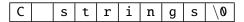
Strings

Pedro Ribeiro

DCC/FCUP

2024/2025



(based and/or partially inspired by Pedro Vasconcelos's slides for Imperative Programming)

Strings

- We can use character arrays to represent text strings
- C strings are terminated with a zero code character (represented as '\0')
 - ► the string "Abba" is A b b a \0
 - ► the empty string "" is \0
- The '\0' character is a terminator that marks the end of the string
 - is not part of the text
- The length of the string is determined by the position of '\0'
- Like other arrays, we must declare strings with a size. As explained before, using a macro can help making modifications easier:

```
#define MAX_SIZE 100
...
char text[MAX_SIZE]; // space for 99 characters + '\0'
```

Initializing strings

Initialize using character constants:

```
char text[6] = { 'H','e','l','l','o','\0' };
```

• We can also initialize with the characters in quotes:

```
char text[6] = "Hello";
```

- In this second form, the terminator '\0' is automatically inserted at the end.
- We can declare a longer length than necessary:

```
char text[100] = "Hello"; // remaining characters are \0
```

• If we omit the size, the compiler reserves only what is necessary:

```
char text[] = "Hello"; // size 6
```

Printing and reading strings

- To print a string we can use:
 - printf with the format %s
 - ▶ or the puts function (prints string followed by a newline '\n')

```
#include <stdio.h>
char text[] = "Hello, world";
printf("%s\n", text); // equivalent to puts
puts(text);
```

- To read a string we can use:
 - scanf with the format %s (read a string separated by whitespaces)
 - ▶ the fgets function (reads a string of at most *k* chars from a stream)

(more on the differences between scanf and fgets on the next slides)

Reading strings - scanf

• Imagine you have a file input.txt with the following content:

```
hello world programming
```

• The following program would read three separated strings (scanf uses *whitespace*, such as spaces or newlines, to separate *tokens*):

```
#include <stdio.h>

#define MAX 100

int main(void) {
  char s1[MAX], s2[MAX], s3[MAX];

  scanf("%s %s %s", s1, s2, s3);
  printf("[%s] [%s] [%s]\n", s1, s2, s3);

  return 0;
}
```

```
$ g++ -Wall -o code code.c && ./code < input.txt
[hello] [world] [programming]</pre>
```

Reading strings - scanf and fgets

- scanf with %s can be *dangerous* if the input has more characters the size of the string it will write to memory spaces it shouldn't
 - ► This is known as a buffer overflow
- To solve this we can use the fgets function:

```
#include <stdio.h>
#define MAX_SIZE 100
...
char text[MAX_SIZE];
fgets(text, MAX_SIZE, stdin);
```

- ► The second argument specifies the maximum size (it stops reading if the inputs has more characters)
- ► The third argument specifies the input channel (stdin is standard input)
- fgets will read everything (including spaces) until MAX_SIZE , newline or EOF is reached
- ▶ If there is no more input, it return NULL

Reading strings - scanf and fgets

• With the same input.txt file as before:

```
hello world programming
```

• Using fgets we get:

```
#include <stdio.h>

#define MAX 100

int main(void) {
  char s[MAX];

while (fgets(s, MAX, stdin) != NULL) {
  printf("%s", s); // s contains the newline
}

return 0;
}
```

```
$ g++ -Wall -o code code.c && ./code < input.txt
hello world
programming</pre>
```

String Library

- C has a string.h library with many string functions
 - You can check the documentation page
- Here are two of the most common functions:
 - strlen(s) return the length of string s
 - strcmp(s1, s2) returns:
 - ★ Negative value if s1 appears before s2 in lexicographical order
 - ★ Zero if s2 and cverb—s2— compare equal
 - ★ Positive value if cverb—s1— appears after s2 in lexicographical order

```
char s1[MAX] = "Pedro", s2[MAX] = "Ribeiro";
int len1 = strlen(s1), len2 = strlen(s2);

printf("%s has length %d\n", s1, len1);
printf("%s has length %d\n", s2, len2);
printf("strcmp(%s,%s) = %d\n", s1, s2, strcmp(s1, s2));
printf("strcmp(%s,%s) = %d\n", s2, s1, strcmp(s2, s1));
```

```
Pedro has length 5
Ribeiro has length 7
strcmp(Pedro, Ribeiro) = -2
strcmp(Ribeiro, Pedro) = 2
```