Introduction to Programming in C

(IS-FEE-10061S)

Białystok University of Technology
Faculty of Electrical Engineering
Academic year 2023/2024

Workshop no. 08 (18.04.2024)

Jarosław Forenc, PhD

Topics

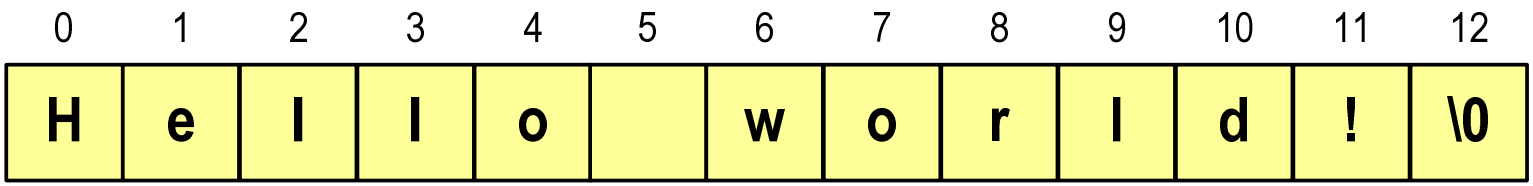
- Strings
  - implementation
  - declaration
  - initialization
  - standard I/O functions
  - string and character printing
  - string and character reading
  - header file string.h
  - header file ctype.h

Strings: implementation

- String (charcter string, character array, string constant, C-string) - a series of one or more characters enclosed in quotation marks

```
"Hello world!"
```

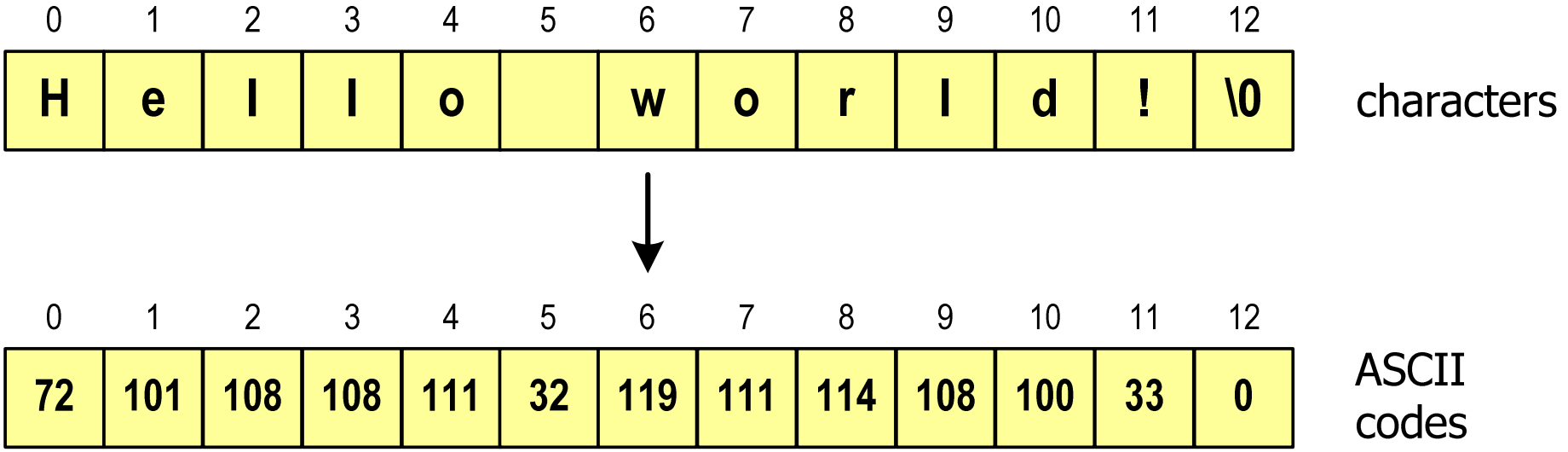
- Implementation - an array whose elements are single characters (char type)



- The last character ('\0', the null character) marks the end of the string

Strings: implementation

- In fact, the array stores the corresponding ASCII codes (numbers) instead of characters



