

Constants and Format Specifiers in C

Dr Bhanu

C Constants

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

Integer Constants

- An **integer** constant must have at least one digit.
- It must not have a decimal point.
- It can either be positive or negative.
- No commas or blanks are allowed within an **integer** constant.
- If no sign precedes an **integer** constant, it is assumed to be positive.
- The allowable range for integer constants depends on their “type” declaration.
 - short int, int, long int, long long int
 - For example, if “short int” is of 2 byte length, the allowable range of “*unsigned*” short int will be 0 to 65535 and that of “*signed*” short int will be -32768 to 32767.

Example:

0

-33

32767

Real Constants

- A **real** constant must have at least one digit.
- It **must** have a decimal point.
- It can either be positive or negative.
- No commas or blanks are allowed within a **real** constant.
- If no sign precedes a **real** constant, it is assumed to be positive.
- The range depends on the number of bytes allowed by its type.
- Exponential form of representation is used if the value of constant is too small or too large.
 - -7.2e-24, 1.38e45 -- mantissa and exponent

Character Constants

- A **character** constant must have at least one alphabet or a single digit or a special symbol, enclosed within single quotes.
- The maximum length of a character constant is one byte only
- Character constants are represented by their corresponding ASCII code
 - The first 32 characters in the ASCII table are non printable characters.

Ex : 'A'
 'v'
 '9'
 '@'

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	' \\ '

A character constant is enclosed in single quotes.

Symbolic Names for Control Characters

Examples

- `const int d = 567;`
- `const float pi = 3.14;`
- `const char ch1 = 'T';`

A variable that is declared as constant can't be changed during the entire program.

Conversion Specification Examples

%	Flag	Minimum Width	Precision	Size	Code
---	------	---------------	-----------	------	------

- The format contains a **start token** (%), four optional modifiers and a **conversion code**
- Size modifier – four sizes - h, l, ll, L
 - Short, long, long long and Long double
- Width modifier – Specifies the minimum number of positions in the output
 - Useful to align output in columns
 - When not specified, it will take enough space to accommodate data

Conversion Specification Examples

%	Flag	Minimum Width	Precision	Size	Code
---	------	---------------	-----------	------	------

- Precision Modifier - used for floating point numbers
 - Specifies the number of decimal places to be printed for the fraction
 - When not specified, printf will print six decimal places
 - When both width and precision are used, the width must be large enough to contain the integral value, decimal point and the number of digits in the fraction.
- Example : %8.5f
 - Total width = 8, Digits in fraction = 4

Conversion Specification Examples

%	Flag	Minimum Width	Precision	Size	Code
---	------	---------------	-----------	------	------

- Example : %8.5f
 - Total width = 8, Digits in fraction = 4

Float a = 23.5806567

	2	3	.	5	8	0	7
--	---	---	---	---	---	---	---

Conversion Specification Examples

%	Flag	Minimum Width	Precision	Size	Code
---	------	---------------	-----------	------	------

- Example : %8.4f
 - Total width = 8, Digits in fraction = 4

Float a = 23.5806567

	2	3	.	5	8	0	7
--	---	---	---	---	---	---	---

Float a = 23.5806567

2	3	.	5	8	0	6	5	6	7
---	---	---	---	---	---	---	---	---	---

What should be the format? : % ?..?f

Conversion Specification Examples

```
printf("%d%c%f \n\n", 23, 'H', 7.9);
```

23H7.900000

```
printf("%d %c %f \n\n", 23, 'H', 7.9);
```

23 H 7.900000

```
int a = 968;
```

```
char b = 'K';
```

```
float c = 5.876;
```

```
printf("%d %c %f \n\n", a, b, c);
```

968 K 5.876000

Conversion Specification Examples

```
int a = 968;  
char b = 'K';  
float c = 5.876;  
printf("%d\t%c\t%f \n\n", a, b, c);
```

```
968    K    5.876000
```

Conversion Specification Examples

```
printf("\t| %5d \n", 23);  
printf("\t| %5d \n", 2345);  
printf("\t| %5d \n", 23456);  
printf("\t| %5d \n", 234);
```

```
| 23  
| 2345  
| 23456  
| 234
```

```
printf("\t| %-5d \n", 23);  
printf("\t| %-5d \n", 2345);  
printf("\t| %-5d \n", 23456);  
printf("\t| %-5d \n", 234);
```

```
| 23  
| 2345  
| 23456  
| 234
```

