To:
Tenouk

C LAB WORKSHEET 6_1
C scanf(), scanf_s() Examples & Exercises 2

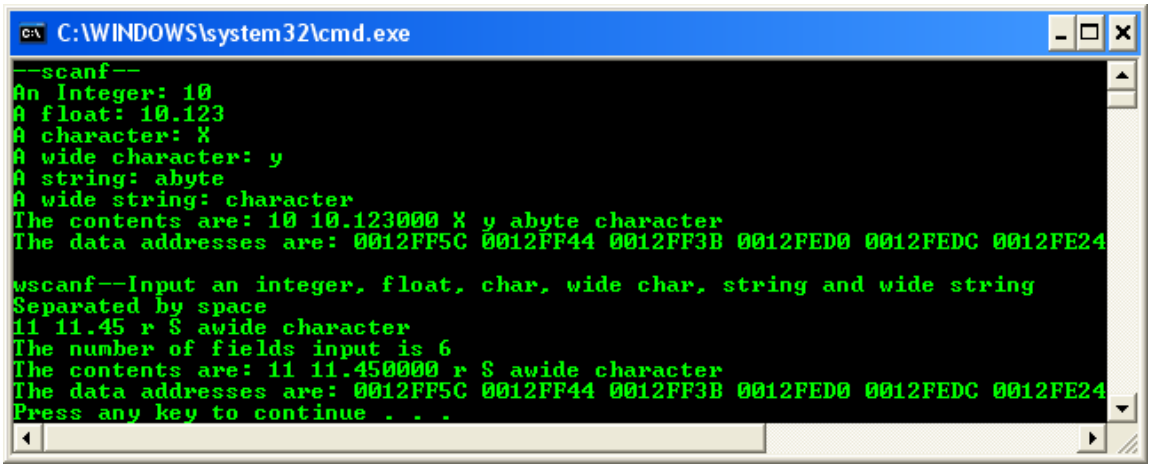
Items in this page:

1. The scanf() and scanf_s() functions examples, questions and activities.
2. Tutorial references are: C/C++ intro & brief history, C/C++ data type 1, C/C++ data type 2, C/C++ data type 3 and C/C++ statement, expression & operator 1, C/C++ statement, expression & operator 2 and C/C++ statement, expression & operator 2. More scanf() and its family examples can be found in C formatted input/output.

- Study the following program source codes and the outputs.

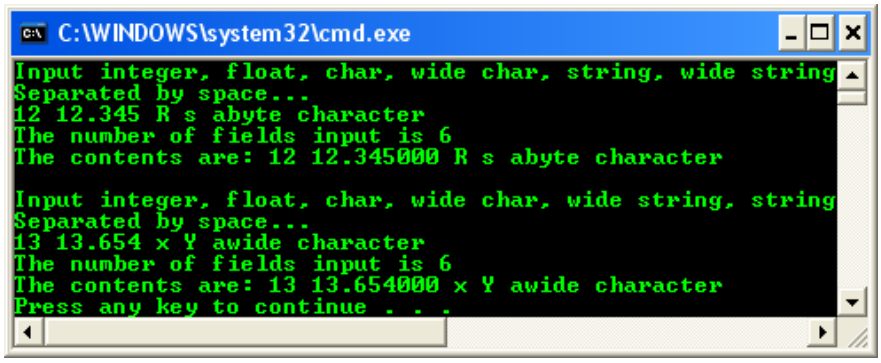
Program Example:

```
/* This program uses the scanf() and wscanf() functions
 * to read formatted input.*/
#include <stdio.h>

int main(void)
{
  int   Myi, TheResult;
  float Fpnt;
  char  Mychar, MyStr[81];
  wchar_t MyWideChar, MyWideStr[81];
  printf("--scanf--\n");
  printf("An Integer: ");
  scanf("%d", &Myi);
  printf("A float: ");
  scanf("%f", &Fpnt);
  printf("A character: ");
  // don't forget a space before %c for the following code (for C and c)
  scanf(" %c", &Mychar);
  printf("A wide character: ");
  scanf(" %C", &MyWideChar);
  printf("A string: ");
  scanf("%80s", MyStr);
  printf("A wide string: ");
  // notice the space before the %, a bug or what?????
  scanf(" %80S", MyWideStr);
  // scanf() and wscanf() are deprecated; consider using scanf_s() and wscanf_s()
  // expect  C4996 warnings...
  printf("The contents are: %d %f %c %C %s %S\n", Myi, Fpnt, Mychar, MyWideChar,
MyStr, MyWideStr);
  printf("The data addresses are: %p %p %p %p %p %p\n", &Myi, &Fpnt, &Mychar,
&MyWideChar, MyStr, MyWideStr);
  printf("\nwscanf--Input an integer, float, char, wide char, string and wide string\n");
  printf("Separated by space\n");
  TheResult = wscanf(L" %d %f %hc %lc %80S %80ls", &Myi, &Fpnt, &Mychar, &MyWideChar,
MyStr, MyWideStr);
  wprintf( L"The number of fields input is %d\n", TheResult);
  // L modifier for Unicode
  wprintf( L"The contents are: %d %f %C %c %hs %s\n", Myi, Fpnt, Mychar, MyWideChar,
MyStr, MyWideStr);
  wprintf( L"The data addresses are: %p %p %p %p %p %p\n", &Myi, &Fpnt, &Mychar,
&MyWideChar, MyStr, MyWideStr);
  return 0;
}
```



