KU LEUVEN

C: an introduction

Strings: basics

char

- C supports the **char data type** for storing a single character.
- char uses one byte of memory, encoded as numbers using the ASCII scheme.
- char constants are enclosed in **single quotes** `char myGrade = 'B';`
- Use %c in printf( ) to print a single character.
- using %c with scanf( ) to input a single character.

KU LEUVEN

Special characters

- \ is used to indicate that the char that follows has special meaning.
  - \n is the newline character
  - \t is the tab character
  - \” is the double quote
  - \’ is the single quote
  - \\ is the backslash

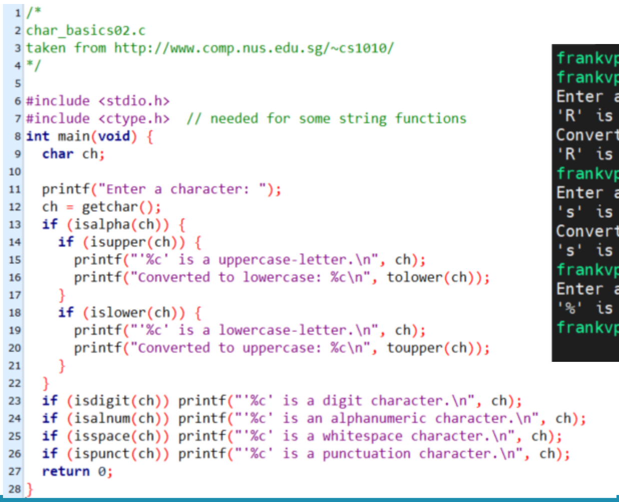
KU LEUVEN

Character library ctype

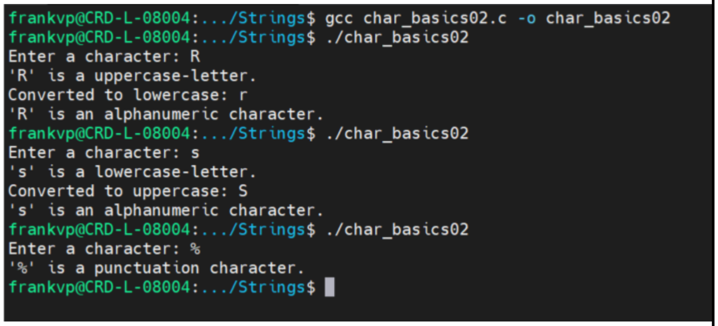
- `#include <ctype.h>`

- `int isdigit (int c);`
  - Determine if c is a decimal digit (‘0’ - ‘9’)
- `int isalpha (int c);`
  - Determines if c is an alphabetic character (‘a’ - ‘z’ or ‘A- ‘Z’)
- `int isspace (int c);`
  - Determines if c is a whitespace character (space, tab)
- `int tolower (int c);`
  - Returns c changed to lower-case
- `int toupper (int c);`
  - Returns c changed to upper-case

KU LEUVEN



```c
/*
char_basics02.c
taken from http://www.comp.nus.edu.sg/~cs1010/
*/

#include <stdio.h>
#include <ctype.h>  // needed for some string functions
int main(void) {
  char ch;

  printf("Enter a character: ");
  ch = getchar();
  if (isalpha(ch)) {
    if (isupper(ch)) {
      printf("'%c' is a uppercase-letter.\n", ch);
      printf("Converted to lowercase: %c\n", tolower(ch));
    }
    if (islower(ch)) {
      printf("'%c' is a lowercase-letter.\n", ch);
      printf("Converted to uppercase: %c\n", toupper(ch));
    }
  }
  if (isdigit(ch)) printf("'%c' is a digit character.\n", ch);
  if (isalnum(ch)) printf("'%c' is an alphanumeric character.\n", ch);
  if (isspace(ch)) printf("'%c' is a whitespace character.\n", ch);
  if (ispunct(ch)) printf("'%c' is a punctuation character.\n", ch);
  return 0;
}
```



```
frankvp@CRD-L-08004:.../Strings$ gcc char_basics02.c -o char_basics02
frankvp@CRD-L-08004:.../Strings$ ./char_basics02
Enter a character: R
'R' is a uppercase-letter.
Converted to lowercase: r
'R' is an alphanumeric character.
frankvp@CRD-L-08004:.../Strings$ ./char_basics02
Enter a character: s
's' is a lowercase-letter.
Converted to uppercase: S
's' is an alphanumeric character.
frankvp@CRD-L-08004:.../Strings$ ./char_basics02
Enter a character: %
'%' is a punctuation character.
frankvp@CRD-L-08004:.../Strings$ 
```

Strings

- C has no string handling facilities built in; strings are defined as arrays of characters.
- Strings are null-terminated (\0’) arrays of characters (*delimited strings*).
- Constant character strings are written inside **double-quotation marks “**
  - Single character variables are declared using single-quotation marks ’
- A string literal is a sequence of characters enclosed within double quotes
- Since a string literal is stored as an array, the compiler treats it as a pointer of type `char *`.
- Use %s in printf( ) to print a string.

