c| is not alphanumeric\n", var3 );
   }
   if( isalnum(var4) )
   {
      printf("var4 = |%c| is alphanumeric\n", var4 );
   }
   else
   {
      printf("var4 = |%c| is not alphanumeric\n", var4 );
   }

   return(0);
}
```

Let us compile and run the above program to produce the following result:

```
var1 = |d| is alphanumeric
var2 = |2| is alphanumeric
var3 = | | is not alphanumeric
var4 = | | is not alphanumeric
```

int isalpha(int c)

Description

The C library function **void isalpha(int c)** checks if the passed character is alphabetic.

Declaration

Following is the declaration for isalpha() function.

```
int isalpha(int c);
```

Parameters

- **c** -- This is the character to be checked.

Return Value

This function returns non-zero value if c is an alphabet, else it returns 0.

Example

The following example shows the usage of isalpha() function.

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

```
int main()
{
   int var1 = 'd';
   int var2 = '2';
   int var3 = '\t';
   int var4 = ' ';

   if( isalpha(var1) )
   {
      printf("var1 = |%c| is an alphabet\n", var1 );
   }
   else
   {
      printf("var1 = |%c| is not an alphabet\n", var1 );
   }
   if( isalpha(var2) )
   {
      printf("var2 = |%c| is an alphabet\n", var2 );
   }
   else
   {
      printf("var2 = |%c| is not an alphabet\n", var2 );
   }
   if( isalpha(var3) )
   {
      printf("var3 = |%c| is an alphabet\n", var3 );
   }
   else
   {
      printf("var3 = |%c| is not an alphabet\n", var3 );
   }
   if( isalpha(var4) )
   {
      printf("var4 = |%c| is an alphabet\n", var4 );
   }
   else
   {
      printf("var4 = |%c| is not an alphabet\n", var4 );
   }

   return(0);
}
```

Let us compile and run the above program to produce the following result:

```
var1 = |d| is an alphabet
var2 = |2| is not an alphabet
var3 = | | is not an alphabet
var4 = | | is not an alphabet
```

int iscntrl(int c)

Description

The C library function **void iscntrl(int c)** checks if the passed character is a control character.

According to standard ASCII character set, control characters are between ASCII codes 0x00 (NUL), 0x1f (US), and 0x7f (DEL). Specific compiler implementations for certain platforms may define additional control characters in the extended character set (above 0x7f).

Declaration

Following is the declaration for iscntrl() function.

```
int iscntrl(int c);
```

Parameters

- **c** -- This is the character to be checked.

Return Value

This function returns non-zero value if c is a control character, else it returns 0.

Example

The following example shows the usage of iscntrl() function.

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

```
#include <ctype.h>

int main ()
{
   int i = 0, j = 0;
   char str1[] = "all \a about \t programming";
   char str2[] = "tutorials \n point";

   /* Prints string till control character \a */
   while( !iscntrl(str1[i]) )
   {
      putchar(str1[i]);
      i++;
   }

   /* Prints string till control character \n */
   while( !iscntrl(str2[j]) )
   {
      putchar(str2[j]);
      j++;
   }
   return(0);
}
```

Let us compile and run the above program to produce the following result:

```
all tutorials
```

int isdigit(int c)

Description

The C library function **void isdigit(int c)** checks if the passed character is a decimal digit character.

Decimal digits are (numbers): 0 1 2 3 4 5 6 7 8 9.

Declaration

Following is the declaration for isdigit() function.

```
int isdigit(int c);
```

Parameters

- **c** -- This is the character to be checked.

Return Value

This function returns non-zero value if c is a digit, else it returns 0.

Example

The following example shows the usage of isdigit() function.

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

int main()
{
   int var1 = 'h';
   int var2 = '2';

   if( isdigit(var1) )
   {
      printf("var1 = |%c| is a digit\n", var1 );
   }
   else
   {
      printf("var1 = |%c| is not a digit\n", var1 );
   }
   if( isdigit(var2) )
   {
      printf("var2 = |%c| is a digit\n", var2 );
   }
   else
   {
      printf("var2 = |%c| is not a digit\n", var2 );
   }


   return(0);
}
```

Let us compile and run the above program to produce the following result:

```
var1 = |h| is not a digit
var2 = |2| is a digit
```

int isgraph(int c)

Description

The C library function **void isgraph(int c)** checks if the character has graphical representation.

The characters with graphical representations are all those characters that can be printed except for whitespace characters (like ' '), which is not considered as **isgraph** characters.

Declaration

Following is the declaration for isgraph() function.

```
int isgraph(int c);
```

Parameters

- **c** -- This is the character to be checked.

Return Value

This function returns non-zero value if c has a graphical representation as character, else it returns 0.

Example

The following example shows the usage of isgraph() function.

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

int main()
{
   int var1 = '3';
   int var2 = 'm';
   int var3 = ' ';

   if( isgraph(var1) )
   {
       printf("var1 = |%c| can be printed\n", var1 );
   }
   else
   {
      printf("var1 = |%c| can't be printed\n", var1 );
   }
   if( isgraph(var2) )
   {
       printf("var2 = |%c| can be printed\n", var2 );
   }
   else
   {
      printf("var2 = |%c| can't be printed\n", var2 );
   }
   if( isgraph(var3) )
   {
       printf("var3 = |%c| can be printed\n", var3 );
   }
   else
   {
      printf("var3 = |%c| can't be printed\n", var3 );
   }
   return(0);
}
```

Let us compile and run the above program to produce the following result:

```
var1 = |3| can be printed
var2 = |m| can be printed
var3 = | | can't be printed
```

int islower(int c)

Description

The C library function **int islower(int c)** checks whether the passed character is a lowercase letter.

Declaration

Following is the declaration for islower() function.