Strings: declaration

- Declaration of a variable storing a string

```
char variable_name[size];
```

Example:

```
chat txt[10];
```

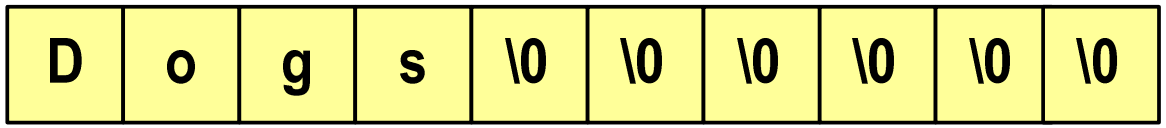
- The txt array can store strings with a maximum length of 9 characters

Strings: initialization

- String initialization

```
char txt1[10] = "Dogs";
char txt2[10] = {'D','o','g','s'};
char txt3[10] = {68,111,103,115};
```

- The other elements of the array are initialized to \0




```
char txt4[]   = "Dogs";
char *txt5    = "Dogs";
```

Strings: initialization

- Initialization is possible only together with the declaration

```
char txt[10];
txt = "Dog";    /* ERROR!!! */
```

- Assigning the text "Dogs" to the txt variable requires calling the strcpy() function from the string.h header file

```
char txt[10];
strcpy(txt,"Dog");
```

Character constant

- A character constant is one character enclosed in single quotes

```
char ch = 'x';
```

- In fact, a character constant is an integer whose value corresponds to the ASCII code of the represented character
- Instead of the above code, we can write:

```
char ch = 120;
```

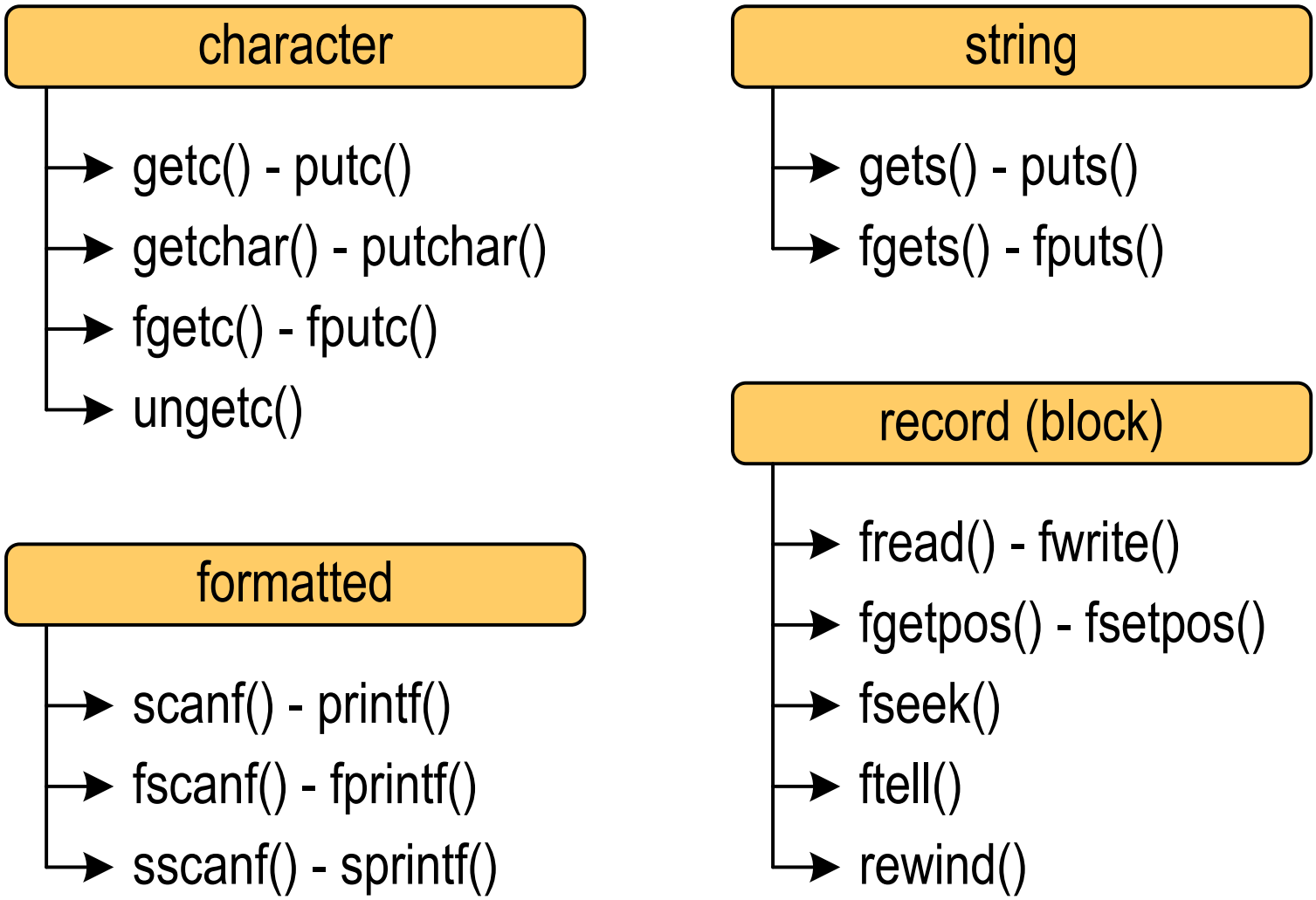
- Note:
  - `'x'` - the character constant (one character)
  - `"x"` - the string constant (two characters: x and \0)