Strings

- Arrays in C are non-assignable and non-copy-initializable. That's just how arrays are in C. Historically, in value context (on the RHS of assignment) arrays decay to pointers, which is what formally prevents assignment and copy-initialization. This applies to all arrays, not only to char arrays. (https://stackoverflow.com/questions/6901090/c-why-is-strcpy-necessary)

```
char s[4];
s = "abc"; //Fails
strcpy(s, "abc"); //Succeeds
```

- See also: https://www.geeksforgeeks.org/storage-for-strings-in-c/

KU LEUVEN

Strings

- Compare:
  - `char date[] = “April 1";`
  - declares date to be an array, characters can be modified
  - `char * date = “April 1";`
  - declares date to be a pointer to a string literal, should not be modified. This creates an unnamed character array just large enough to hold the string (including the null character) and places the address of the first element of the array in the char pointer

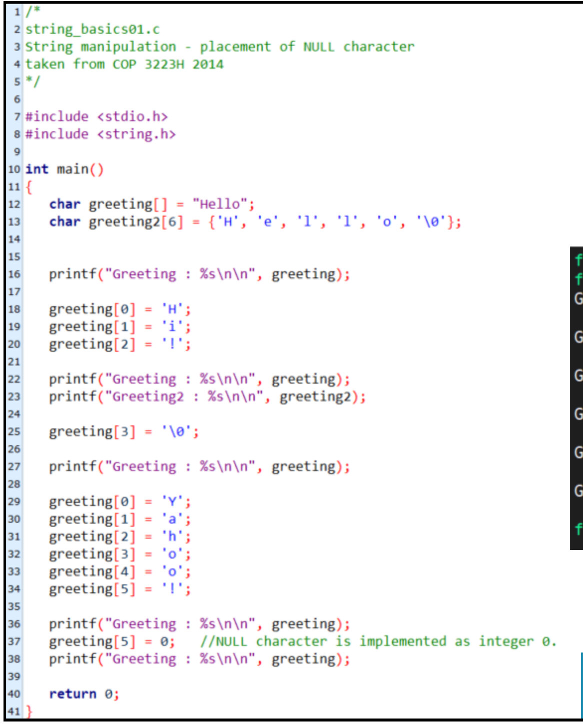
KU LEUVEN

Initializing character strings

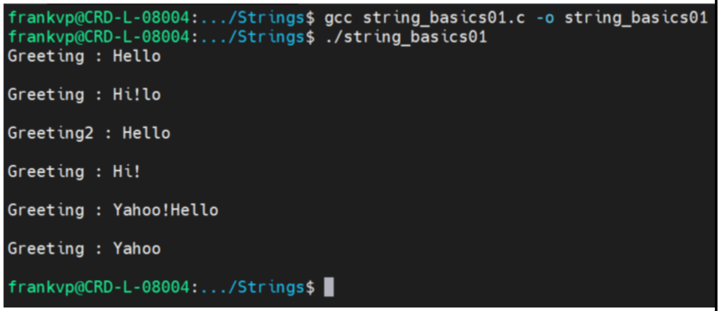
- Initializing a string:
  - `char word[] = "Hello!";`

  equivalent with:
  - `char word[] = { 'H', 'e', 'l', 'l', 'o', '!', '\0' };`
- The null string: A character string that contains no characters other than the null character
  - `char empty[]= "";`
  - `char buf[100]= "";`
- Initializing a very long string over several lines:
  - `char letters[] = { "abcdefghijklmnopqrstuvwxyz\`
    `ABCDEFGHIJKLMNOPQRSTUVWXYZ" };`

KU LEUVEN



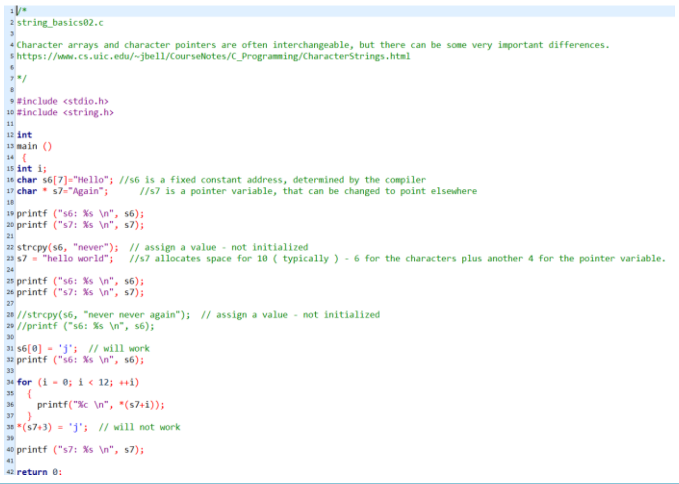
```c
/*
string_basics01.c
String manipulation - placement of NULL character
taken from COP 3223H 2014
*/

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

int main()
{
   char greeting[] = "Hello";
   char greeting2[6] = {'H', 'e', 'l', 'l', 'o', '\0'};


   printf("Greeting : %s\n\n", greeting);

   greeting[0] = 'H';
   greeting[1] = 'i';
   greeting[2] = '!';

   printf("Greeting : %s\n\n", greeting);
   printf("Greeting2 : %s\n\n", greeting2);

   greeting[3] = '\0';

   printf("Greeting : %s\n\n", greeting);

   greeting[0] = 'Y';
   greeting[1] = 'a';
   greeting[2] = 'h';
   greeting[3] = 'o';
   greeting[4] = 'o';
   greeting[5] = '!';

   printf("Greeting : %s\n\n", greeting);
   greeting[5] = 0;    //NULL character is implemented as integer 0.
   printf("Greeting : %s\n\n", greeting);

   return 0;
}
```



