

Notes for scanf

These notes do not guarantee that the content will appear on quizzes, midterms, or finals. As long as you understand the concepts and practice coding regularly, you'll be fine.

Purpose of scanf:

scanf is used to take input from the user.

It's like asking the computer to *listen* when you type something on the keyboard. When you enter a value, scanf grabs it and stores it in the computer's memory so the program can use it later.

- Reads user input from the keyboard.
- Stores the input in variables via their **memory addresses**.

Syntax:

```
scanf("formatSpecifier", &variable);
```

- **&** is **required** for primitive types (int, float, double, char)
- **&** is **not needed** for non-primitive types such as strings and arrays.

What is primitive types?

- Basic built-in data types in C
- Fixed size: a fixed number of bytes
- Int, float, double, char are some primitive types.

Common Format Specifiers:

Specifier	Type	Example Input
%d	int	42
%f	float	3.14
%lf	double	2.718
%c	char	A
%s	string	hello

Overall:

- scanf takes input from the user and stores it in the computer's memory.
- Format specifier-> % tells scanf **what type of data to expect** from the user.
- & stores the variable into the **memory address**.
- You use & only for primitive types (int, char, float, double)

Difference between %f and %lf

The main difference between them is with their size.

Type	Size (typical)	Precision
float	4 bytes	~6 decimal digits
double	8 bytes	~15 decimal digits

Example code that uses scanf

Problem Statement:

Write a C program that asks the user to enter a number (or multiple numbers), reads the input, and stores it in memory and print the number that the user has entered.

Solution:

We can use scanf to get the input from the user

1. Define a variable to store the input:

```
c  
  
int number;
```

Always make sure to use the exact name of the variable you declared. For example, if you declared int number; at the beginning, using num later will cause the compiler to give an error.

2. Ask the user for input:

```
c  
  
printf("Enter a number: ");
```

3. Get the number from the user and store it in memory using scanf:

scanf **reads the input** from the user. It has two main parts which are format specifier and memory address. The syntax for scanf is shown below:

```
scanf("formatSpecifier", &variable);
```

- The **format specifier** (like %d) tells scanf what **type of data** to expect.
- The **memory address (&)** tells scanf **where to store** the input in the computer's memory.

Example – single input:

```
c  
scanf("%d", &number);
```

Example – multiple inputs:

```
c  
  
int number1, number2, number3;  
scanf("%d %d %d", &number1, &number2, &number3); // Recommended
```

You can also take inputs separately but it **not a good practice**.

```
c  
  
scanf("%d", &number1);  
scanf("%d", &number2);
```

Note: Always remember to **end with a semicolon (;)**.

4. Print the Number

```
printf("Hey, your number is %d", number);
```

Key Points to Remember When Using printf

1. Always use the correct format specifier for the variable type:

- %d → integer
- %f → float
- %c → character

2. Match variables with format specifiers in the same order.

- Example: printf("X = %d, Y = %d", x, y);

3. Enclose the text in **double quotes " "**; do not use single quotes.
 - In printf, **text (strings) must be enclosed in double quotes " "**.
 - **Single quotes '' are only for a single character**, like 'A' or '9'
4. End the statement with a **semicolon (;)**.
5. **Keep the variable names correct** – they must exactly match what you declared.

Important Note: Please make sure to include % in the format specifier. Some students forget it and write the variable name directly, but that will cause an error.

Note: I saw that a lot of students made this common mistake.

```
printf("Hey there, your number is", number); // X Wrong
```

Why it's wrong:

1. You **want to print the value of number**, but you didn't tell printf **what type of value it is**.
2. The **format specifier** (like %d for integers) tells printf to **replace it with the actual value** of the variable.
3. Without %d, printf just prints the text "Hey there, your number is" and **ignores the variable**. Some compilers may also give a warning or error.

```
printf("Hey there, your number is %d", number); // ✓ Correct
```

Key Points to Remember:

1. **Always include the correct format specifier** for the variable you want to print.
2. **Match the variable with its format specifier** in order.
3. Forgetting %d (or %f, %c) will cause **errors or incorrect output**.

```
main.c [practice lab 1] - Code::Blocks 25.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins Doxygen Settings Help
main.c X
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main()
5 {
6 // ****
7 // C, comments cannot start with # like in Python or shell scripts.
8
9 // Use // for single-line comments.
10
11 // Use /* ... */ for multi-line comments.
12
13 // ****
14
15
16 // DECIMALS
17 // define the variable
18 float decimalNumber;
19 printf("Give me a decimal number: ");
20 scanf("%f", &decimalNumber);
21 printf("Hey there! Your number is %f\n", decimalNumber);
22
23
24 // Character (remember: character -> single character. For strings -> group of characters -> example: name)
25 char alphabet;
26 printf("Enter your favorite alphabet: ");
27 scanf(" %c", &alphabet);
28 printf("Your alphabet is %c", alphabet);
29
30
31 // INTEGERS
32 // define the variable
33 int number;
34 printf("\nEnter a number: ");
35 scanf("%d", &number);
36 printf("Hey there! Your number is %d", number);
37
38 }
```

```
C:\Users\tsrap\OneDrive\Desktop> "C:\Users\tsrap\OneDrive\Desktop\main.exe"
Give me a decimal number: 3.5698
Hey there! Your number is 3.569800
Enter your favorite alphabet: t
Your alphabet is t
Enter a number: 54
Hey there! Your number is 54
Process returned 0 (0x0)   execution time : 25.783 s
Press any key to continue.
```

Things to note:

Take a look at line 27. There is a space before the character format specifier.

