Strings

Victor Eijkhout, Harika Gurram,
Je'aime Powell, Charley Dey

Fall 2018

Characters

Characters and ints

- Type *char*;
- represents '7-bit ASCII': printable and (some) unprintable characters.
- Single quotes: `char c = 'a'`
- Equivalent to (short) integer: `'x'-'a'` is distance `a--x`

Exercise 1

Code:

```
int user_number;
string user_text{""};
vector<string> names{ "zero","one","two","three",
    "four","five","six","seven","eight","nine"};
cout << names.size() << "<"
     << names[0] << "-" << names[9] << ">" << endl;

cout << "Give a number: ";
cin >> user_number; cout << endl;

for (int d=0; user_number>0; d++) {
  int remember = user_number,digit;
  user_number = user_number/10;
  digit = remember-10*user_number;
  string name = names[digit];
  cout << "Digit " << d << " from the end: " << digit << "=" << name << endl;
  user_text =  name + user_text;
}
cout << "That number in digits: " << user_text << endl;
```

Output from running digits in code directory string:

```
echo 51 | ./digits
10<zero-nine>
Give a number:
Digit 0 from the end: 1=one
Digit 1 from the end: 5=five
That number in digits: fiveone
echo 2136 | ./digits
10<zero-nine>
Give a number:
Digit 0 from the end: 6=six
Digit 1 from the end: 3=three
Digit 2 from the end: 1=one
Digit 3 from the end: 2=two
That number in digits: twoonethreesix
```

Strings

String declaration

```
#include <string>
using std::string;

// .. and now you can use ‘string’
```

(Do not use the C legacy mechanisms.)

String creation

A *string* variable contains a string of characters.

```
string txt;
```

You can initialize the string variable (use `-std=c++11`), or assign it dynamically:

```
string txt{"this is text"};
string moretxt("this is also text");
txt = "and now it is another text";
```

Concatenation

Strings can be *concatenated*:

```
txt = txt1+txt2;
txt += txt3;
```

String is like vector

You can query the *size*:

```
int txtlen = txt.size();
```

or use subscripts:

```
cout << "The second character is <<" <<
        txt[1] << ">>" << endl;
```

More vector methods

Other methods for the vector class apply: `insert`, `empty`, `erase`, `push_back`, et cetera.

`http://en.cppreference.com/w/cpp/string/basic_string`

Exercise 2

Code:

```
int user_number;
string user_text{""};
vector<string> names{ "zero","one","two","three",
    "four","five","six","seven","eight","nine"};
cout << names.size() << "<"
     << names[0] << "-" << names[9] << ">" << endl;

cout << "Give a number: ";
cin >> user_number; cout << endl;

for (int d=0; user_number>0; d++) {
  int remember = user_number,digit;
  user_number = user_number/10;
  digit = remember-10*user_number;
  string name = names[digit];
  cout << "Digit " << d << " from the end: " << digit << "=" << name << endl;
  user_text =  name + user_text;
}
cout << "That number in digits: " << user_text << endl;
```

Output from running digits in code directory string:

```
echo 51 | ./digits
10<zero-nine>
Give a number:
Digit 0 from the end: 1=one
Digit 1 from the end: 5=five
That number in digits: fiveone
echo 2136 | ./digits
10<zero-nine>
Give a number:
Digit 0 from the end: 6=six
Digit 1 from the end: 3=three
Digit 2 from the end: 1=one
Digit 3 from the end: 2=two
That number in digits: twoonethreesix
```

Exercise 3

Write a function to convert an integer to a string: the input 205 should give `two hundred fifteen`, et cetera.