```
int islower(int c);
```

Parameters

- **c** -- This is the character to be checked.

Return Value

This function returns a non-zero value(true) if c is a lowercase alphabetic letter else, zero (false).

Example

The following example shows the usage of islower() function.

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

int main()
{
   int var1 = 'Q';
   int var2 = 'q';
   int var3 = '3';

   if( islower(var1) )
   {
       printf("var1 = |%c| is lowercase character\n", var1 );
   }
   else
   {
      printf("var1 = |%c| is not lowercase character\n", var1 );
   }
   if( islower(var2) )
   {
       printf("var2 = |%c| is lowercase character\n", var2 );
   }
   else
   {
      printf("var2 = |%c| is not lowercase character\n", var2 );
   }
   if( islower(var3) )
   {
       printf("var3 = |%c| is lowercase character\n", var3 );
   }
   else
   {
      printf("var3 = |%c| is not lowercase character\n", var3 );
   }

   return(0);
}
```

Let us compile and run the above program to produce the following result:

```
var1 = |Q| is not lowercase character
var2 = |q| is lowercase character
var3 = |3| is not lowercase character
```

int isprint(int c)

Description

The C library function **int isprint(int c)** checks whether the passed character is printable. A printable character is a character that is not a control character.

Declaration

Following is the declaration for isprint() function.

```
int isprint(int c);
```

Parameters

- **c** -- This is the character to be checked.

Return Value

This function returns a non-zero value(true) if c is a printable character else, zero (false).

Example

The following example shows the usage of isprint() function.

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

int main()
{
   int var1 = 'k';
   int var2 = '8';
   int var3 = '\t';
   int var4 = ' ';

   if( isprint(var1) )
   {
      printf("var1 = |%c| can be printed\n", var1 );
   }
   else
   {
      printf("var1 = |%c| can't be printed\n", var1 );
   }
   if( isprint(var2) )
   {
      printf("var2 = |%c| can be printed\n", var2 );
   }
   else
   {
      printf("var2 = |%c| can't be printed\n", var2 );
   }
   if( isprint(var3) )
   {
      printf("var3 = |%c| can be printed\n", var3 );
   }
   else
   {
      printf("var3 = |%c| can't be printed\n", var3 );
   }
   if( isprint(var4) )
   {
      printf("var4 = |%c| can be printed\n", var4 );
   }
   else
   {
      printf("var4 = |%c| can't be printed\n", var4 );
   }

   return(0);
}
```

Let us compile and run the above program to produce the following result:

```
var1 = |k| can be printed
var2 = |8| can be printed
var3 = |	| can't be printed
var4 = | | can be printed
```

int ispunct(int c)

Description

The C library function **int ispunct(int c)** checks whether the passed character is a punctuation character. A punctuation character is any graphic character (as in isgraph) that is not alphanumeric (as in isalnum).

Declaration

Following is the declaration for ispunct() function.

```
int ispunct(int c);
```

Parameters

- **c** -- This is the character to be checked.

Return Value

This function returns a non-zero value(true) if c is a punctuation character else, zero (false).

Example

The following example shows the usage of ispunct() function.

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

int main()
{
   int var1 = 't';
   int var2 = '1';
   int var3 = '/';
   int var4 = ' ';

   if( ispunct(var1) )
   {
       printf("var1 = |%c| is a punctuation character\n", var1 );
   }
   else
   {
       printf("var1 = |%c| is not a punctuation character\n", var1 );
   }
   if( ispunct(var2) )
   {
       printf("var2 = |%c| is a punctuation character\n", var2 );
   }
   else
   {
       printf("var2 = |%c| is not a punctuation character\n", var2 );
   }
   if( ispunct(var3) )
   {
       printf("var3 = |%c| is a punctuation character\n", var3 );
   }
   else
   {
       printf("var3 = |%c| is not a punctuation character\n", var3 );
   }
   if( ispunct(var4) )
   {
       printf("var4 = |%c| is a punctuation character\n", var4 );
   }
   else
   {
       printf("var4 = |%c| is not a punctuation character\n", var4 );
   }

   return(0);
}
```

Let us compile and run the above program that will produce the following result:

```
var1 = |t| is not a punctuation character
var2 = |1| is not a punctuation character
var3 = |/| is a punctuation character
var4 = | | is not a punctuation character
```

int isspace(int c)

Description

The C library function **int isspace(int c)** checks whether the passed character is white-space.

Standard white-space characters are:

```
' '     (0x20)    space (SPC)
'\t' (0x09) horizontal tab (TAB)
'\n' (0x0a) newline (LF)
'\v' (0x0b) vertical tab (VT)
'\f' (0x0c) feed (FF)
```

```
'\r'  (0x0d) carriage return (CR)
```

Declaration

Following is the declaration for isspace() function.

```
int isspace(int c);
```

Parameters

- **c** -- This is the character to be checked.

Return Value

This function returns a non-zero value(true) if c is a white-space character else, zero (false).