Lesson 2

Data types. Program structure / Input – output. Selection

Conclusion:

Problem solving phase

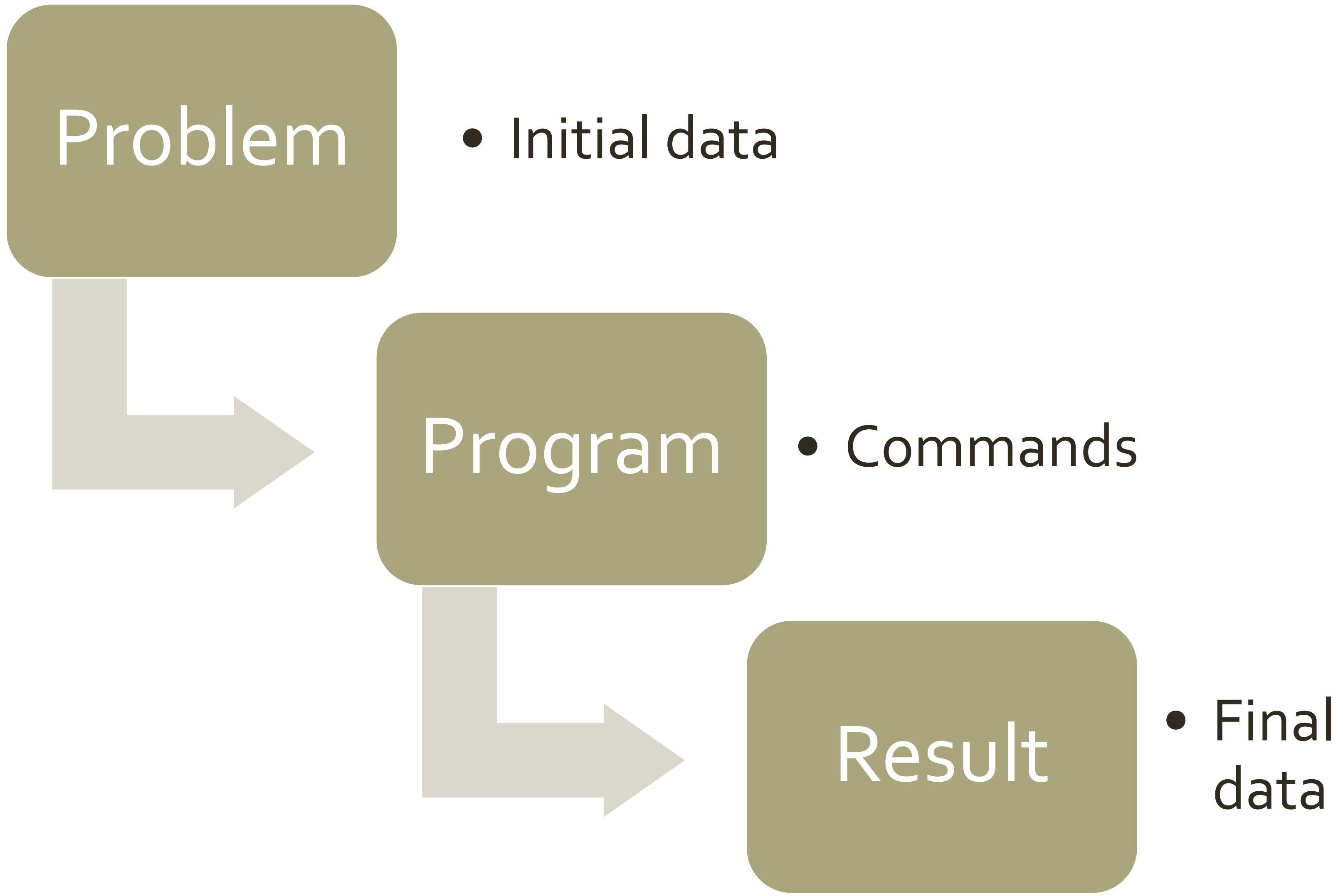
- Analysis
- Algorithm
- Test solution

Implementation phase

- Program
- Test
- Use

Review:

Programming



So, what is programming?

The art of control and communication!

The first name of the science :

Cybernetics

Just for fun

Cybernetics

"the scientific study of control and communication in the animal and the machine.“

Norbert Wiener, 1948

Кибернетика

– реакционная лженаука, возникшая в США после второй мировой войны и получившая широкое распространение и в других капиталистических странах: форма современного механицизма[1]

Краткий философский словарь, Москва, 1954

1 Cybernetics - reactionary pseudoscience that arose in the USA after the Second World War and became widespread in other capitalist countries: a form of modern mechanism

Information,

Data

and

Data types

Information

Data

- Information — is a designation of content that we received from the outside world in the process of adapting us and our feelings to it. (Norbert Winner)

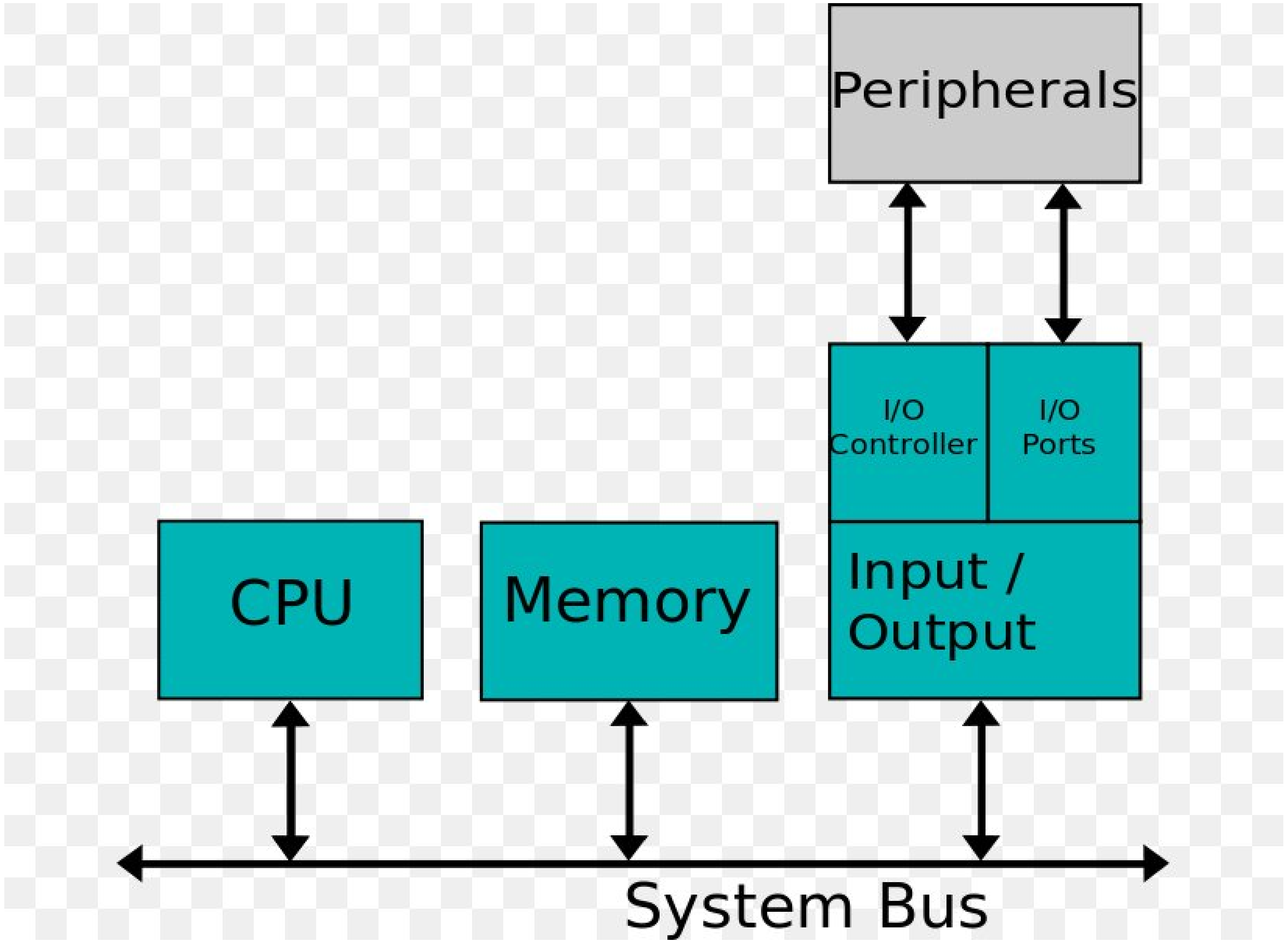
- Data - a reusable presentation of information in a **formalized form suitable for transmission, communication or processing**(ISO/IEC 2382:2015)

Information:
visual,
sound,
tactile,
taste,
olfactory.

