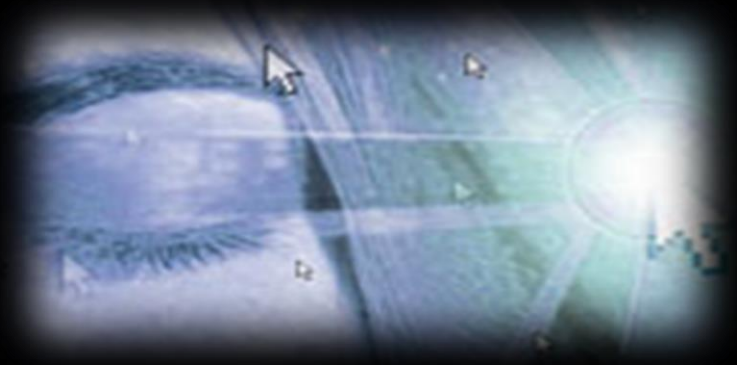
C Formatted I/O

Printing / Reading Formatted Data, Formatting Data



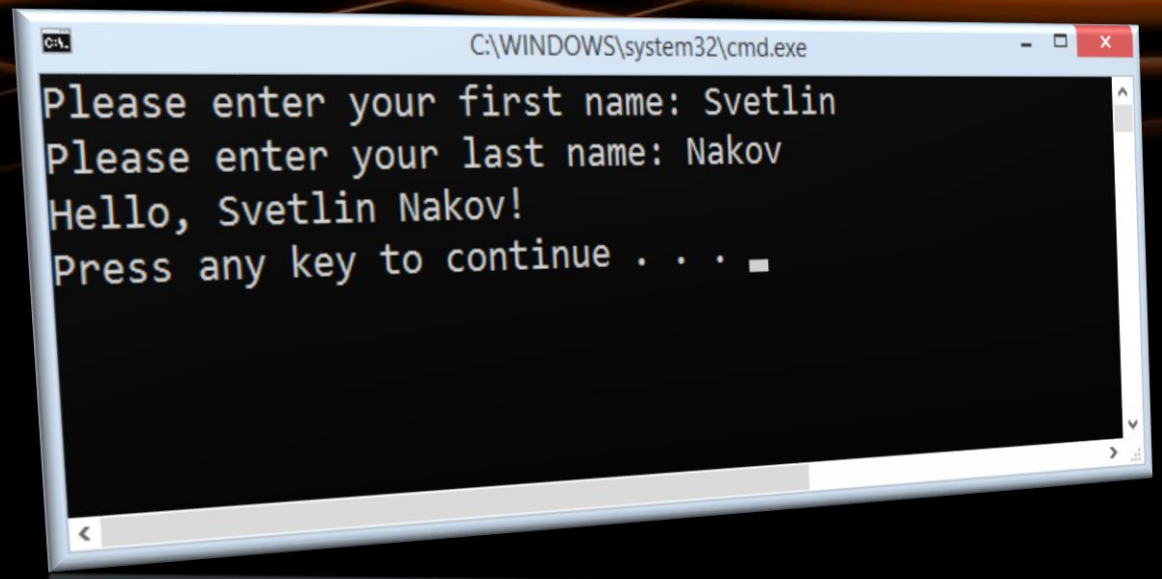
SoftUni Team

Technical Trainers

Software University

http://softuni.bg

Table of Contents



Formatting Output with `printf()`

Printing to the Console

- `printf(const char *format, arg_list)` – prints the format string to the standard output stream
  - Inserts the passed arguments to placeholders in the format
    - Arguments enter format specifiers in the order they are passed

```
char *name = "Gosho";
int age = 20;
float kg = 78.45f;

printf("Name: %s, Age: %d, Weight: %.2f kg", name, age, kg);
// Name: Gosho, Age: 20, Weight: 78.45 kg
```

Format specifier for floating-point number

Integer Conversion Specifiers

Conversion Specifier	Type
`d`	signed integer
`i`	signed integer
`u`	unsigned integer
`ld`	singed long
`lu`	unsigned long
`lld`	signed long long
`llu`	unsigned long long

- Format specifiers are always preceded with a **%** character
- Note: Some format specifiers are different under Windows

Formatting Integers – Example

```
#include <math.h>
#include <limits.h>

int main()
{
    int a = -5, b = 10;
    printf("%d + %d = %d\n", a, b, a + b);
    printf("%f\n", 1.23456789);

    unsigned int num = UINT_MAX;
    printf("%u\n", num);

    long l = LONG_MAX;
    printf("%ld\n", l);

    return 0;
}
```

Floating-Point Conversion Specifiers

Conversion Specifier	Type
`e or E`	exponential notation
`f or F`	floating-point
`L (upper case)`	placed before any floating-point specifier to indicate `long double`

```
float num = 1.2345E-5;  // 1.2345 * 10^-5
printf("Normal: %f\n", num);     // Normal: 0.000012
printf("Scientific: %e\n", num); // Scientific: 1.234500e-05

long double large = 1.23456789123456789;
printf("Long double: %Lf\n", large); // 1.23456789123456789
```