Example – secure version

```
// this program uses the scanf_s() and wscanf_s() functions
// secure versions, to read formatted input.
#include <stdio.h>

int main(void)
{
   int      i, result;
   float    fp;
   char     c, s[61];
   wchar_t  wc, ws[61];
   printf("Input integer, float, char, wide char, string, wide string\nSeparated by space...\n");
   result = scanf_s( "%d %f %c %C %s %S", &i, &fp, &c, 1, &wc, 1, s, 60, ws, 60 );
   printf( "The number of fields input is %d\n", result );
   printf( "The contents are: %d %f %c %C %s %S\n", i, fp, c, wc, s, ws);
   printf("\nInput integer, float, char, wide char, wide string, string\nSeparated by space...\n");
   result = wscanf_s( L"%d %f %hc %lc %S %ls", &i, &fp, &c, 2, &wc, 1, s, 60, ws, 60 );
   wprintf( L"The number of fields input is %d\n", result );
   wprintf( L"The contents are: %d %f %C %c %hs %s\n", i, fp, c, wc, s, ws);
   return 0;
}
```



Questions And Activities:

1. The preprocessor #include <stdio.h> copies the stdio.h file to the source file before main(). If the contents of the stdio.h file are given below, show the converted source file of the C main() program.

```
/* Assuming this is stdio.h  file's content
 The beginning of stdio.h
 Lines of codes
 The end of stdio.h file */
```

```
/* Assuming this is stdio.h  file's content
 The beginning of stdio.h
 Lines of codes
 The end of stdio.h file */

int main()
{
     return 0;
}
```

2. A preprocessor directive #define LIMIT 100 will replace all occurrences of LIMIT in the program to the number 100. Shows the converted source code once the define directive is applied.

```
#include <stdio.h>
#define LIMIT 100

void main(void)
{
    char MyLim[LIMIT] = "The beginning of the hard time";
    char YourLim[LIMIT] = "Of learning C/C++ programming";
    /* other codes */
}
```

```
#include <stdio.h>
#define LIMIT 100

void main(void)
{
    char MyLim[100] = "The beginning of the hard time";
    char YourLim[100] = "Of learning C/C++ programming";
     /* other codes */
}
```

3. Repeat for the following codes.

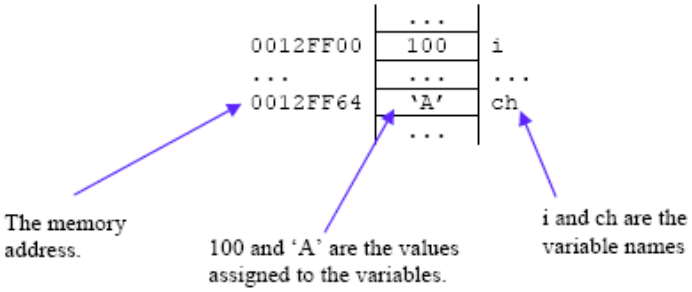
```
#include <stdio.h>
#define WIDTH 4.52
#define LENGTH 4.00

void main(void)
{
    printf("The length of your side = %.2f\n", LENGTH);
    printf("The area is = %.2f\n", LENGTH * WIDTH);
}
```

```
#include <stdio.h>
#define WIDTH 4.52
#define LENGTH 4.00

void main(void)
{
    printf("The length of your side = %.2f\n", 4.00);
    printf("The area is = %.2f\n", 4.00 * 4.52);
}
```

- Variables have a **name**, a **value**, an **address** and **associated with certain type**. In the previous program example, addresses are given in hexadecimal notation. No need for us to know the address of the variable however we can see where the data is is stored using the %p format string. The following Figure tries to describe the variable storage in the computer memory (Random Access Memory – RAM). The address is represented by 8 hexadecimal characters. For every single hex character it consists of 3 binary digits (bits). So the total is:

  8 x 3 bits = 32 bits.