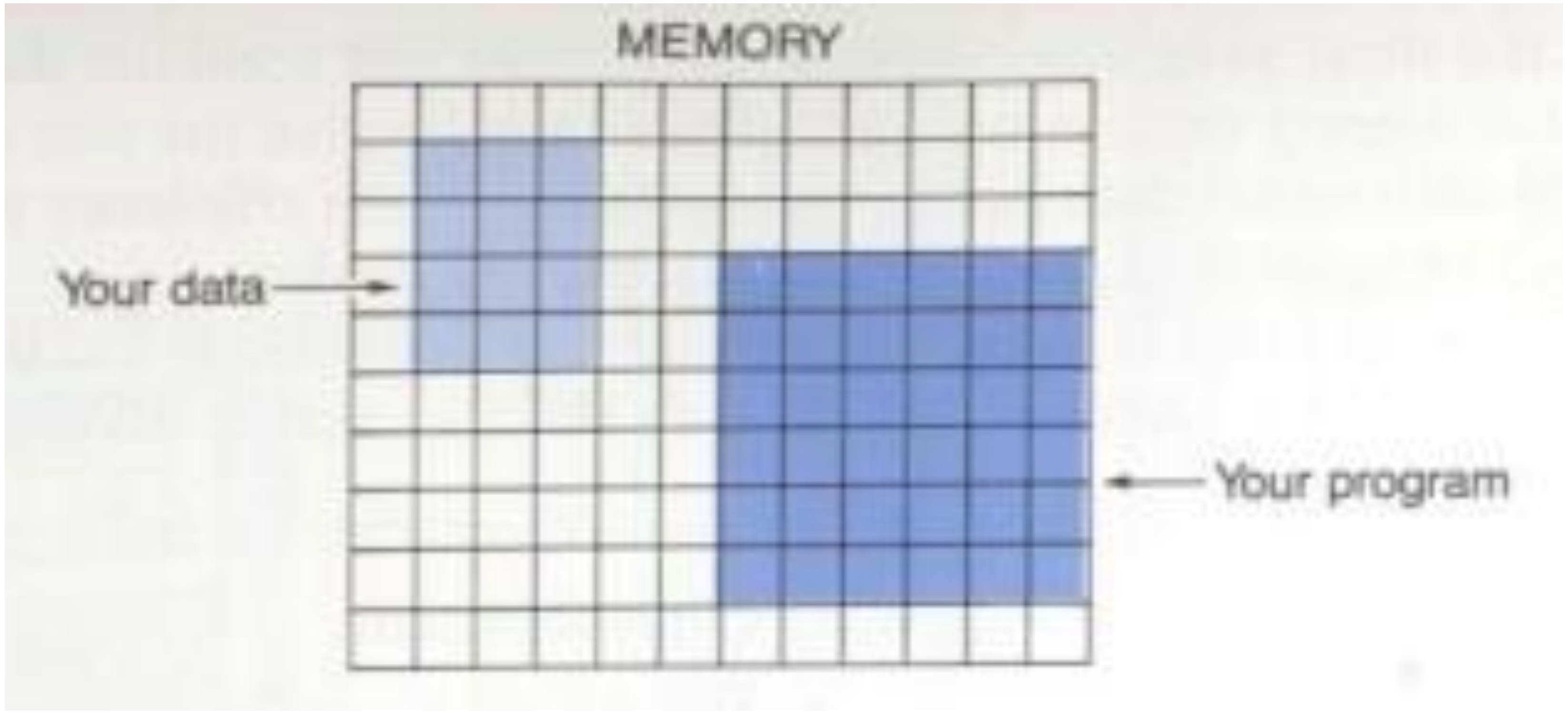
Data:

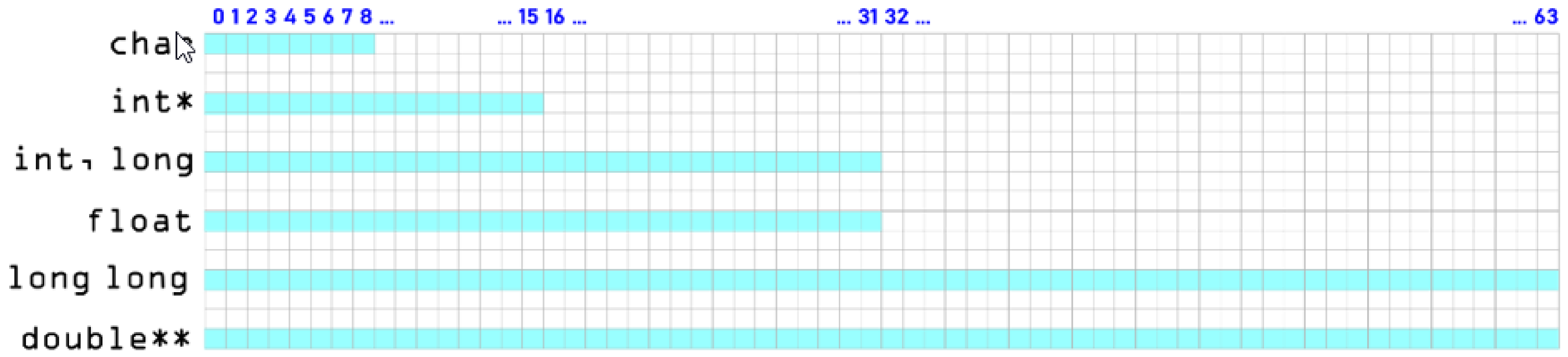
0 1

General scheme of computer system



Data and memory





* pentru compilatoare mai puțin performante
** numere reale pentru calcule de mare precizie

Data and memory

C basics

Program structure / Input - output

C language alphabet

- Alphabet letters: from `a` to `z`, from `A` to `Z`;
- Digit symbols: from `0` to `9`;
- Special symbols like: `@ # $ % ^ & * ( ) { } [ ] | \ , : ;` etc.

C language

“words”

identifiers - sequence of letters, numbers and the _ symbol, which does not contain spaces. The first symbol of the identifier cannot be a digit.

Examples:

```
number_1, n2, A, b6, draw_object
```

keywords - standard identifiers, which do not have to be declared in the program and are uniquely recognized by the compiler.

Example:

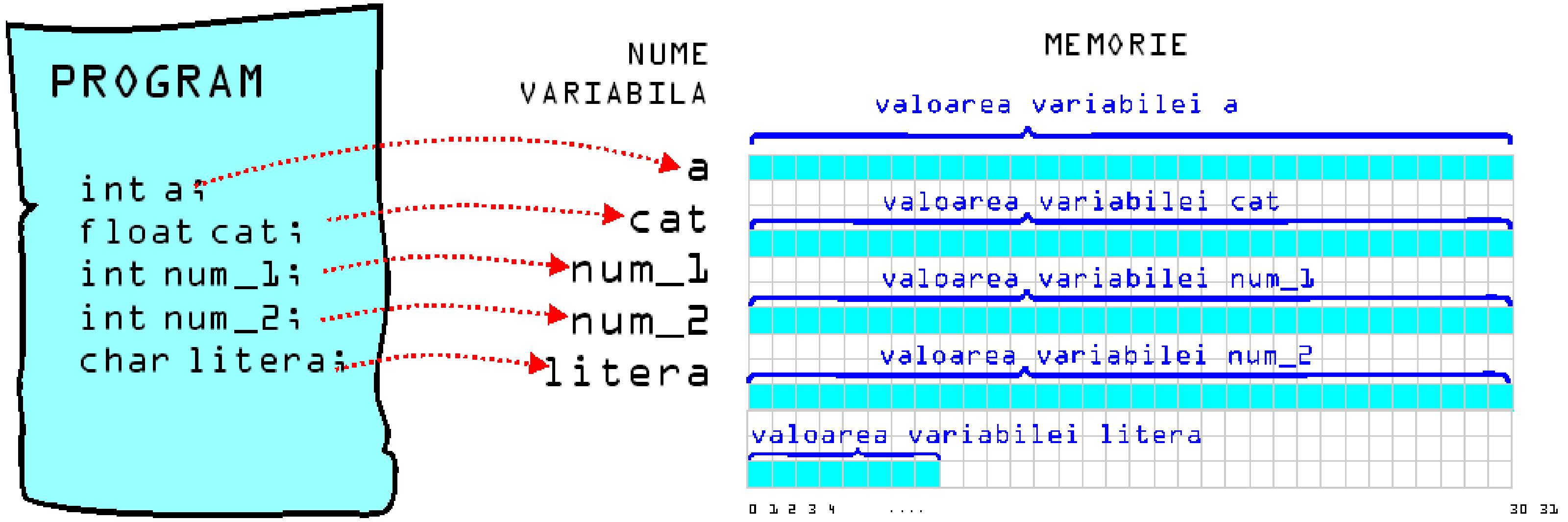
```
int, double, float, system, getch, struct, return,…
```

Variables

Variables are memory locations of predefined sizes that can store data of a certain type and are identified by name.

