Format Specifiers in C

Specifier	Used For
%x	a hexadecimal (base 16) integer
%p	an address (or pointer)
%f	a floating point number for floats
%u	int unsigned decimal

Format Specifiers in C - freeCodeCamp

www.freecodecamp.org › news › fo...

```c
14    //strncpy(string1, string2, 2);                // copy n characters of string2 to string1

15

16    //strset(string1, '?');      //sets all characters of a string to a given character
17    //strnset(string1, 'x', 1); //sets first n characters of a string to a given character
18    //strrev(string1);           //reverses a string

19

20    //int result = strlen(string1);               // returns string length as int
21    //int result = strcmp(string1, string2);      // string compare all characters
22    //int result = strncmp(string1, string2, 1);  // string compare n characters
23    int result = strcmpi(string1, string1);    // string compare all (ignore case)
24    int result = strnicmp(string1, string1, 1);  // string compare n characters (ignore case)

25

26    printf("%d", result);

27

28    /*
29    if(result == 0)
30    {
```

```
PS C:\Users\Cakow\Desktop\C Files> cd "c:\Users\Cakow\Desktop\C Files\" ; if ($?) { gcc HelloWorl
d.c -o HelloWorld } ; if ($?) { .\HelloWorld }
These strings are the same
PS C:\Users\Cakow\Desktop\C Files>
```

File   Edit   Selection   View   Go   Run   Terminal   Help

C HelloWorld.c 1 ×

C HelloWorld.c > ⊗ main()

```c
 1  #include <stdio.h>
 2  #include <string.h>
 3
 4  int main(){
 5
 6      char string1[] = "Bro";
 7      char string2[] = "Code";
 8
 9      strlwr(string1);                                // converts a string to lowercase
10      //strupr(string1);                             // converts a string to uppercase
11      //strcat(string1, string2);                    // appends string2 to end of string1
12      //strncat(string1, string2, 1);                // appends n characters from string2 to str
13      //strcpy(string1, string2);                    // copy string2 to string1
14      //strncpy(string1, string2, 2);                // copy n characters of string2 to string1
15
16      //strset(string1, '?');      //sets all characters of a string to a given character
17      //strnset(string1, 'x', 1); //sets first n characters of a string to a given character
```

PROBLEMS 1   OUTPUT   TERMINAL   DEBUG CONSOLE

```
PS C:\Users\Cakow\Desktop\C Files> cd "c:\Users\Cakow\Desktop\C Files\" ; if ($?) { gcc HelloWorl
d.c -o HelloWorld } ; if ($?) { .\HelloWorld }
These strings are the same
PS C:\Users\Cakow\Desktop\C Files>
```

powershell

Code

⊗ 1 ⚠ 0   Ln 7, Col 28   Spaces: 3   UTF-8   CRLF   C   Win32

C HelloWorld.c ✕

C HelloWorld.c > ⬡ main()

```c
    }

 7

 8  int main()
 9  {
10      // pointer = a "variable-like" reference that holds a memory address to another variable, a
11      //          some tasks are performed more easily with pointers
12      //          * = indirection operator (value at address)
13
14      int age = 21;
15      int *pAge = NULL; //good practice to assign NULL if declaring a pointer
16      pAge = &age;
17
18      //printf("address of age: %p\n", &age);
19      //printf("value of pAge: %p\n", pAge);
```

PROBLEMS   OUTPUT   TERMINAL   DEBUG CONSOLE

```c
5      // pointer = a "variable-like" reference that holds a memory address to another va
6      //              some tasks are performed more easily with pointers
7      //              * = indirection operator (value at address)
8
9      int age = 21;
10     int *pAge = &age;
11
12     printf("address of age: %p\n", &age);
13     printf("value of pAge: %p\n", pAge);
14
15     printf("value of age: %d\n", age);
16     printf("value at stored address: %d\n", *pAge); //dereferencing
17
18     return 0;
```

PROBLEMS   OUTPUT   **TERMINAL**   DEBUG CONSOLE

```
value of pAge: 000000000061FE14
PS C:\Users\Cakow\Desktop\C Files> cd "c:\Users\Cakow\Desktop\C Files\" ; if ($?) { gcc Hell
c -o HelloWorld } ; if ($?) { .\HelloWorld }
address of age: 000000000061FE14
value of pAge: 000000000061FE14
value of age: 21
value at stored address: 21
PS C:\Users
```

so we have 21 stored within our edge
and after dereferencing this pointer

C HelloWorld.c ✕

C HelloWorld.c > ⬡ main()

```c
1   #include <stdio.h>
2
3   int main()
4   {
5
6
7
8
9
10
11
12
```

Advantages of Using Pointers

- Less time in program execution.

- Working on the original variable.

- With the help of pointers, we can create data structures (linked-list, stack, queue).

- Returning more than one values from functions.

- Searching and sorting large data very easily.

- Dynamically memory allocation.

PROBLEMS

Window

Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\Cakow\Desktop\C Files>