Advanced Programming in C++

فرشاد حکیم پور

Character Strings in C++

- Class "string" in the C++ string library provides support for working with strings

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


String Constructor

```
#include <string>
using std::string;
int main()
{
string name("John Doe")
// string name = "John Dow";
string s;   // string s = "";
}
```


C vs. C++ Strings

- C uses an ***array*** of characters to implement strings (with last element always set to null)

```
char[] name = "John Dow";
```


C++ Strings

- C++ strings present more functionalities compared to C strings

```
string name = "John Dow";
cout << "Name: " + name << endl;
cout << "Name length: ";
cout << name.size() << endl;
```

Concatenate two strings

Length of the character string

C++ Strings

- Strings behave like arrays, too.
- "at" method stops the program with error if the position is beyond the length of the string

```
string firstName = "John";
string lastName = "Dow";
cout << firstName[0] << ".";
cout << lastName.at(0) << "."<< endl;
// Output: J.D.
```

Character at position zero

Character at position zero

C++ Strings

- "push_back" method adds a character at the end of string

```
string firstName = "John";
string lastName = "Dow";
string initials;
initials.push_back(firstName.at(0));
initials.push_back('.');
initials.push_back(lastName.at(0));
initials.push_back('.');
cout << initials << endl;
// Output: J.D
```


C++ Strings

- "+=" and "append" method add a string at the end of another string

```
string name = "John";
string lastName = "Dow";
name.append(" " + lastName);
// name += " " + lastName;
cout << name << endl;
// Output: John Dow
```


C++ Strings

- "substr" method returns part of a string

```
string name = "John Dow";
string firstName = name.substr(0,4);
cout << firstName << endl;
// Output: John
string lastName = name.substr(5,3);
cout << lastName << endl;
// Output: Dow
```


C++ Strings

- "find" method finds a string pattern and returns the index of its beginning

```
string s = "John is the first name of John Dow";
int idx = s.find("John");
cout << idx << endl; // Output: 0
// start the search at position 5
idx = s.find("John", 5);
cout << idx << endl;// Output: 26
// Searching from the right side
idx = s.rfind("John");
cout << idx << endl; // Output: 0
```


Strings and Vectors

- C++ strings behave very much like C++ vectors

```
#include <vector>
using std::vector;
. . .
Vector<char> name;
vector<double> xList;
vector<double> yList;
```


Strings and Vectors

- There are many common methods, for example:

```
vector<double> xList (2, 0);
xlist.push_back(100);
cout << xlist.size() << endl; // output: 3
cout << xlist[2] << endl;     // output: 100
cout << xlist.at(2) << endl; // output: 100
```

Exception Handling

- It is a standard method to handle errors
- It prevent software system crash in case of error (fault tolerance)

```
string name("john Dow");
cout << name.at(10) << endl;
```


Exception Handling Example

```
string name("john Dow");
try
{
   cout << name.at(10) << endl;
}
catch(exception &e)
{
   cout<<"Error while processing the name."
   << endl;
}
```


Standard Exception Classes

- Exception classes are defined in <stdexcept> library
- Class “exception” is the root (or super-class) of all 12 classes in this library

Exception Handling Example

```
string name("john Dow");
try
{
   cout << name.at(10) << endl;
}
catch(out_of_range &e)
{
   cout<<"Error while processing the name."
   << endl;
}
```


Exercise 4

- Develop a class for polygon
  - It should support the following methods
    - addPoint(double x, double y)
    - close()
    - bool isClosed()
    - double area()
    - double perimeter()
  - Deadline 4th Dey
  - Send the Source code to tamrin.ut@google.com
  - Subject line: EX4 – student no. – student name