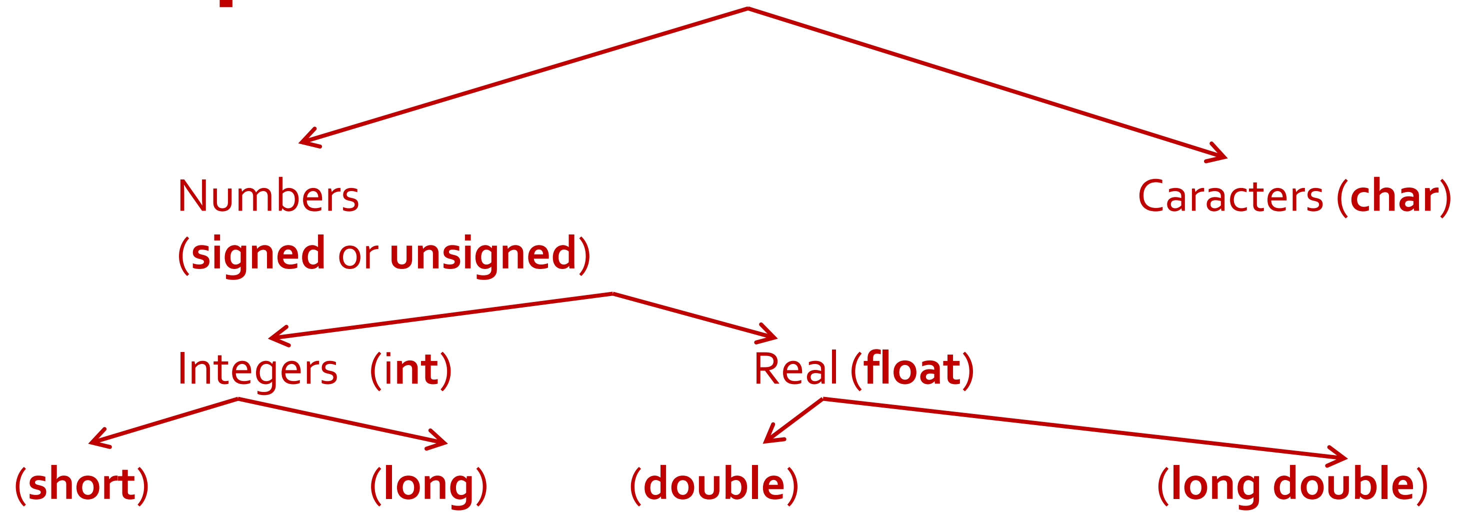
The **variable name** is an identifier (word) consisting of one or more letters and numbers, the first symbol being a letter.

The variable name can also contain the underscore symbol



Basic data types in C

Simple Variables



type	values	memory
char, signed char	-128 .. 127	1 byte
unsigned char	0 .. 255	1 byte
short, signed short	- 32768 .. 32767	2 bytes
unsigned short	0 .. 65535	2 bytes
int, signed int	- 32768 .. 32767	2 bytes
unsigned, unsigned int	0 .. 65535	2 bytes
long, signed long	-2147483648 .. 2147483648	4 bytes
float	-3.4e38 .. 3.4e38	4 bytes
double	-1.7e308 .. 1.7e308	8 bytes
long double	-3.4e4932 .. 3.4e4932	10 bytes

```
#include <stdio.h>
#include <stdlib.h>

#define pi 3.14

int a, b, c;
```

Directive și declarații

Directive: #include, #define

Variabile declarate: a, b, c

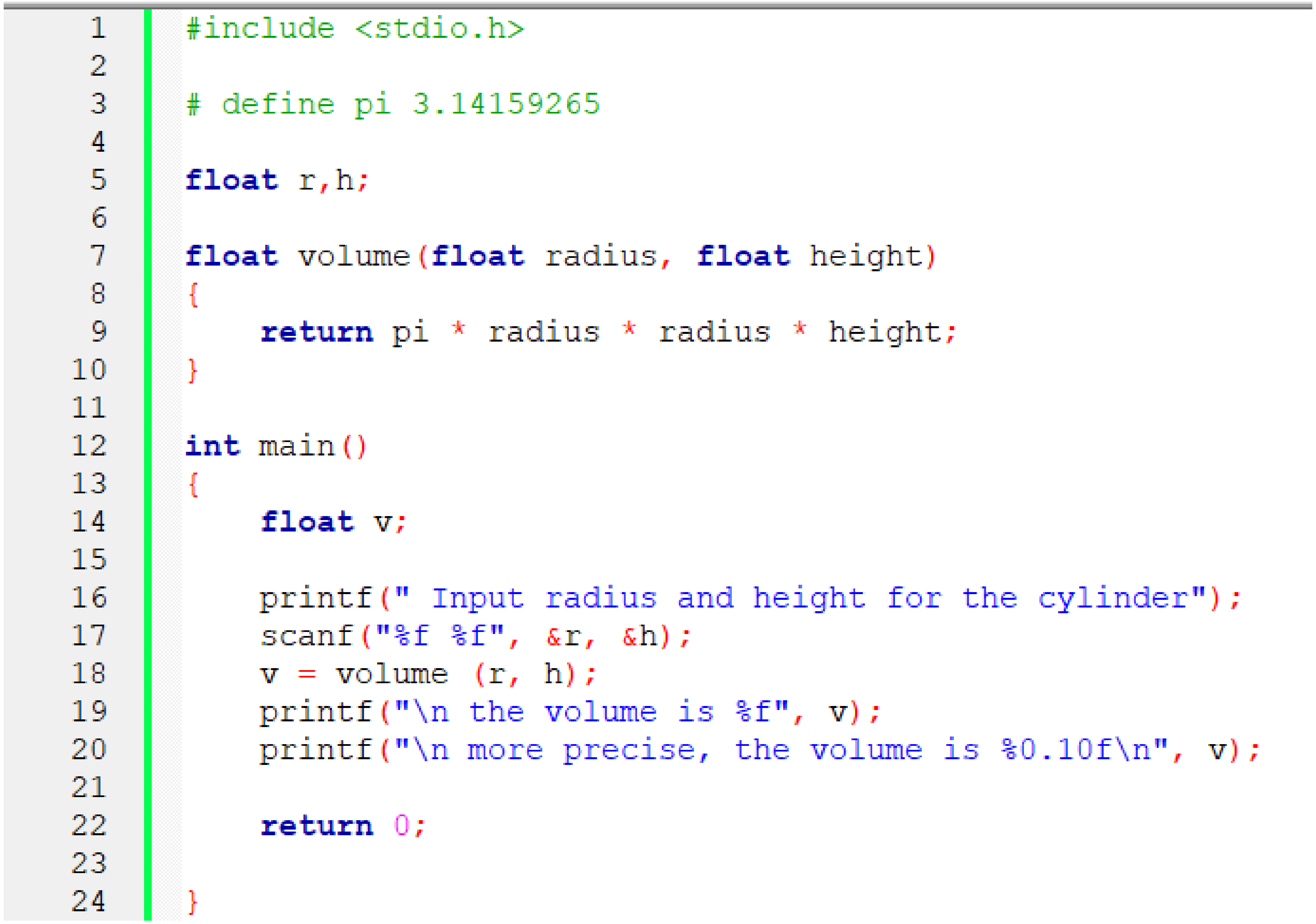
```
int main()
{
        a = 25;
        b = 37;

        c = a * b;

        printf("%d * %d = %d", a, b, c);

        return 0;
}
```

