C / C++ Basics

Materials adapted from Dianna Xu, Walter Savitch, Jim Cohoon and Jack Davidson
Today’s Goals

• Short-hands, prefix and postfix
• `for` loops
• Arrays
• Arrays and `chars`
• `ctype.h`
• `stdlib.h`
• C++ strings
• C++ I/O streams
float y = 12.5;
short temperature = 32;
char letter = 'c';
short number;
Shorthand

```java
int i = 1, end = 100, val = 0;

while (i <= end) {
    val += i;
    i += 1;
}
```

val += i; ⇔ val = val + i;
Also for other operators
Further Shorthand

```java
int i = 1, end = 100, val = 0;

while (i <= end) {
    val += i;
    i++;
}
```

```
i++; ⇔ i = i + 1;
```
Shorthands: \texttt{op} =, \texttt{++i}, \texttt{i++}

- \texttt{+=}, \texttt{-=}, \texttt{ *=}, \texttt{ /=}, \texttt{ %=}

- Prefix form increments \texttt{i}'s value \texttt{before} it is referenced
  - \texttt{i} = 5;
  - \texttt{x} = (\texttt{++i}) + 6;

- Postfix form increments \texttt{i}'s value \texttt{after} it is referenced
  - \texttt{i} = 5;
  - \texttt{x} = (\texttt{i++}) + 6;
for Loops

```java
int i = 1, end = 100, val = 0;

while (i <= end) {
    val += i;
    i++;
}
```

```java
int i, end = 100, val = 0;

for (i = 1; i <= end; i++) {
    val += i;
}
```
for Loop

- Pattern

  ```
  for ( init; condition; update ) {
      body
  }
  ```

  - Each section can be blank.
  - Sequence: ①②③④...②③④② (cond fails)
break Statements

• Exit from a loop
• Typically used with an if statement
  (as in the previous page)

```java
while (cond) {
    break;
}
```
Example

```c
int i, val;

for(i=1, val=0; i<=100; i++) {
    if (val > 50)
        break;
    val += i;
}
```
continue  Statements

• Continue to the beginning of a loop
  □ I.e., the condition will be checked
• Typically used with an if statement

```java
while (cond) {
    continue;
}
```
Example

```c
int i, val;

for (i=1, val=0; i<=100; i++) {
    if (i > 20 && i < 30)
        continue;
    val += i;
}
```
Variations

- No braces

```c
int i;
for (i = 0; i < 10; i++)
    printf("%d\n",i);
```

- Omission of component(s)

```c
int i = 0;
for (; i < 10;)
    printf("!");
i++;
```

```c
for (;;) {
    printf("!");
    i++;
}
```
Nested for

```c
int i, j, end = 10;

for (i = 1; i <= end; i++) {
    for (j = 1; j <= i; j++) {
        printf("*"自觉);
    }
    printf("\n")自觉;
}
```

triangle.c
Arrays

- To store a large number of data of homogenous type (e.g. `int` only)

- Schematic representation

```
0 1 2  k-2  k-1
```

Index

Element

Flippo Brunelleschi
Ospdale degli Innocenti
Firenze, Italy, 1421
Array Operations

• Declaration
  ```
  int a[5];
  ```

• Assignment
  ```
  a[0] = 1;
  ```

• Reference
  ```
  int y = a[0];
  ```
int main() {
    int digits[10] = {0}, i; char c;

    while((c = getchar()) != EOF) {
        if (c >= '0' && c <= '9')
            digits[c-'0']++;
    }

    return 0;
}
C library containing a bunch of very useful character functions.

These functions take an integer (not necessarily a `char`) and return 0 or 1.

- `int isdigit(int c);`
- `isalpha, isalnum, isspace, islower, isupper`
- `int tolower/toupper (int c);`
• **void exit(int status);**
• Terminates a C program.
• Non-zero parameter values indicate program error to parent.
• A call to `exit(1)` is often used in conjunction with error detection.
C++: STREAMS AND STRINGS
• Files for I/O are the same type of files used to store programs

• A stream is a flow of data
  □ Input stream: Data flows into the program
    ◆ If input stream flows from keyboard, the program will accept data from the keyboard
    ◆ If input stream flows from a file, the program will accept data from the file
  □ Output stream: Data flows out of the program
    ◆ To the screen
    ◆ To a file
C++: cin & cout Streams

• cin
  ▮ Input stream connected to the keyboard
• cout
  ▮ Output stream connected to the screen
• cin and cout defined in the iostream library
  ▮ Use include directive: #include <iostream>
• You can declare your own streams to use with files.
C++ Strings

• Class string
  □ Used to represent a sequence of characters as a single object

• Some definitions

```cpp
string name = "Joanne";
string decimalPoint = ".";
string empty = "";
string copy = name;
string question = '?';  // illegal
```
Nonfundamental Types

• To access a library use a preprocessor directive to add its definitions to your program file
  
  ```
  #include <string>
  ```

• The *using* statement makes syntax less clumsy
  
  □ Without it
    ```
    std::string s = "Sharp";
    std::string t = "Spiffy";
    ```
  
  □ With it
    ```
    using namespace std; // std contains string
    string s = "Sharp";
    string t = "Spiffy";
    ```
Some string member functions

- `size()` determines number of characters in the string
  ```cpp
  string saying = "Rambling with Gambling";
  cout << saying.size() << endl; // 22
  ```

- `substr()` determines a substring (Note first position has index 0)
  ```cpp
  string word = saying.substr(9, 4); // with
  ```

- `find()` computes the position of a subsequence
  ```cpp
  int j = saying.find("it"); // 10
  int k = saying.find("its"); // ?
  ```
Class string

• Auxiliary functions and operators

  □ getline() extracts the next input line
    ```cpp
    string response;
    cout << "Enter text: ";
    getline(cin, response, '\n');
    cout << "Response is \"" << response
         << \"" << endl;
    ```

  □ Example run
    ```
    Enter text: Want what you do
    Response is "Want what you do"
    ```
Class string

• Auxiliary operators
  
  + string concatenation
  
  ```
  string part1 = "Me";
  string part2 = " and ";
  string part3 = "You";
  string all = part1 + part2 + part3;
  ```

  += compound concatenation assignment
  
  ```
  string thePlace = "Brooklyn";
  thePlace += ", NY";
  ```
```cpp
#include <iostream>
using namespace std;

int main() {
    string date, month, day, year, newDate;
    int i,k;
    cout << "Enter the date (e.g., January 1, 2001) : ";
    getline(cin, date, '\n');
    i = date.find(" ");
    month = date.substr(0, i);
    k = date.find(",");
    day = date.substr(i + 1, k - i - 1);
    year = date.substr(k + 2, date.size() - 1);
    newDate = day + " " + month + " " + year;
    cout << "Original date: " << date << endl;
    cout << "Converted date: " << newDate << endl;
    return 0;
}
```
Summary

• Be careful with prefix and postfix, especially postfix

• Loops are where a program spends most of its time. Learn to write efficient ones!

• Learn to use array and characters flexibly

• Learn to use strings and streams in C++