Character constant

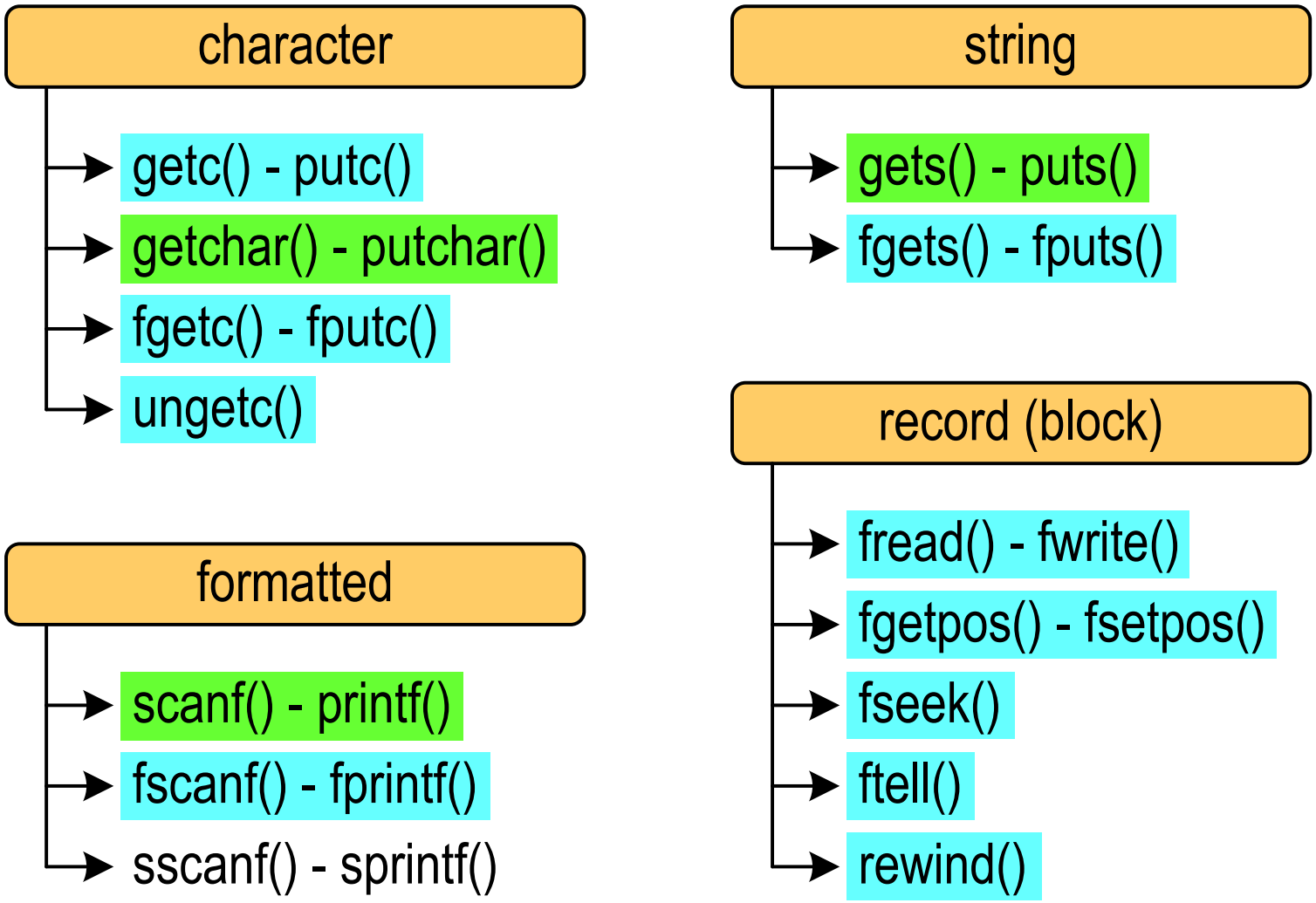
- Some characters can be represented in character constants by escape sequences that appear as two characters but actually represent only one character