Descrieri funcții

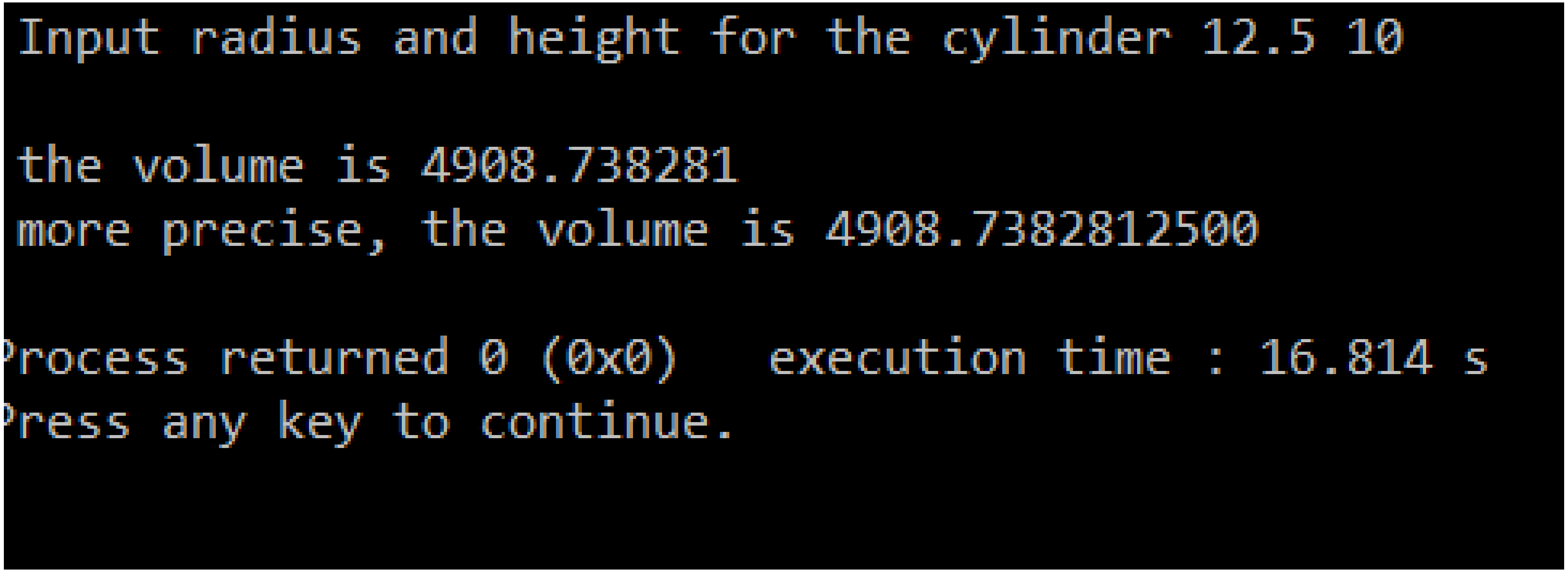
Funcția main() – funcția principală din program. Prezența ei este obligatorie.

Programul poate conține și alte funcții, scrise de programator

Program Structure



```c
#include <stdio.h>

# define pi 3.14159265

float r,h;

float volume(float radius, float height)
{
    return pi * radius * radius * height;
}

int main()
{
    float v;

    printf(" Input radius and height for the cylinder");
    scanf("%f %f", &r, &h);
    v = volume (r, h);
    printf("\n the volume is %f", v);
    printf("\n more precise, the volume is %0.10f\n", v);

    return 0;

}
```



```
Input radius and height for the cylinder 12.5 10

the volume is 4908.738281
more precise, the volume is 4908.7382812500

rocess returned 0 (0x0)   execution time : 16.814 s
ress any key to continue.
```

Operators

Arithmetic

• Inverse sign	-
• Increment	++
• Decrement	--
• Addition	+
• Substraction	-
• Multiplication	*
• Division	/
• Remainder	%

Operators

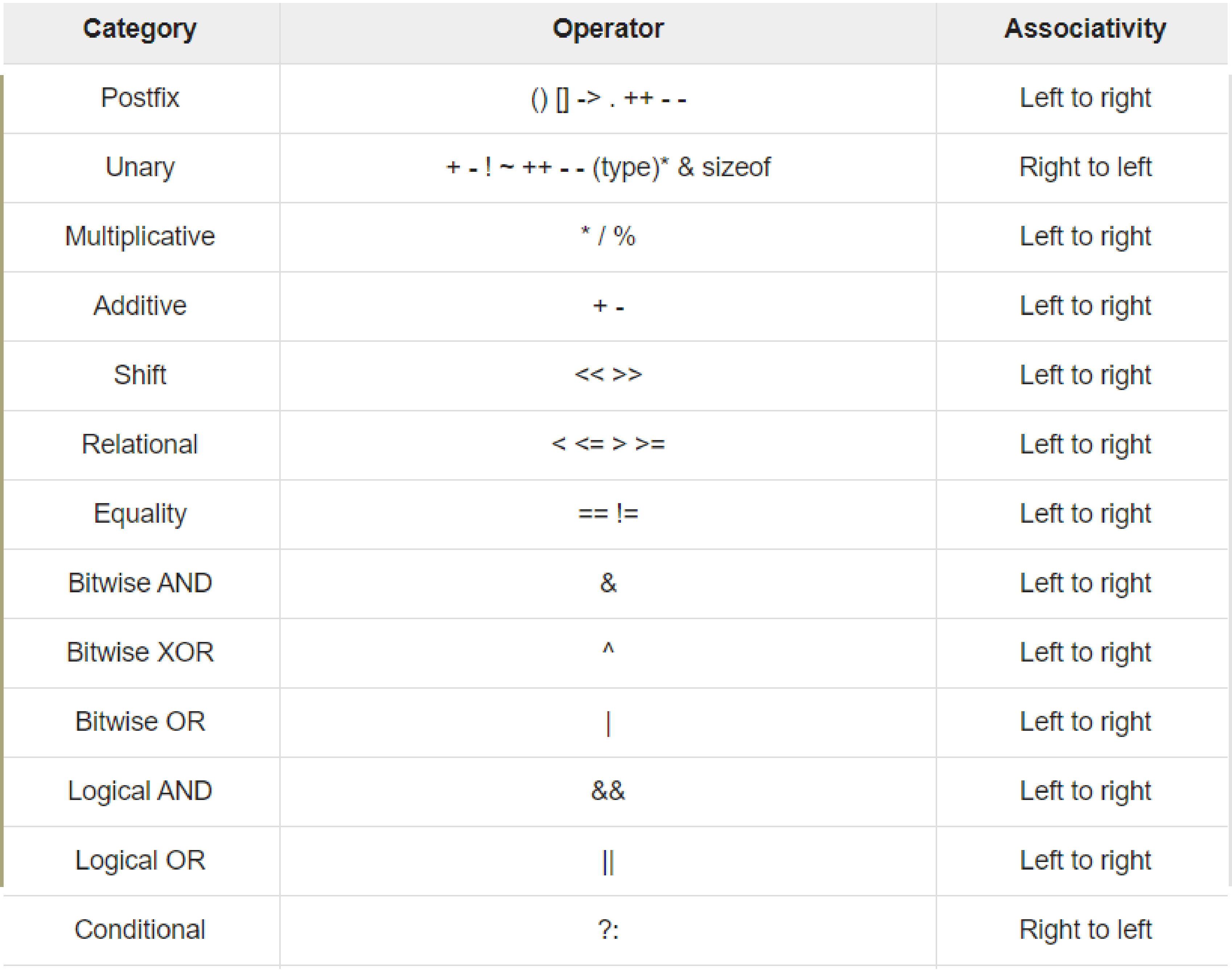
Relational

Equal	==	Not equal	!=
Less	<	Greather	>
Less or equal	<=	Greater or equal	>=

Logic

Negation	!
Conjuncion	&&
Disjunction	||

Operators priority



Category	Operator	Associativity
Postfix	() [] -> . ++ - -	Left to right
Unary	+ - ! ~ ++ - - (type)* & sizeof	Right to left
Multiplicative	* / %	Left to right
Additive	+ -	Left to right
Shift	<< >>	Left to right
Relational	< <= > >=	Left to right
Equality	== !=	Left to right
Bitwise AND	&	Left to right
Bitwise XOR	^	Left to right
Bitwise OR	|	Left to right
Logical AND	&&	Left to right
Logical OR	||	Left to right
Conditional	?:	Right to left

Input / Output

Primary IO functions

Description: The C library function **int getchar(void)** gets a character (an unsigned char) from stdin.

Declaration: **int getchar(void)**

Parameters - NONE

Return Value: the function returns the character read as an unsigned char cast to an int or EOF on end of file or error.