- This is a typical for the 32 bits system. When you compile your program that includes the memory address, your output may be different compared to your friends. This is because different computer may have different memory size because of the different specifications.



4. The float variable named MyFloat, which is initialized to 10.25, is stored in memory location 00FF1244.

```
float MyFloat = 10.25;
```

a. What is the name of the variable?
b. What is the data type of the variable?
c. What is the address of the variable?
d. What is the value of the variable (the stored value)?

a. MyFloat.
b. float.
c. 00FF1244.
d. 10.25

5. Form the previous example, %p (p – pointer to) is used to print a memory address and variable is preceded by the character & (indirection/ address-of operator) to access its address. How should the printf() be called to print the address of variable MyVar and the value that is in it? The output should be something as shown below.

   The variable address = xxxxxxxx, and the value stored = 10.55

```
printf("The variable address = %p, and the value stored = %.2f",
                &MyVar, MyVar);
```

6. The scanf() function allows a user to enter a value into a variable. Its format string specifies the data type to be read and after the format string, the address of the variable must be passed. Write the scanf() to read in a float value for MyVar and precede it with a printf() to prompt the user as shown below. The bold is what is entered by the user.

   ----------Output----------
   Enter the value for MyVar: **10.25**

```
printf("Enter the value for MyVar:");
scanf("%f", &MyVar);
```

7. Similarly, write a printf() and scanf() to read a character code into variable MyChar, as shown below.

   ----------Output----------
   What was the sales code? **S**.

```
printf("What was the sales code?");
scanf("%c", &MyChar);
```

8. scanf() can be used to read more than one variables. Try the following code to read an integer into integer MyInt and a character MyChar.

   ----------Output----------
   Input an integer and a character:**1234 R**

   a. Try the following code:

      ----------Output----------
      Input a character and an integer: **R 1234**

   b. What about the following code?

      ----------Output----------
      Input an integer and a character:**1234R**

   c. What about the following code?

      ----------Output----------
      Input a character and an integer: **R1234**

   In this example be careful with the whitespace. The scanf()/scanf_s() will terminate when there is a whitespace and an empty space is a whitespace. The input must tally with the type character in the format field.
   It is better to be safe by separating all the input into different line such as:

```
printf("Input a character:");
```

```
printf("Input an integer and a character:");
scanf("%d %c", &MyInt, &MyChar);
```

a.
```
printf("Input a character and an integer:");
scanf("%c%d", &MyChar, &MyInt);
```

or

```
printf("Input a character and an integer:");
scanf("%c %d", &MyChar, &MyInt);
```

b.
```
printf("Input an integer and a character:");
scanf("%d%c", &MyInt, &MyChar);
```

or

```
printf("Input an integer and a character:");
scanf("%d%c", &MyInt, &MyChar);
```

c.
```
printf("Input a character and an integer:");
scanf("%c%d", &MyChar, &MyInt);
```

or

```
printf("Input a character and an integer:");
scanf("%c %d", &MyChar, &MyInt);
```

```
scanf("%c", &MyChar);
printf("Input an integer:");
scanf("%d", &MyInt);
```

9. Again, repeat the previous step to read in the character variable MyChar and the integer variable MyInt.

   ----------Output----------
   Input a character and an integer separated by a space: **T 1234**

   Take note that it is even better to provide an accurate prompt for user.

```
printf("Input a character and an integer separated by a space:");
scanf("%c%d", &MyChar, &MyInt);
```

10. Using one scanf(), read in two strings into variables FName and LName. No spaces may be placed within a string because space marks the separation of fields. Print the names as shown below.