Why is that?

- **%c reads a single character literally:** Unlike %d or %f, %c does **not skip whitespace** (spaces, tabs, or newlines). So if there's leftover whitespace in the input buffer—like from pressing Enter after a previous scanf—%c would read that whitespace instead of the character you actually want.

- **The space tells scanf to skip whitespace:**

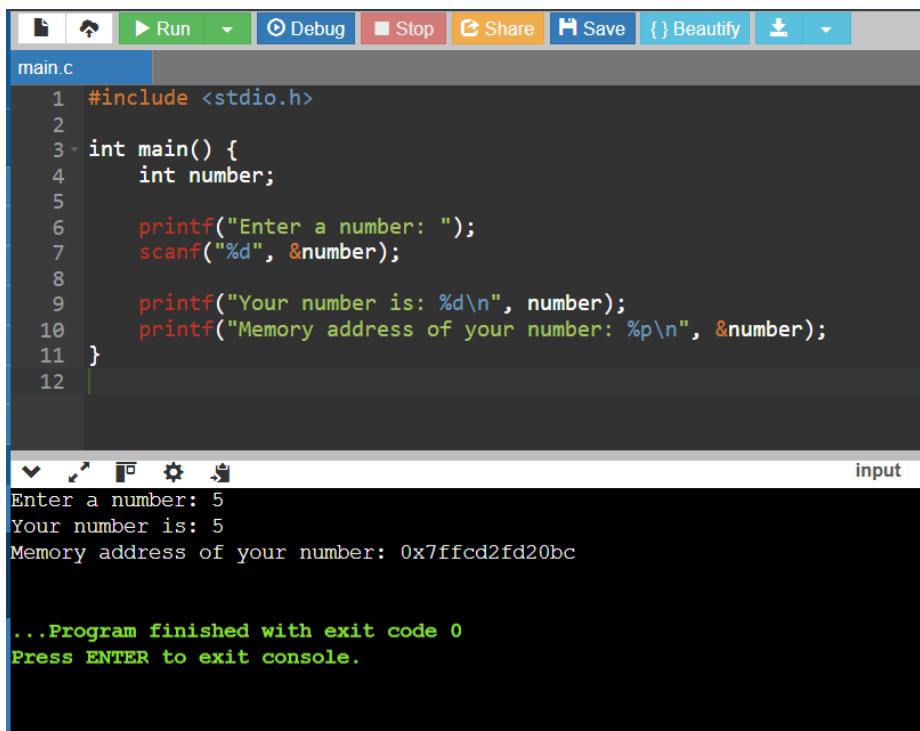
When you write "%c", the leading space instructs scanf to **ignore any whitespace characters** (spaces, tabs, newlines) **before reading the actual character**. This ensures you get the next non-whitespace character.

Note that I used \n. Now, what is \n?

- **Meaning:** \n is an **escape sequence** that represents a **newline** (line break).
 - **Function:** When used in printf, it moves the cursor to the **beginning of the next line**.
-

If you want to find out **where a variable is stored in memory**, you can use the **address-of operator &** with printf and the %p format specifier.

Below are a few example programs to demonstrate this:



```
main.c
1 #include <stdio.h>
2
3 int main() {
4     int number;
5
6     printf("Enter a number: ");
7     scanf("%d", &number);
8
9     printf("Your number is: %d\n", number);
10    printf("Memory address of your number: %p\n", &number);
11 }
12
```

input

```
Enter a number: 5
Your number is: 5
Memory address of your number: 0x7ffcd2fd20bc

...Program finished with exit code 0
Press ENTER to exit console.
```

The following code demonstrates how to take multiple inputs from the user and display their memory addresses.

```
main.c
1 #include <stdio.h>
2
3 int main() {
4     int num1, num2, num3;
5
6     // Take input for 3 numbers in one line
7     printf("Enter 3 numbers separated by space: ");
8     scanf("%d %d %d", &num1, &num2, &num3);
9
10    // Display numbers and their memory addresses
11    printf("\nNumbers and their memory addresses:\n");
12    printf("Number 1: %d, Address: %p\n", num1, &num1);
13    printf("Number 2: %d, Address: %p\n", num2, &num2);
14    printf("Number 3: %d, Address: %p\n", num3, &num3);
15 }
16
```

Enter 3 numbers separated by space: 10 50 100

Numbers and their memory addresses:

Number 1: 10, Address: 0x7ffc9accc9ec

Number 2: 50, Address: 0x7ffc9accc9e8

Number 3: 100, Address: 0x7ffc9accc9e4

...Program finished with exit code 0

Press ENTER to exit console.