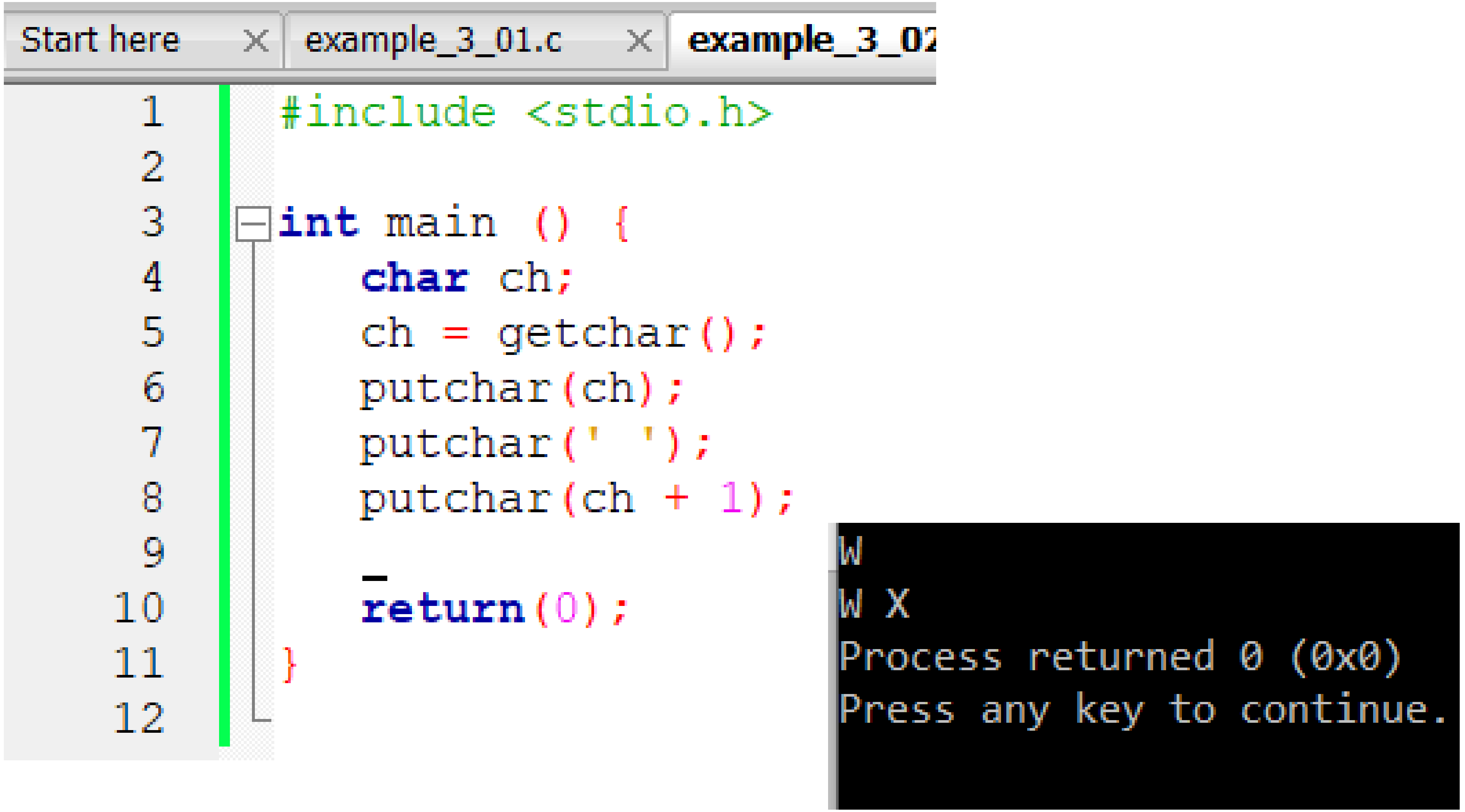
Description: The C library function int putchar(int char) writes a character (an unsigned char) specified by the argument char to stdout.

Declaration: **int putchar(int char)**

Parameters: char – This is the character to be written.

Return Value: the function returns the character written as an unsigned char cast to an int or EOF on error.

Prototypes in: **<stdio.h>**



Start here × example_3_01.c × example_3_02

```c
#include <stdio.h>

int main () {
    char ch;
    ch = getchar();
    putchar(ch);
    putchar(' ');
    putchar(ch + 1);

    return(0);
}
```

```
W
W X
Process returned 0 (0x0)
Press any key to continue.
```

gets / puts

Description: the C library function **char *gets(char *str)** reads a line from stdin and stores it into the string pointed to by str. It stops when either the newline character is read or when the end-of-file is reached, whichever comes first.

Declaration: char *gets(char *str)

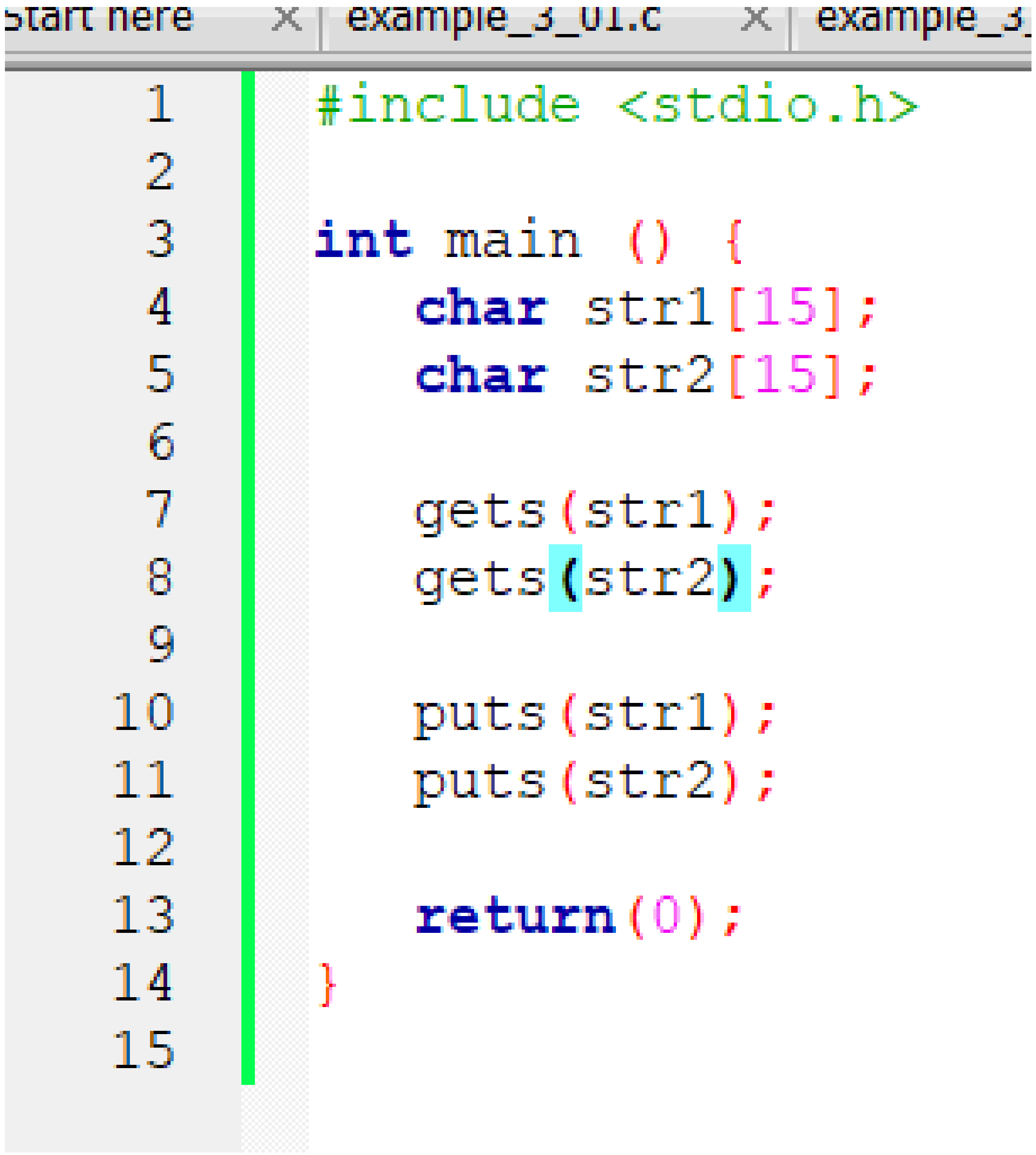
Parameters: str – the pointer to an array of chars.

Return Value: the function returns str on success, and NULL on error or when end of file occurs, while no characters have been read.