    ----------Output----------
    Enter your first and last names separated by a space: **Jodie Foster**
    Your name is Foster, Jodie

```
printf("Enter your first and last names separated by a space:");
scanf("%s%s", FName, LName);
printf("Your name is %s, %s\n", LName, FName);
```

11. Shows the output for the following code snippet by assuming 20.5, 'P' and 2 are entered.

```
printf("Enter a float, then a character, then an integer ");
scanf("%f %c %d", &x, &b, &j);
printf("The answer of x * j = %.2f, a character entered is = %c.\n", x * j, b);
```

```
#include <stdio.h>

void main(void)
{
   float x = 0.0;
   char b = 'A';
   int j = 0;
   printf("Enter a float, then a character, then an integer ");
   scanf("%f %c %d", &x, &b, &j);
   printf("The answer of x * j = %2.f, a character entered is = %c.\n", x * j, b);
}
```

```
Enter a float, then a character, then an integer 20.5 P 2
The answer of x * j = 41.00, a character entered is = P.
Press any key to continue . . .
```

12. Suppose that you have an integer named MyInt.

    a. How would you print this integer's name?
    b. How would you print its value?
    c. How would you know and print out the address in memory where the value is located?

We would declare something like this: int MyInt; then,

a. printf("MyInt");
b. printf("%d", MyInt);
c. printf("%p", &MyInt);

13. Try the following code snippets and answer the questions.

```
#include <stdio.h>

int main(void)
{
        int i = 10;
        printf("variable's name is %c\n", 'i');
        printf("Variable value's is %d\n", i);
        printf("It's address is %p\n", &i);
        return 0;
}
```

```
variable's name is i
Variable value's is 10
It's address is 0012FF60
Press any key to continue . . .
```

a. How do you show the name of a

a. printf("variable's name is %c\n", 'i');
b. printf("Variable value's is %d\n", i);

variable that is called i?
   b. How do you show what value exists in i?
   c. How do you show where i is stored in memory?
   d. What format specifier is used to see the address where a variable is stored?

c. Using the address-of operator (&) as in printf("It's address is %p\n", &i);
d. %p.

14. Try another code.

```
#include <stdio.h>

int main(void)
{
    int i = 100;
    printf("Variable value and its address: %d %p\n", i, &i);
    i = i + 1;
    printf("New variable value and its address: %d %p\n", i, &i);
    return 0;
}
```

```
Variable value and its address: 100 0012FF60
New variable value and its address: 101 0012FF60
Press any key to continue . . .
```

   a. Where was i stored?
   b. Compare to your friend output. The address is same or different?
   c. When 1 was added to i, did the value of i change?
   d. Did the address of i change? Of course the name of i, which is "i", doesn't change.

a. At 0012FF60 (in hex, yours should be different).
b. Some are similar, some are different. It depends on the computer specification.
c. Yes, the value of i does change. It is a variable.
d. No, the address doesn't change. When we declare a variable (in this case it is i), a location in memory was reserved though the value at the location can be changed, the location (address) is fix in this case.

15. Try more code.

```
#include <stdio.h>

int main(void)
{
    int i = 10, j = 20;
    j = j + 1;
    printf("The address of i is %p\n", &i);
    printf("The address of j is %p\n", &j);
    printf("The value of j is %d\n", j);
    return 0;
}
```

```
The address of i is 0012FF60
The address of j is 0012FF54
The value of j is 21
Press any key to continue . . .
```

   a. In which memory location is i stored?
   b. In which memory location is j stored?
   c. Is j's value or address being changed?

a. In my computer, it is 0012FF60 (hex).
b. In my computer, it is 0012FF54 (hex).
c. j's value.

16. scanf() and printf() question.

```
#include <stdio.h>

int main(void)
{
    int i;
    printf("Please enter a whole number: ");
    scanf("%d", &i);
    printf("You entered %d, and ", i);
    printf("the twice value is %d\n", 2*i );
    return 0;
}
```

```
Please enter a whole number: 1234
You entered 1234, and the twice value is 2468
Press any key to continue . . .
```

   a. What does the scanf() function do that is different from what the printf() does?

a. scanf() read stream from standard input, whereas the printf() write stream to the standard output.
b. Yes.
c. In scanf() it is a whitespace such as newline, a space and tab. In

b. Does the scanf() function require format string, similar to what the printf() required?
c. What is different in the way the format strings end in printf() and scanf()?
d. When passing a variable to scanf(), what is being passed, the variable **name** or **value** or **address**?

printf() it is a newline, '\n'.
d. An address of the variable being passed by using the address-of operator (&).

| Main |< scanf() and scanf_s() 1 | scanf() and scanf_s() 3 >| Site Index | Download |

The C scanf() and scanf_s() family: Part 1 | Part 2 | Part 3