`'\n'` - new line	`'\\'` - \ (backslash)
`'\t'` - horizontal tab	`'\''` - apostrophe
`'\v'` - vertical tab	`'\"'` - quotation marks
`'\a'` - alert	`'\?'` - question mark

Standard I/O functions



Standard I/O functions



String printing

- Printing string with the printf() function requires the %s specifier

```
char txt[15] = "John Smith";
printf("Person: [%s]\n", txt);
```

```
Person: [John Smith]
```

- In the %s specifier, the width specifies the field width, and precision specifies max. number of characters to be printed from the string

```
char txt[15] = "John Smith";
printf("[%10.6s]\n", txt);
```

```
[      John S]
```

String printing

- The puts() function can be used to print text

puts()	`int puts(const char *s);`

- The puts() function writes every character from the null-terminated ('\0') string s to the output stream stdout (screen), replacing the '\0' character with '\n' (newline)

```
char txt[15] = "John Smith";
puts(txt);
```

```
John Smith
```

Character constant printing

- The printf() requires the %c specifier to print a character

```
char ch = 'x';
printf("Character: [%c]\n", ch);
```

```
Character: [x]
```

- The putchar() function can also be used to print a character

putchar()	int putchar(int ch);

```
putchar('C'); putchar(97); putchar(0x74);
```

```
Cat
```

String printing

- A string is essentially an array, allowing us to refer to its individual elements

```
char txt[18] = "Kate has a laptop";

printf("Characters: ");
for (int i=0; i<18; i++) printf("%c ",txt[i]);
```

```
Characters: K a t e   h a s   a   l a p t o p
```

```
printf("Codes: ");
for (int i=0; i<18; i++) printf("%d ",txt[i]);
```

```
Codes: 75 97 116 101 32 104 97 115 32 97 32 108 97 112 116 111 112 0
```

String reading

- The %s specifier is used to read string with the scanf() function

```
char txt[15];
scanf("%s", txt);
```

no & operator

- We can specify a width in the %s format specifier

```
char napis[15];
scanf("%10s", napis);
```

- In the example above, scanf() will terminate reading the text after encountering the first whitespace (space, tab, or newline) or after reading 10 characters

String reading

- In case the text "This is a text" is entered, the scanf() function will read only the word "This"
- To read and store the entire line of text (until Enter is pressed), the gets() function is used

gets()	char *gets(char *s);

- The gets() function reads characters from the standard input (stdin) and stores them into the s array until a newline character ('\n') is encountered, the newline character ('\n') is replaced with '\0'

```
char txt[15];
gets(txt);
```