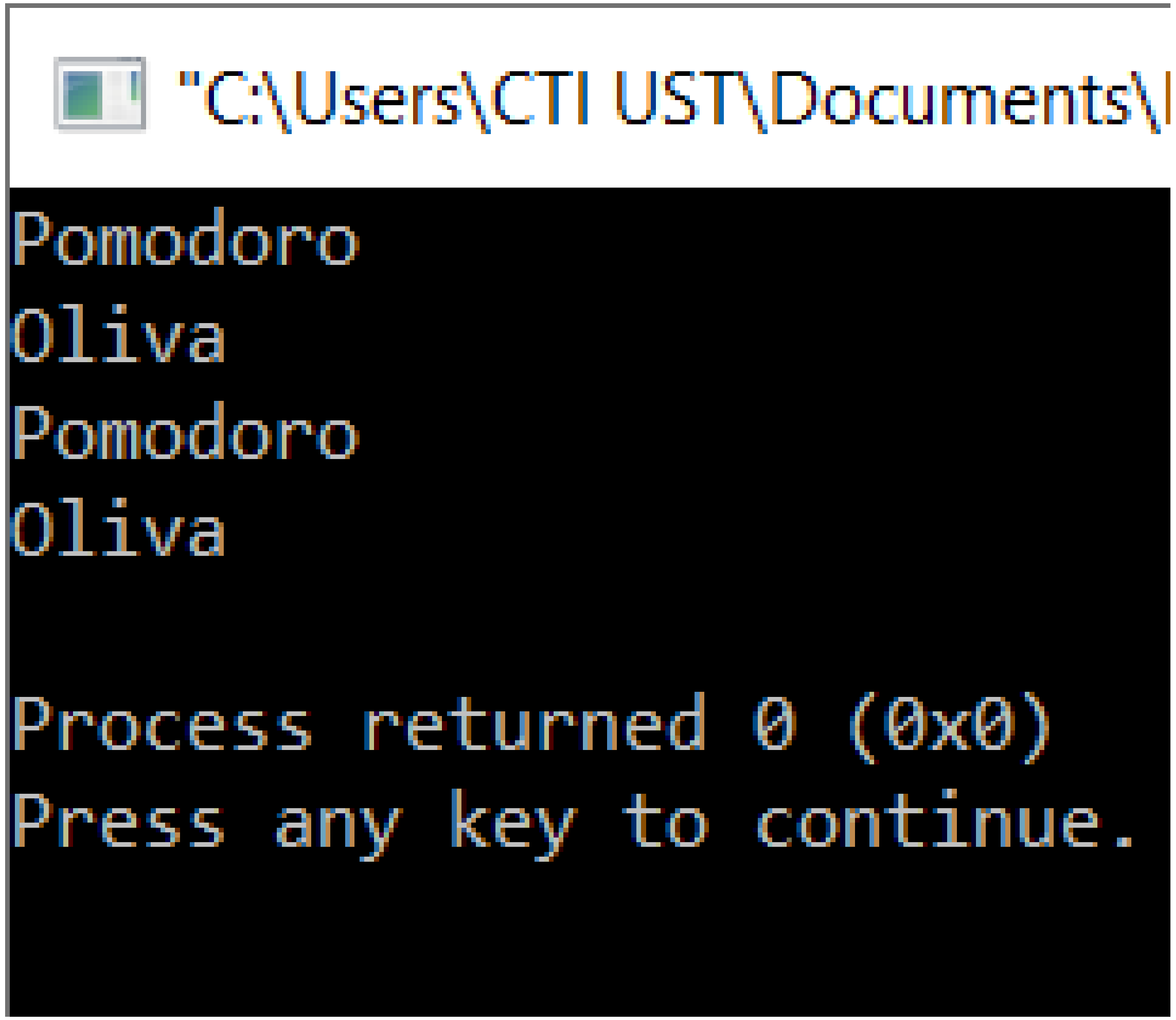
Description: the C library function **int puts(const char *str)** writes a string to stdout up to but not including the null character. A newline character is appended to the output.

Declaration: **int puts(const char *str)**
Parameters: str – the C string to be written.

Return Value: If successful, non-negative value is returned. On error, the function returns EOF.



```c
#include <stdio.h>

int main () {
    char str1[15];
    char str2[15];

    gets(str1);
    gets(str2);

    puts(str1);
    puts(str2);

    return(0);
}
```



```
Pomodoro
Oliva
Pomodoro
Oliva


Process returned 0 (0x0)
Press any key to continue.
```

scanf()

The C library function `int scanf(const char *format, ...)` reads formatted input from stdin.

Declaration: `int scanf(const char *format, ...)`

Parameters:
`format` – This is the C string that contains one or more of the following items – whitespace character, Non-whitespace character and Format specifiers.

A format specifier will be like `[%] [*][width][modifiers][type]`

Format specifier	Description
`%d`	întreg zecimal cu semn
`%i`	întreg zecimal, octal (0) sau hexazecimal (0x, 0X)
`%o`	întreg în octal, fără 0 la inceput
`%u`	întreg zecimal fără semn
`%x, %X`	întreg hexazecimal, fără 0x/0X; se folosesc cifrele hexazecimale a-f în cazul %x, și A - F pentru %X
`%c`	caracter
`%s`	şir de caractere, până la spațiu. De asemenea poate fi fixat numărul de caractere citite
`%f, %F`	real fără exponentă;
`%e, %E`	numere reale cu mantisă şi exponent zecimal
`%g, %G`	numere reale în format %f sau %e, funcţie de valoare
`%p`	pointer, în formatul tipărit de printf

length	Description	
lipsește	Conform specificatorului de tip	
hh	Character cu semn	
h	întreg scurt	
l	întreg lung	
ll	Long long (cu d, i)	
L	(cu f, F, e, E, g, G, a, A) – long double	

```
%ld, %lld      long, long long

%lf, %llf      double, long double
```

`printf()`

The C library function

`int printf(const char *format, ...)`

sends formatted output to stdout.

Declaration: `int printf(const char *format, ...)`

Parameters:

- **format** – This is the string that contains the text to be written to stdout. It can optionally contain embedded format tags that are replaced by the values specified in subsequent additional arguments and formatted as requested.
- Format tags prototype is `%[flags][width][.precision][length]specifier`

width	Descriere
(number)	Numărul minim de caractere care urmează să fie afișat. La o lungime mai mică a șirului în față se adaugă spații. În cazul unui șir mai lung, acesta se va afișa integral.
*****	Lățimea nu este specificată explicit, dar ca un argument adițional de tip întreg care precede argumentul care trebuie formatat.

Flag	Descriere
-	Aliniere la stânga în câmpul de lățime dată; Implicit este setată alinierea la dreapta.
+	Afișare forțată a semnului numărului (+ sau -) chiar și pentru numere pozitive. Implicit doar numerele negative sunt afișate cu semn.
(spațiu)	Dacă nu se afișează semnul, se adaugă un spațiu în fața valorii numărului.
#	Folosit cu specificatorii o, x sau X în fața valorii va apărea 0, 0x sau 0X respectiv (pentru valori diferite de 0). Folosit cu a, A, e, E, f, F, g or G obligă afișarea punctului zecimal (chiar dacă partea reală este nulă).
0	Adaugă (0) în locul spațiilor în fața valorii afișate dacă a fost setat specificatorul de lățime.

