Tutorial 2

NWEN241

Systems Programming: Input and Output

Kirita-Rose Escott

kirita-rose.escott@ecs.vuw.ac.nz

Content

- I/O in Standard C
  - #include <stdio.h>
  - Non-formatted vs. Formatted
- I/O in Standard C++
  - #include <iostream>
  - Streams

I/O in Standard C

- For standard input and output capability, C provides functions which can be accessed by including the **stdio.h** header file
- From the program's point of view, data input and output are made possible through files
- Every C program has access to 3 such files: **stdin, stdout, stderr**

File	Description	Remarks
stdin	Standard input file	Connected to the keyboard
stdout	Standard output file	Connected to the screen
stderr	Standard error file	Connected to the screen

I/O in Standard C: Non-formatted vs. Formatted

- Input and output functions can be classified into the following:
  - Non-formatted I/O:
    - **getchar**
    - **putchar**
    - **gets**
    - **puts**
  - Formatted I/O:
    - **printf** and its variants
    - **scanf** and its variants

Non-formatted I/O Examples

- getchar() / putchar()

```
char c;
c = getchar();          /* input a char */
putchar(c);             /* output a char */
```

- gets() / puts()

```
char line [80];
gets(line);             /* input a line/string */
puts(line);             /*outout a line */
```

Formatted I/O Example

```
int i;
float f;
char c;
char s[80];

scanf("%d", &i);                /* %d is format information, d is conversion character */

scanf("%f", &f);                /* &f is f's memory address, input is sent to &f */

printf("\nYou typed in \" %f\ "\n ", f);       /* \n start a new line, \ treats f as an ordinary character */

scanf(" %c", &c);               /* blank space preceding %c is to ignore the \n typed in earlier */

scanf("%s ", s);                /* a sequence of non-white space characters */
```

I/O in Standard C++

- For standard input and output capability, C provides functions which can be accessed by including the **iostream** class
- From the program's point of view, data input and output are made possible through streams
- C++ comes with four predefined standard stream objects that have been setup for your use

Stream	Description	Remarks
cin	A class tied to the standard input	Typically tied to the keyboard
cout	A class tied to the standard output	Typically tied to the monitor
cerr	A class tied to the standard error, providing unbuffered* output	Typically tied to the monitor
clog	A class tied to the standard error, providing buffered* output	Typically tied to the monitor

* Unbuffered output is typically handled immediately, where is buffered output is typically stored

Operators in Standard C++

- Scope resolution operator
  - Used access namespace members:
  - **std::cout << "Enter your age: " << std::endl;**

- Operators are used to insert and extract values from the streams
  - With **input** streams, the **extraction operator (>>)** is used to remove values from the stream
  - With **output** streams, the **insertion operator (<<)** is used to put values in the stream

Non-formatted I/O Examples

- cin/cout

```
char c;
std::cin >> c;                  /* input a char */
std::cout << c;                 /* output a char */
```

- get()/getline()

```
char strBuf[11];
std::cin.getline(strBuf, 11);           /* read up to 10 characters */

string strBuf;
getline(cin, strBuf);                   /* special getline method for string*/
```

Formatted I/O

- For output, there are two ways to change the formatting options: **flags**, and **manipulators**
  - Flags can be seen as boolean variables that can be turned on and off

```
std::cout.setf(std::showpos);          /* turn on the std::showpos flag */
std::cout << 27 << std::endl;
std::cout.unsetf(std::showpos);                /* turn off the std::showpos flag */
```

  - Manipulators:

```
std::cout << std::showpos << 27 << std::endl;  /* print 27 using the manipulator instead of flag */
```

Basic C++ Stream I/O Example

```
#include <iostream>
#include <cstdlib>       /* for exit()*/

int main()
{
    std::cout << "Enter your age: " << std::endl;              /* use the insertion operator on cout to print text to the monitor */

    int age;
    std::cin >> age;                                          /* use the extraction operator on cin to get input from the user   */

    if (age <= 0)    {
      std::cerr << "Oops, you entered an invalid age!" << std::endl; /* use the insertion operator on cerr to print an error message  */
      exit(1);
     }

    std::cout << "You entered " << age << " years old" << std::endl;          /* use insertion again on cout to print a result */

    return 0;
}
```