Characters

- Built-in data type
- Value: a single character
- Literals: ‘a’, ‘!’, ‘\n’, ‘8’, ...
- Operations:
  - assignment: =
  - compare: ==, <, etc.

```c
char ch;
ch = 'a';
if (ch=='A') ...
```

Character Testing

10.1 Character Testing

- The C++ library provides several functions for testing characters.
- Requires the cctype header file
- These functions have this signature:
  - int isupper (int c);
- They take a char (or int) argument
- They return non-zero for true, 0 for false.

```c
char input;
...
if (isupper(input)) ...
```

```c
char ch;
ch = 'a';
if (ch=='A') ...;
cin >> ch;
```
10.2 Character Case conversion

- `int toupper (int c)`
  - converts lowercase letters to uppercase
  - otherwise returns c
- `int tolower (int c)`
  - converts uppercase letters to lowercase
  - otherwise returns c
- Does NOT change argument

```
char x = 'A';
char y = tolower(x);
cout << x << " " << y << endl;
```

Output: A a

10.3 C-Strings

- In any programming language, a “string” is a sequence of characters.
- In C++, a C-String is a certain way of representing a string in memory
- A C-String is:
  - a sequence of characters
  - stored in consecutive memory locations
  - ALWAYS terminated by a null character (`\0`, ascii=0)

```
Hi There!
```

C-String

- String literals are stored in memory as C-Strings:
  - “Jim Kase”, “A00123456”, “$2.35/lb”
  - they have type `char[]`

- A C-String can be stored in a char array.
  - Make sure array is large enough for the null char!
- Do NOT pass size to functions taking C-strings as arguments
  - Unnecessary, because the null char marks the end.

```
char cstr[10];
cout << "Enter a name: ";
cin.getline(cstr,10);
cout << "You entered: " << cstr << endl;
```

Enter a name: Tom Fox
You entered: Tom Fox

Operations over C-Strings

- Don’t use = or == on char[] (arrays: doesn’t work)
- input: can use `>>`
  - input stops at whitespace (space, tab, newline)!
  - copying to memory does NOT stop at end of array
- input: can use `cin.getline(char s[], int n)`
  - input stops at ‘\n’ OR after n-1 characters have been read
- output: can use `<<`

```
char cstr[10];
cout << "Enter a name: ";
cin.getline(cstr,10);
cout << "You entered: " << cstr << endl;
```

Enter a name: Tom Johnson
You entered: Tom Jôhns
### 10.4 Library Functions for C-Strings

- Usually require the cstring header
- Function headers look like this: `func(char *s)`
  - `char *s` is similar to `char s[]`
- the argument can be:
  - the name of a char array
  - a literal string (sometimes)

### C-string length

- `int strlen (char* str)`
- Returns the number of characters in a C-string (up to but not including the null char).

```c
char cstr[30] = "Economics";
cout << strlen(cstr) << endl; //prints 9
```

### C-string copy

- Use `strcpy` to perform assignment for C-strings
  ```c
  char* strcpy (char *destination, char *source);
  ```
- Copies source C-string to destination
  - destination is modified
  - destination must be long enough
  - ignore returned value
- example:

```c
char string1[13] = "Hello ";
char string2[7] = "World!";
//simulate: string1 = string2;
strcpy(string1, string2);
cout << string1 << endl;
```

Output:
```
World!
```

### C-string compare

- Use `strcmp` to perform comparison for C-strings
  ```c
  int strcmp (char *str1, char *str2);
  ```
- Compares `str1` and `str2`
  - if `str1` and `str2` are the same, return 0
  - if `str2` comes after `str1` alphabetically, return -1
  - if `str2` comes before `str1` alphabetically, return 1
- example:

```c
char string1[13] = "Hello ";
char string2[7] = "World!";
// if (string1<string2)...
if (strcmp(string1, string2) < 0)
cout << "Negative" << endl;
```

Output:
```
Negative
```
10.7 More about the C++ string class

- string is a data type provided by the C++ library.
  - Specifically it is a class.
- string requires the <string> header file
  - <iostream> may work as well
- To define a string variable:
  - string name1;
  - name1 is a string object.
- The representation in memory of a string object is hidden from the programmer.

Operations over string objects

- initialization using = with a C-String literal
  ```cpp
  string namel = "Steve Jobs";
  // can do this with char arrays too:
  char name2[20] = "Steve Jobs";
  ```
- assignment using =
  ```cpp
  string name1, name2;
  cout << "Enter a name: ";
  cin >> name1;
  name2 = name1; // can’t do with char arrays
  ```
- assignment of C-Strings to string objects:
  ```cpp
  string namel;
  namel = "Andre Johnson";
  ```

Operations over string objects

- output using <<
  ```cpp
  string namel;
  namel = "Steve Jobs";
  cout << "Name " << namel << endl;
  ```
- input using >>
  ```cpp
  string namel;
  cout << "Enter your name ";
  cin >> namel;
  ```
- input using getline
  ```cpp
  string namel;
  cout << "Enter your name ";
  getline (cin, namel);
  ```

Operations over string objects

- comparing string objects: < <= > >= == != (alphabetical order)
  ```cpp
  string string1, string2;
  string1 = "Hello ";
  string2 = "World!";
  if (string1 < string2)
    cout << "Hello comes before World" << endl;
  ```
- string objects can be compared to C-strings
  ```cpp
  string string1;
  cout << "Enter a word: ";
  cin >> string1;
  if (string1 == "Hello")
    cout << "You entered Hello." << endl;
  ```
More operations over string objects

- [n] subscript notation, returns char at position n
- or use string.at(n)--performs bounds check

```cpp
string string1 = "Hello ";
cout << string1[4] << endl;
cout << string1.at(1) << endl;
```

Output:
```
0 e
```

```cpp
string1[0] = 'h'; //this works
string1[6] = 's'; //this gets ignored (6>=length)
string1.at(6) = 's'; //this causes an error
```

string class member functions

- string class has many member functions that operate over the string object

- theString.length() : returns length of string stored in theString (can also use .size()).

```cpp
string theString = "Hello";
cout << theString.length() << endl;  //outputs 5
```

- theString.append(str): appends str (string object or c-string) to the end of theString.

```cpp
string theString = "Hello";
theString.append(" World");
cout << theString << endl;  //outputs: Hello World
```

more string class member functions

- theString.append(n, 'z') : appends n copies of char to end of string

```cpp
string theString = "Hello ";
theString.append(2,’z’);
cout << theString << endl;  //outputs: Hello zz
```

- theString.substr(x,n): returns a new string, copies n chars starting at position x from theString.

```cpp
string string1 = "hello there";
cout << string1.substr(6,1) << endl;  //outputs: the
```

Exercise

- Write a function countDigits that takes a string as an argument and outputs the number of digits it contains.

```cpp
int countDigits (string p) {
    int count = 0;
    for (int i=0; i < p.length(); i++) {
        if (isdigit(p[i]))
            count++;
    }
    return count;
}
```