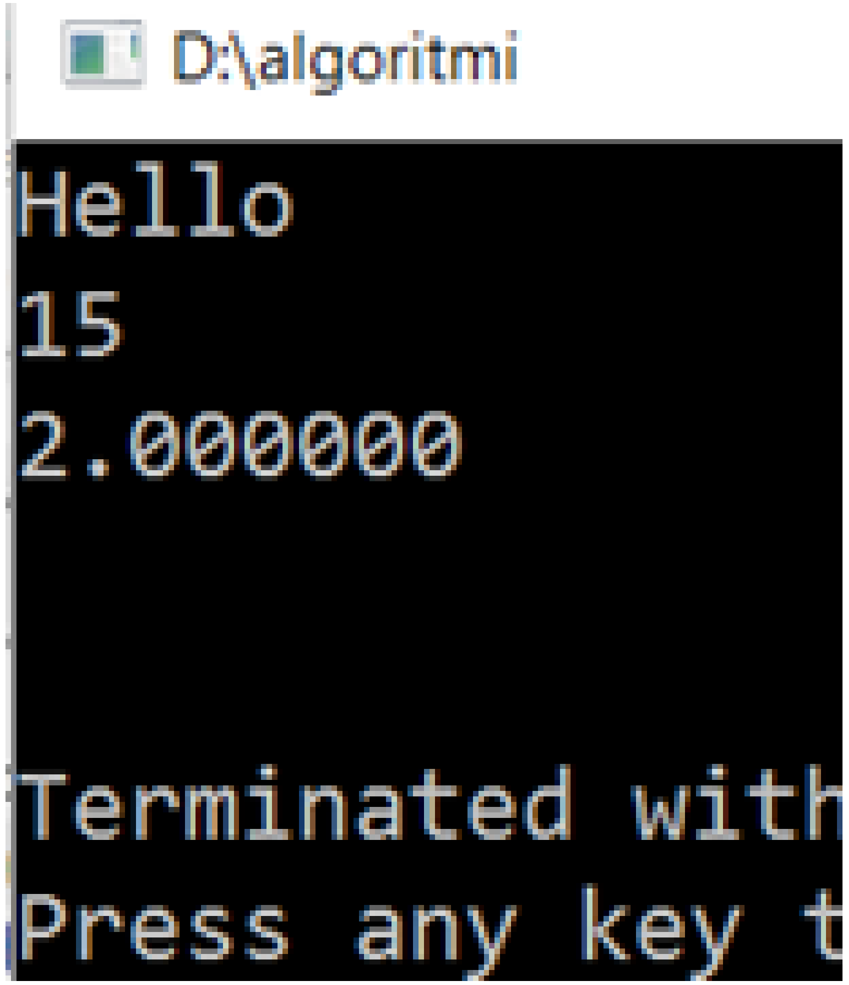
.precizie	Descriere
.număr	Pentru specificatorii întregi (d, i, o, u, x, X): numărul minim de caractere pentru afișare. Posibile 0 în față, dacă lungimea reală este mai mica decât cea cerută. Numerele mai lungi nu se trunchiază. Precizia 0 înseamnă că nu se va scris nici un caracter pentru valoarea 0. Pentru a, A, e, E, f și F specifică numărul de cifre tipărite după punctul zecimal. Pentru g și G: Numărul maximal de cifre semnificative care urmează a fi tipărit. Pentru s: numărul maximal de caractere pentru a fi tipărite. Implicit se afișează toate caracterele până la ‘\0’.
.*	Precizia nu este specificată explicit, dar ca un argument adițional de tip întreg care precede argumentul care trebuie formatat.

Escape sequences

Escape sequence Description

\n	Newline.	Position the cursor at the beginning of the next line.
\t	Horizontal tab.	Move the cursor to the next tab stop.
\a	Alert.	Produces a sound or visible alert without changing the current cursor position.
\\	Backslash.	Insert a backslash character in a string.
\"	Double quote.	Insert a double-quote character in a string.

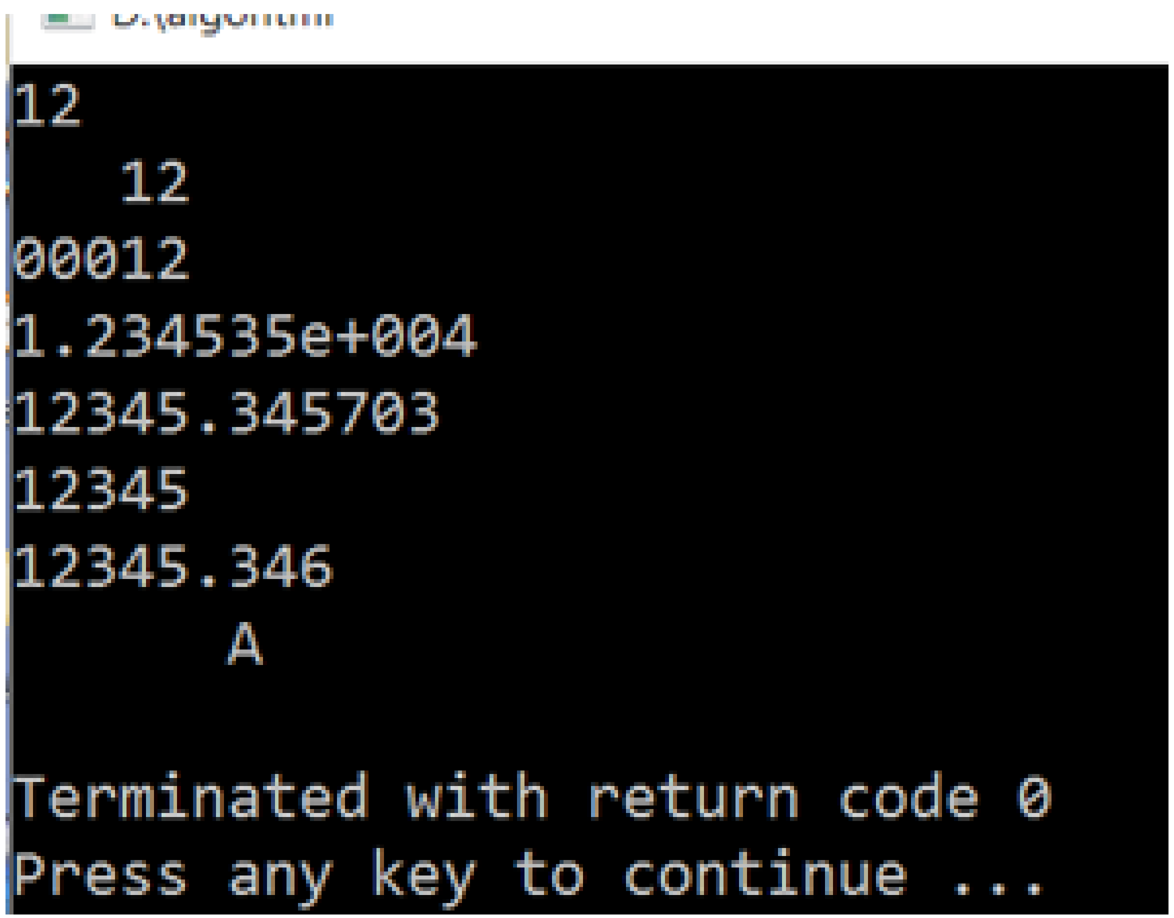
```
#include <stdio.h>
   int a;
   float count;
   main ()
   {  printf ("Hello\n");
      a= 15;
      printf("%d", a);
      count = a / 7.5;
      printf("%f", count);
      return 0;
   }
```



D:\algoritmi

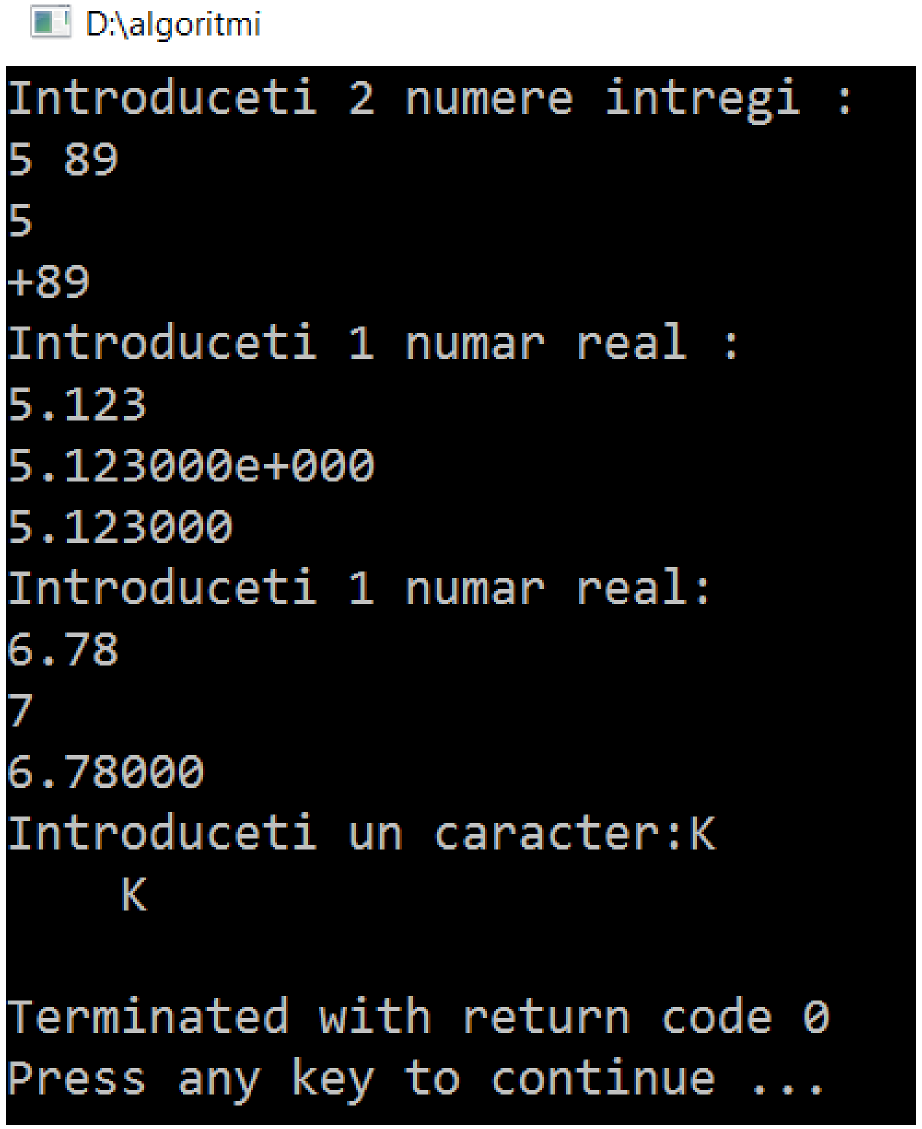
```
Hello
15
2.000000



Terminated with
Press any key t
```

```c
#include <stdio.h>
int a, b;
float c;
char d;
main()
{  a = 12; b=-34; c= 12345.345678; d='A';
   printf ("%d\n", a);
   printf ("%5d\n", a);
   printf ("%05d\n", a);
   printf ("%e\n", c);
   printf ("%f\n", c);
   printf ("%.0f\n", c);
   printf ("%.3f\n", c);
   printf ("%7c", d);
   return 0;
}
```