Character constant reading

- Reading one character with the scanf() function requires the %c format specifier (the & operator must appear before the character variable)

```
int ch;
scanf("%c",&ch);
```

- The getchar() function can also be used to read a character

getchar()	int getchar(void);

```
int ch;
ch = getchar();
```

Header file string.h

Function	Description
strlen(str)	computes the length of the string str, excluding the terminating null character ('\0')
strcpy(dest, src)	copies the string src to the string dest
strcat(dest, src)	appends a copy of the string src to the string dest
strcmp(str1, str2)	compares the strings str1 and str2 in a case-sensitive manner
strcmpi(str1, str2)	compares the strings str1 and str2 in a case-insensitive manner
strchr(str, str2)	searches for the first occurrence of the character c in the string str

Header file string.h

Function	Description
strlwr(str)	converts uppercase letters to lowercase in the string str
strupr(str)	converts lowercase letters to uppercase in the string str
strrev(str)	reverses the order of characters in the string str

- The above functions are not standard and may not be available in some compilers

Header file string.h: example

```c
#include <stdio.h>
#include <string.h>

int main(void)
{
    char txt1[] = "Text in the buffer", txt2[20];

    printf("txt1: %s \n",txt1);
    int len = strlen(txt1);
    printf("Number of characters in txt1: %d \n",len);
    strcpy(txt2,txt1);
    printf("txt2: %s \n",txt2);
    strrev(txt2);
    printf("txt2 (rev): %s \n",txt2);

    return 0;
}
```

Header file string.h: example

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

int main(void)
{
    char txt1[] = "T
    printf("txt1: %s \n",txt1);
    int len = strlen(txt1);
    printf("Number of characters in txt1: %d \n",len);
    strcpy(txt2,txt1);
    printf("txt2: %s \n",txt2);
    strrev(txt2);
    printf("txt2 (rev): %s \n",txt2);

    return 0;
}
```

```
txt1: Text in the buffer
Number of characters in txt1: 18
txt2: Text in the buffer
txt2 (rev): reffub eht ni txeT
```

Header file ctype.h

Function	Description
isalnum(chr)	returns non zero value if the passed argument chr is alphanumeric character
isalpha(chr)	returns non zero value if the passed argument chr is an alphabet
isblank(chr)	returns non zero value if the passed argument chr is a blank space
isdigit(chr)	returns non zero value if the passed argument chr is a number
isprint(chr)	returns non zero value if the passed argument chr is a printable character
isspace(chr)	returns non zero value if the passed argument chr is a white-space character

Header file ctype.h

Function	Description
islower(chr)	returns non zero value if the passed argument chr is a lowercase alphabet
isupper(chr)	returns non zero value if the passed argument chr is an uppercase alphabet
tolower(chr)	returns lowercase alphabet of the corresponding uppercase alphabet
toupper(chr)	returns uppercase alphabet of the corresponding lowercase alphabet

Header file ctype.h: example

```c
#include <stdio.h>
#include <ctype.h>

int main(void)
{
    int chr;

    printf("Enter character: ");
    chr = getchar();

    if (isalnum(chr)!=0) printf("%c - alphanumeric\n",chr);
    if (isalpha(chr)!=0) printf("%c - an alphabet\n",chr);
    if (isdigit(chr)!=0) printf("%c - a number\n",chr);
    if (isprint(chr)!=0) printf("%c - a printable\n",chr);

    return 0;
}
```

Header file ctype.h: example

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

int main(void)
{
    int chr;

    printf("Enter character: ");
    chr = getchar();

    if (isalnum(chr)!=0) printf("%c -
    if (isalpha(chr)!=0) printf("%c -
    if (isdigit(chr)!=0) printf("%c -
    if (isprint(chr)!=0) printf("%c -

    return 0;
}
```

```
Enter character: A
A - alphanumeric
A - an alphabet
A - a printable
```

```
Enter character: 7
7 - alphanumeric
7 - a number
7 - a printable
```

```
Enter character: %
% - a printable
```

End of workshop no. 08

Thank you for your attention!