3 Character I/O

- All data is input and output as characters
  - Output of the number 10 is two characters '1' and '0'
  - Input of the number 10 is also done as '1' and '0'
  - Interpretation of 10 as the number 10 or as characters depends on the program
  - Conversion between characters and numbers is usually automatic

3.1 Low Level Character I/O

- Low level C++ functions for character I/O
  - Perform character input and output
  - Do not perform automatic conversions
  - Allow you to do input and output in anyway you can devise

3.2 Member Function get

- Function get
  - Member function of every input stream
  - Reads one character from an input stream
  - Stores the character read in a variable of type char,the single argument the function takes
  - Does not use the extraction operator (>>) which performs some automatic work
  - Does not skip blanks

3.2.1 Using get

- These lines use get to read a character and store it in the variable next_symbol

```
char next_symbol;
cin.get(next_symbol);
```

  - Any character will be read with these statements
    - Blank spaces too!
    - '\n' too! (The newline character)

3.2.2 get Syntax

- **input_stream.get(char_variable);**
- Examples:

```
char next_symbol;
cin.get(next_symbol);
ifstream in_stream;
in_stream.open("infile.dat");
in_stream.get(next_symbol);
```

3.2.3 More About get

- Given this code:
```
char c1, c2, c3;
cin.get(c1);
cin.get(c2);
cin.get(c3);
```
and this input:

AB
CD

- c1 = 'A' c2 = 'B' c3 = '\n'
  - **cin >> c1 >> c2 >> c3;** would place 'C' in c3(the ">>" operator skips the newline character)

3.2.4 The End of The Line

- To read and echo a line of input
  - Look for '\n' at the end of the input line:
```
cout << "Enter a line of input and I will "
        << "echo it.\n"; char symbol;
do
{
        cin.get(symbol);
        cout << symbol;
} while (symbol != '\n');
```
  - All characters, including '\n' will be output
  - **'\n'**
    - A value of type char
    - Can be stored in a variable of type char
  - **"\n"**
    - A string containing only one character
    - Cannot be stored in a variable of type char
  - In a cout-statement they produce the same result

3.3 Member Function Peek

- Returns the next character in the input sequence, without extracting it:
  - The character is left as the next character to be extracted from the stream.

3.3.1 sample16.cpp

```
#include <iostream>

int main (int argc, char *argv[]) {
    std::cout << "Type in :";
    char string[256]; // char array!

    while(std::cin.peek() != '\n'){
        std::cin >> string;
        std::cout << string << std::endl;
    }
    return 0;
}
```

3.4 Member Function put

- Function put
  - Member function of every output stream
  - Requires one argument of type char
  - Places its argument of type char in the output stream
  - Does not do allow you to do more than previous output with the insertion operator and **cout**

3.4.1 put Syntax

- **Output_stream.put(Char_expression);**
- Examples:

```
cout.put(next_symbol);
cout.put('a');
ofstream out_stream;
out_stream.open("outfile.dat");
out_stream.put('Z');
```

3.5 Member Function putback

- The **putback** member function places a character in the **input** stream
  - **putback** is a member function of every input stream
  - Useful when input continues until a specific character is read, but you do not want to process the character
  - Places its argument of type char in the input stream
  - Character placed in the stream does not have to be a character read from the stream

3.5.1 putback Example

- The following code reads up to the first blank in the input stream fin, and writes the characters to the file connected to the output stream **fout**

```
fin.get(next);
while (next != ' ')
{
        fout.put(next);
        fin.get(next);
}
fin.putback(next);
```

  - The blank space read to end the loop is put back into the input stream

3.6 Program Example Checking Input

- Incorrect input can produce worthless output
- Use input functions that allow the user to re-enter input until it is correct, such as
  - Echoing the input and asking the user if it is correct
  - If the input is not correct, allow the user to enter the data again

```cpp
#include <iostream>

int main(int argc, const char * argv[]) {
    int x;
    std::cout << "Input a whole number :";
    while(!(std::cin >> x)){
        std::cout << "Try again :";
        // Clearing the error flags which are set when std::cin fails to interpret the input
        std::cin.clear();
        // Removing the input contents that could caused the read failure.
        // (Remove 1000 charactors until '\n')
        std::cin.ignore(1000,'\n');
    }
    std::cout << "Number you input :" << x << std::endl;
    return 0;
}
```

3.7 Character Functions

- Several predefined functions exist to facilitate working with characters
- The **cctype** library is required

```cpp
#include <cctype>
using namespace std;
```

3.7.1 The toupper Function

- **toupper** returns the argument's upper case character
  - toupper('a') returns 'A'
  - toupper('A') return 'A'

toupper Returns an int

- Characters are actually stored as an integer assigned to the character
- **toupper** and **tolower** actually return the integer representing the character
  - ```cpp
    cout << toupper('a'); //prints the integer for 'A'
    ```
  - ```cpp
    char c = toupper('a'); //places the integer for 'A' in c
    cout << c; //prints 'A'
    ```
  - ```cpp
    cout << static_cast<char>(toupper('a')); //works too
    ```

3.7.2 The isspace Function

- **isspace** returns true if the argument is whitespace
  - Whitespace is spaces, tabs, and newlines
  - **isspace**(' ') returns **true**
  - Example:

```cpp
if (isspace(next) )
        cout << '-';
else
        cout << next;
```

  - Prints a '-' if next contains a space, tab, or newline character
- See more character functions in
```