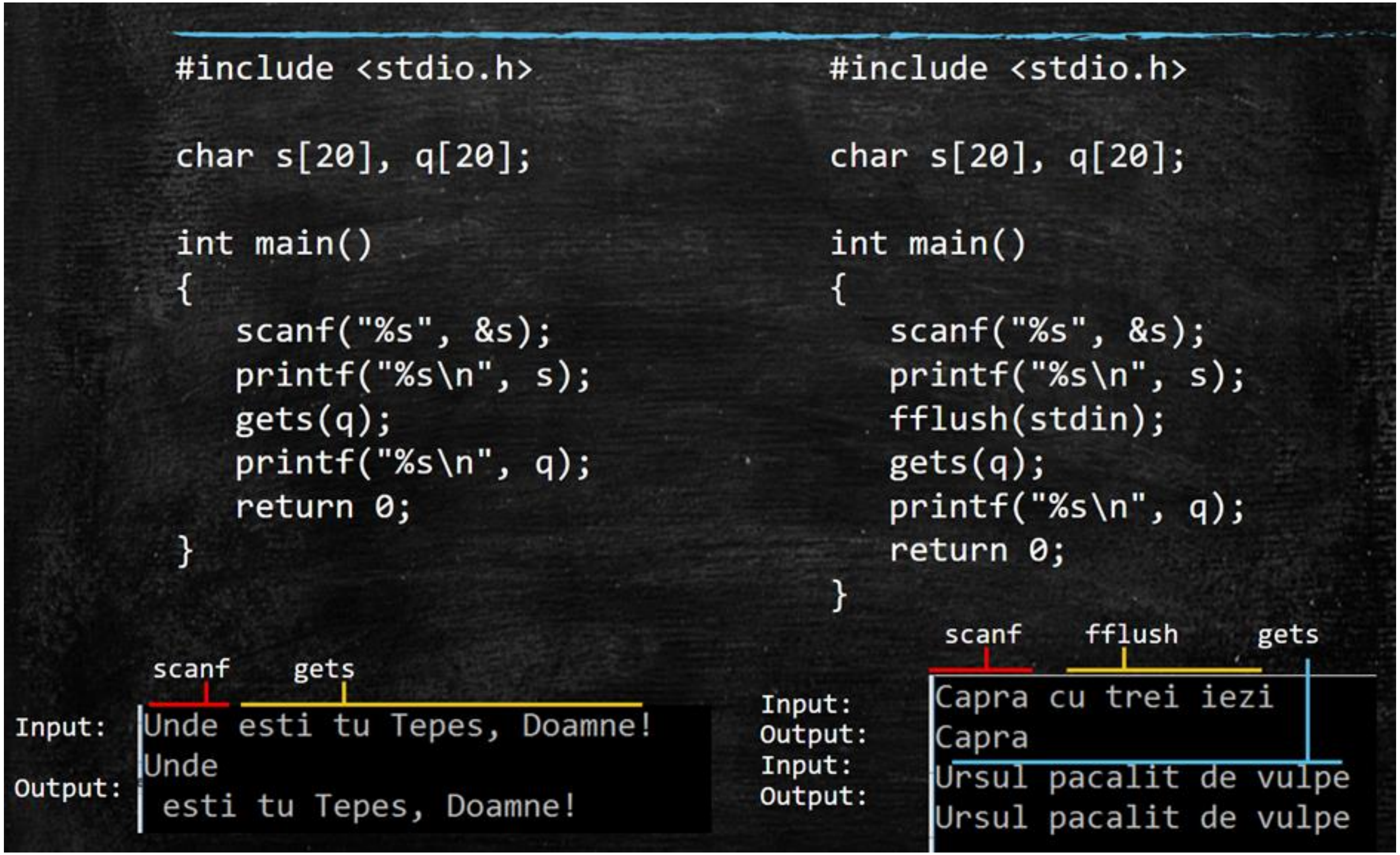
```
12
   12
00012
1.234535e+004
12345.345703
12345
12345.346
      A

Terminated with return code 0
Press any key to continue ...
```

```
#include <stdio.h>
int a, b; float c, c2; char d;
main()
{  printf ("Introduceţi 2 numere întregi :\n");
   scanf ("%d%d", &a, &b);
   printf ("%d\n", a); printf ("%+d\n", b);
   printf ("Introduceţi 1 număr real :\n");
   scanf ("%f", &c);
   printf ("%e\n", c); printf ("%f\n", c);
   printf ("Introduceţi 1 număr real:\n");
   scanf ("%f", &c2;   fflush(stdin);
   printf ("%.0f\n", c2); printf("%.5f\n",c2);
   printf ("Introduceţi un caracter:");
   scanf ("%c", &d); printf ("%5c", d);
   return 0;
}
```


D:\algoritmi

```
Introduceti 2 numere intregi :
5 89
5
+89
Introduceti 1 numar real :
5.123
5.123000e+000
5.123000
Introduceti 1 numar real:
6.78
7
6.78000
Introduceti un caracter:K
    K

Terminated with return code 0
Press any key to continue ...
```

Note. Exclude the operator **fflush(stdin).** Verify the new result after execution.



```c
#include <stdio.h>

char s[20], q[20];

int main()
{
    scanf("%s", &s);
    printf("%s\n", s);
    gets(q);
    printf("%s\n", q);
    return 0;
}
```

```
Input:  Unde esti tu Tepes, Doamne!
Output: Unde
         esti tu Tepes, Doamne!
```

```c
#include <stdio.h>

char s[20], q[20];

int main()
{
    scanf("%s", &s);
    printf("%s\n", s);
    fflush(stdin);
    gets(q);
    printf("%s\n", q);
    return 0;
}
```

```
Input:  Capra cu trei iezi
Output: Capra
Input:  Ursul pacalit de vulpe
Output: Ursul pacalit de vulpe
```

Homework

(Laboratory Lesson)

1. Write a program to convert x MDL to USD and EUR. Use rates for the day of code writing.
2. Write a program to calculate the volume of a sphere (try find the formula!). Print results with at least 10 significant digits
3. Write a program to print using pseudo graphics symbols a table of 3 rows and 4 columns. The first column will contain student name (no more than 15 symbols); second column – student birthplace(no more than 20 symbols); third column – student school (no more than 25 symbols) ; last one – the student amount of Facebook friends (no more than 5000).
   Data will be entered from the keyboard. You have to obtain a prefect alignment for any data!
   Example:

```
     Ion      Duruitoarea   Liceu    257
   Maria           Pepeni   Liceu     18
Emanuela          Ciocana    CEFB   3214
```

```
(this is an example for 50% points)
```

LOOP JOKE OF THE DAY

An IT Professional died at 45 years and went to heaven. He asked God why he died at such an early age.
God replied:
"Son, the way u have filled up your time-sheet for number of hours worked, u have already lived 82 years"