Conversion Specification Examples

```
printf("\t| %05d \n", 23);  
printf("\t| %05d \n", 2345);  
printf("\t| %05d \n", 23456);  
printf("\t| %05d \n", 234);
```

```
| 00023  
| 02345  
| 23456  
| 00234
```

```
printf("\t| %+7d \n", -23);  
printf("\t| %+7d \n", 2345);  
printf("\t| %+7d \n", 23456);  
printf("\t| %+7d \n", -234);
```

```
|    -23  
|  +2345  
|+23456  
|    -234
```

Conversion Specification Examples

```
printf("\t| %05d \n", 23);  
printf("\t| %05d \n", 2345);  
printf("\t| %05d \n", 23456);  
printf("\t| %05d \n", 234);
```

```
| 00023  
| 02345  
| 23456  
| 00234
```

```
printf("\t| %+7d \n", -23);  
printf("\t| %+7d \n", 2345);  
printf("\t| %+7d \n", 23456);  
printf("\t| %+7d \n", -234);
```

```
|    -23  
|  +2345  
|+23456  
|    -234
```


Format Specifiers in C

Dr Bhanu

Format Specifiers in C

- Format specifiers in C are used for input and output purposes.
- The type of data that is stored in a [variable](#) using scanf / printf statements is decided by format specifiers

Format Specifier	Description
%d	Integer Format Specifier
%f	Float Format Specifier
%c	Character Format Specifier
%s	String Format Specifier
%u	Unsigned Integer Format Specifier
%ld	Long Int Format Specifier
%x	Hexadecimal format specifier
%o	Octal format specifier
%e, %g	Scientific notation of floats
%hi	Signed integer (short)
%hu	Unsigned Integer (short)

Format Specifiers in C

- A number mentioned after % symbol specifies the minimum field width.
- A period (.) is used to separate field width and precision.

Format Specifier	Description
%i	Integer Format Specifier
%p	Pointer Format Specifier
%l, %ld, %li	long Format Specifier
%lf	double Format Specifier
%Lf	Long double Format Specifier
%lu	Unsigned Long Format Specifier
%lli, %lld	Long long format specifier
%llu	Long long unsigned format specifier
%%	Prints % character

Format Specifiers in C

- There is no specific difference between the `%i` and `%d` format specifiers.
- But both format specifiers behave differently with `scanf` function.
- The `%d` format specifier takes the integer number as decimal
- The `%i` format specifier takes the integer number as decimal, hexadecimal or octal type.
 - You must put '0x' for hexadecimal number and '0' for octal number while entering the input number.

//1. Format specifier (character): %c

```
char data = 'A';  
printf("%c\n", data);
```

```
int data1 = 65;  
printf("%c\n", data1);*/
```

/*2. Format specifiers %d, %i, %u

```
int data = 65;  
printf("%d\n", data);  
printf("%u\n", data);  
printf("%i\n", data);*/
```

Format Specifiers (%x, %o) in C

```
/*int data1, data2, data3;
 printf("Enter value in decimal format:");
 scanf("%d",&data1);
 printf("data1 = %i\n\n", data1);
 printf("Enter value in hexadecimal format (begin with 0x.):");
 scanf("%i",&data2);
 printf("data2 = %i\n\n", data2);
 printf("Enter value in octal format (begin with zero):");
 scanf("%i",&data3);
 printf("data3 = %i\n\n", data3);*/
```

// Format specifiers (octal number): %o

```
/*int data = 65;
 printf("%o\n", data); */
```

```
// Format specifier (Hexadecimal number): %x, %X
/*int data = 11;
 printf("%x\n", data);*/
```

Format Specifiers (%f) in C

```
/* Use of special elements with %f
float data = 6.276240;
printf("%f\n", data);
printf("%0.2f\n", data);
printf("%0.4f\n", data);*/
```

```
/*double data1 = 123456.0;
printf("%e\n", data1);
printf("%f\n", data1);
printf("%g\n", data1);
printf("\n");
```

```
double data2 = 1234567.0;
printf("%e\n", data2);
printf("%f\n", data2);
printf("%g\n", data2);*/
```

- `//char data = 'A';`
- `/*char data;`
-
- `printf(" Enter any character ");`
- `scanf("%c", &data);*/`
-
- `int data;`
-
- `printf(" Enter any integer between 32 and 127 ----->: ");`
- `scanf("%d", &data);`
-
- `printf("%c\n", data);`
- `printf("%d\n", data);`
- `printf("%x\n", data);`
- `printf("%o\n", data);`
- `printf("data2 = %i\n\n", data3);*/`

String Format Specifier (%s) in C

//6. Format specifier (character array or string): %s

```
//char stmt[] = "My name is Gurudeep Singh";  
//printf("%s\n", stmt);
```

```
//printing individual elements using %c  
char stmt[] = " My name is Gurudeep Singh ";  
//printf("%c\n", stmt[12]);
```