Other Format Specifiers

Conversion Specifier	Type
c	displays as a character
s	displays a char sequence (string)
p	displays a pointer (varies on platforms)

```
printf("Char: %c\n", 'Z');
printf("Char: %c\n", 65);

char *language = "Python";
printf("C is awesomer than %s!\n", language);
printf("%s is at address %p in memory", language, &language);
```

Precision and Length Specifiers

- The precision of floating-point values and the length of strings can be set with the `.{length}` specifier

```
double pi = 3.14159265359;
printf("Double PI: %.11f\n", pi);
printf("Float PI: %.5f", (float) pi);

// Double PI: 3.14159265359
// Float PI: 3.14159

char *str = "Floating-Point";
printf("%.5s");
// Float
```

Text Justification

- Text can be aligned to the left or right by specifying a width in the format `%{width}{specifier}`
  - Aligns text to left if `width` < 0
  - Aligns text to right if `width` > 0
  - Example:

```
char *format = "|%-3d|%10s|\n";
printf(format, 5, "Five");
printf(format, 3, "Three");
printf(format, 11, "Eleven");
```

```
|5  |      Five|
|3  |     Three|
|11 |    Eleven|
```

Text Justification – Example

```
#define TABLE_HEADER_FORMAT "|%10s|%10s|%10s|%10s|\n"
#define TABLE_ROW_FORMAT "|%10d|%10d|%10d|%10d|\n"

int main()
{
    int num = 24;
    int digitOne = num % 10;
    int digitTwo = (num / 10) % 10;

    printf(TABLE_HEADER_FORMAT, "Number", "Digit 1", "Digit 2", "Sum");
    printf(TABLE_ROW_FORMAT, num, digitOne, digitTwo,
        digitOne + digitTwo);

    return 0;
}
```

Formatting Text

Live Demo

Escape Sequences

- Newline **\n** – moves the cursor to the next line

```
printf("Hello, C!\n");
```

- Single **'** and double **"** quote – outputs the literal

```
printf("\'Hey, Taxi!\', yelled the pedestrian.\n");
printf("\"Hey, pedestrian!\", responded the taxi driver.");
```

- Carriage return **\r** – returns to the cursor to the start of the line

```
printf("ABC\rD"); // DBC
```

Escape Sequences (2)

- Backslash **** - outputs the backslash literal

```
printf("\\ - Backslash");
```

- Tab **\t** – outputs the tabulation literal

```
printf("Text\n");
printf("\tIndented");
```

- All escape sequences: https://en.wikipedia.org/wiki/Escape_sequences_in_C#Table_of_escape_sequences

Escape Sequences

Live Demo

Reading Formatted Input

Reading Input from the Console

- `int scanf(const char *format, arg_list)` – scans input in the specified `*format` from the standard input stream
  - All matched format specifiers are assigned to a variable
  - Returns the number of items successfully matched and assigned

```
int a;
int matches = scanf("%d", &a);
if (matches == 0)
    printf("Invalid integer format");
else
    printf("You entered the number %d", a);
```

Pass a pointer (address) to the variable

Reading Numbers from the Console

- `scanf()` allows reading multiple numeric values
  - E.g., reading numbers in the following format:

```
float x, y;
int n;
int matches = scanf("X=%f Y=%f N=%d", &x, &y, &n);
if (matches != 3)
    printf("Invalid input");
else
    printf("X is %f, Y is %f, N is %d", x, y, n);
```

  - Note: Behavior is undefined if overflow occurs! Do not use when expecting overflows.

Reading Long and Long Double

- Reading unsigned long long

```
unsigned long long num;
scanf("%llu", &num);
```

- Reading `long double`

```
long double d;
scanf("%llf", &d);
```

Reading Strings

- When reading with **scanf()** always limit the input string!

```
char name[10];
scanf("%s", name);
```

  - The above code will cause buffer overflow, corrupting adjacent memory

```
char name[10];
scanf("%9s", name);
```

    - **%{limit}s** specifies the maximum size to be read

Reading Strings (2)

- `fgets(char *buffer, int size, FILE *stream)` – reads fixed size bytes from `*stream` into `*buffer`
  - Stream is usually stdin (standard input stream)
  - No danger of buffer overflow

```
#define BUFFER_SIZE 15

int main()
{
    char name[BUFFER_SIZE];
    fgets(name, BUFFER_SIZE, stdin);
    printf("Welcome, %s!", name);
    ...
```

Reading Strings

Live Demo

C Formatted I/O

Questions?

https://softuni.bg/trainings/1212/C-Programming-October-2015

License

- This course (slides, examples, demos, videos, homework, etc.) is licensed under the "Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International" license

- Attribution: this work may contain portions from
  - "Programming Basics" course by Software University under CC-BY-SA license

Free Trainings @ Software University

- Software University Foundation – softuni.org
- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University @ YouTube
  - youtube.com/SoftwareUniversity
- Software University Forums – forum.softuni.bg