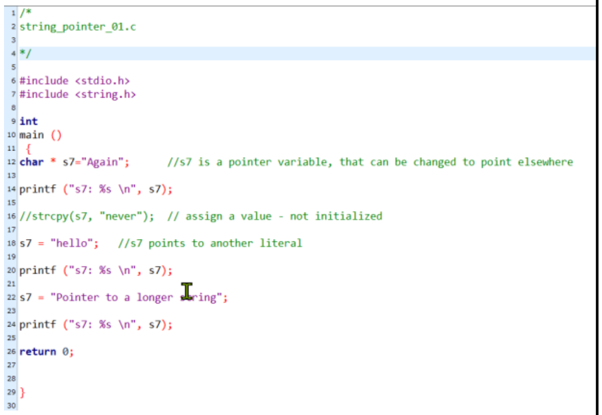
```
frankvp@CRD-L-08004:.../Strings$ gcc string_basics01.c -o string_basics01
frankvp@CRD-L-08004:.../Strings$ ./string_basics01
Greeting : Hello

Greeting : Hi!lo

Greeting2 : Hello

Greeting : Hi!

Greeting : Yahoo!Hello

Greeting : Yahoo

frankvp@CRD-L-08004:.../Strings$
```

KU LEUVEN

Strings



```c
/*
string_basics02.c

Character arrays and character pointers are often interchangeable, but there can be some very important differences.
https://www.cs.uic.edu/~jbell/CourseNotes/C_Programming/CharacterStrings.html

*/

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

int
main ()
 {
int i;
char s6[7]="Hello"; //s6 is a fixed constant address, determined by the compiler
char * s7="Again";      //s7 is a pointer variable, that can be changed to point elsewhere

printf ("s6: %s \n", s6);
printf ("s7: %s \n", s7);

strcpy(s6, "never");  // assign a value - not initialized
s7 = "hello world";   //s7 allocates space for 10 ( typically ) - 6 for the characters plus another 4 for the pointer variable.

printf ("s6: %s \n", s6);
printf ("s7: %s \n", s7);

//strcpy(s6, "never never again");  // assign a value - not initialized
//printf ("s6: %s \n", s6);

s6[0] = 'j';  // will work
printf ("s6: %s \n", s6);

for (i = 0; i < 12; ++i)
  {
    printf("%c \n", *(s7+i));
  }
*(s7+3) = 'j';  // will not work

printf ("s7: %s \n", s7);

return 0;
```



```c
/*
string_pointer_01.c

*/

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

int
main ()
 {
char * s7="Again";      //s7 is a pointer variable, that can be changed to point elsewhere

printf ("s7: %s \n", s7);

//strcpy(s7, "never");  // assign a value - not initialized

s7 = "hello";   //s7 points to another literal

printf ("s7: %s \n", s7);

s7 = "Pointer to a longer string";

printf ("s7: %s \n", s7);

return 0;


}
```

C string library

- The C library supplies several string-handling functions. To use the string functions, include **`<string.h>`**
- String handling via the C standard library: https://en.wikipedia.org/wiki/C_string_handling
- Commonly used functions.
  - **`strcat`**
  - **`strlen`**
  - **`strcpy`**
  - **`strcmp`**
- *File: string_strlen.c*
- *File: string_strcmp.c*
- *File: string_manip.c*

String functions

- `strcpy(s1, s2)` – Copies the string s2 to s1
  - `s1 = s2` assignment is not working
- `strcat(s1, s2)` – Concatenates string s2 to the end of s1, putting \0 at the end.
- `strcmp(s1, s2)` – Compares strings s1 and s2 and returns a value:
  - less than zero if s1 < s2,
  - equal to zero if s1 == s2,
  - greater than zero if s1 > s2.
- `strlen(s)` – Returns the number of characters in s, excluding \0

String to number conversion

- A number can be stored in a numeric form or as character string array
  - `char numb[] = “123”;`
  - `int numb = 123;`
- `<stlib.h>` needed
  - atoi(s) converts string s to a type int value and returns it. The function converts characters until it encounters something that is not part of an integer.
  - atof() converts a string to a floating point number
- *File: string_ato.c*