CS 246 Spring 2019 - Tutorial 3

May 29th, 2019

1 Summary

- C++-style strings
- Streams
- Parameters

2 Strings

In C++, there is a `std::string` type to replace C-style character arrays.

- `#include <string>`
- Common operations supported (some as a member function of `std::string`):
  - indexed access using `[]`
  - concatenation using `+, +=` (both with `std::string` and with C-style string)
  - comparison using `==, !=, <, >, <=, >=`
  - others: `length, clear, substr, find`
- Use `c_str()` member function to access a C-style version (`const char*` [1]) of the string.

3 Streams

In C++, streams are used to handle IO and files.

3.1 Input Streams

- An input stream is a stream which information can be read from.
- By default, reading from an input stream is whitespace delimited.
- Functions common to all input streams:
  - `eof()`: returns true if the stream has reached an end-of-file.

[1] i.e. you should not modify whatever this pointer points to. In fact, that is an undefined behavior.

- `fail()`: returns true if a read from the stream has failed, including reaching EOF.
- `clear()`: sets the failbit to false.
- `ignore()`: skips the next character in the stream.
- `std::istream >> std::string`: reads the next word from `std::istream` and stores it in `std::string` where `std::string` is the name of a variable of type std::string.
- `std::istream >> int`: reads the next int from `std::istream` and stores it in `int` where `int` is the name of a variable of type int. If the next characters in `std::istream` cannot be interpreted as an int, the failbit is set to true.
- Similar functions exist for all built in C++ types, e.g. bools, chars, floats, etc.

3.2 Output Streams

- An output stream is a stream into which information can be placed.
- Functions common to all output streams:
  - `std::ostream << var`: puts the information stored in `var` in `std::ostream`. This function exists for all built in C++ types.

3.3 IO Streams

- `#include <iostream>`
- Includes `std::cin` (stdin), `std::cout` (stdout), and `std::cerr` (stderr).
- As previously described, these are the three streams which all programs have. Input and output can be redirected to and from these streams.

3.4 File Streams

- `#include <fstream>`
  - `std::ofstream`: file stream only for output
  - `std::ifstream`: file stream only for input
- For example, to open a file for reading:
  - `std::ifstream file{filename};` [2]
- By default, opening an ofstream to a file which exists overwrites the data in the file. If the file doesn't exist, it is created.

[2] In C++03 and earlier version, filename needs to be a C-style string. Also, in C++03 and earlier, you need to use a pair of round bracket instead of a pair of curly braces, i.e. `std::ifstream file("filename.txt");`

3.5 String Streams

- `#include <sstream>`
- String streams are streams into which formatted information can be stored in and from which a string matching the stored information can be obtained.
  - `std::ostringstream`: stringstream only for output
  - `std::istringstream`: stringstream only for input
- `str()`: obtain a C++ style string matching the information stored in a stringstream. Note: the string returned is temporary and will be removed from memory once returned; hence, the following expression will result in a dangling pointer:

```
// oss is an std::ostringstream
const char* p = oss.str().c_str();
```

4 Example: Converting String From And To Integer

- We now see a real life usage of `stringstream`.
- In C, there is a function that converts a (C-style) string to int (`int atoi(const char *str)`), and some compilers have a function that converts integer to C-style string (although that is not in the C/C++ standard).
- Note that in CS246 `atoi()` is forbidden, for `<cstdlib>` is not allowed to be included in the headers. How do we achieve integer/string conversion in C++?
- Turns out that we can use `stringstream`s!
- To convert a `std::string` to a `int`:

```
std::string s = "42 -42";
std::istringstream iss{s};
int x,y;
iss >> x >> y;
```

- To convert a `int` to a `std::string` [3]:

```
std::ostringstream oss;
oss << 23 << "+" << 124;
std::string s = oss.str();
```

- You may also use `std::string::stoi()` and `to_string()` (in `std::string`) to convert integers and other data from and to strings. **Note: stoi() throws an exception when the conversion failed. Before you learn how do deal with exceptions, make sure the string you pass into the function is a valid digit (or use `stringstream`s).**

5 Parameters

Parameters are a list of arguments that a function expects when it is being called.

[3] Note that since this also applies for std::cout and std::cerr you don't need to convert integers to strings most of the time.

5.1 Overloading

- In C++, we can have multiple functions with the same name as long as the number of parameters or the types of parameters are different.

```
int foo(char c, int n);
int foo(int n);
```

- Note: functions cannot be overloaded based on return type alone.

5.2 Default Parameters

- The parameters of a function can be given default values.

  For example,

```
void foo(int n = 75);
```

  There are now two ways to call foo:

```
void foo();
void foo(10);
```

  Using default parameters is equivalent to having two functions with the same body and different parameters (and it's a way to reduce code duplication).

- Any number of parameters to a function can have default parameters but if a parameter has a default value, all parameters to its right must have default values.

  Example:

```
void foo(int n = 75, char c); // not valid
void foo(int n = 75, char c = 'a'); // valid
```

- **Question:** Which of the following is not a valid overload of `bool foo(int x, char c);`?

  1. `int foo();`
  2. `char foo(char x, int c);`
  3. `bool foo(int c);`
  4. `int foo(int x, char c, int y = 10);`
  5. None of the above.

6 Vi Tip of the Week: Tabs

- When programming, it is often useful to have multiple files open at the same time. Multiple files can be opened with tab both when opening vi and when it is already opened.
- To open multiple tabs when opening vi, type multiple file names after vi followed by `-p`.
- To open additional files, enter command mode and type `:tabedit file`.
- When multiple tabs are open, enter `gt` to switch to the tab to the right of the current tab. `gT` switches to the tab to the left of the current tab.
- If multiple files are opened, all files can be closed and saved by entering `:wqa`.

7 Vi Tip of the Week: Matching Brackets

- `%`: when the cursor is on a bracket, pressing `%` moves the cursor to the matching bracket (in command mode).