Tutorial 3
NWEN241
Systems Programming

Kirita-Rose Escott

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

Content

- User-defined Types in Standard C
- C++ Programming Problem
- Strings
    - Strings in Standard C
    - Strings in Standard C++

User Defined Types

User Defined Type Problem

- A user wants to enter and record information about movies
- How can they store this information?

Solution

- Create a struct called movie

```
#define DEFAULT_STRLEN 100

typedef struct movie {
	char title[DEFAULT_STRLEN];
	char director[DEFAULT_STRLEN];
	short release_year;
	short oscars_won;
	char imdb_link[DEFAULT_STRLEN];
	char origin_country[DEFAULT_STRLEN];
} movie_t;
```

C++ Programming Problem

Problem

- Define a class to represent a bank account which can store the following data:
  - Account holder’s name
  - Account number
  - Account type (savings or checking)
  - Balance
  - Status
- The following operations are allowed on a bank account:
  - Display account information
  - Deposit
  - Withdraw
  - Close

First Cut

- See ba1.hh

Second Cut

- See ba2.hh

Third Cut

- See ba3.hh

Implement

- Best practice: implement member functions in separate C++ source file
- See ba3.cc

Creating an Instance

- In the current class definition of BankAccount, instance can be created using default constructor
- But the member variables are not initialized

- Create to constructor for creating an instance of BankAccount where
  - Account holder’s name, account number and account type are given
  - Balance is initialized to 0
  - Status is initialized to open
- See ba4.hh

Creating an Instance

- Make sure that we do not create an account without name, number and type
  - Solution: Hide the default constructor
- See ba5.hh

Put It All Together

- Write a main() function to test the implementation
- See ba5.cc

Strings

Strings in Standard C

- Functions in the **string.h** header file
  - strcpy()
  - strcat()
  - strlen()
  - strcmp()

Copying strings to an array of chars

- Will this work?

```
#include <stdio.h>

int main(void) {
    char str[100];
    str = "Hello world.\n";
    printf(str);

    return 0;
}
```

Copying strings to an array of chard

The right way:

- Use either strcpy() or strncpy()

```
#include <stdio.h>

int main(void) {
   char str[100];

   strcpy(str, "Hello world.\n");
   printf(str);

   return 0;
}
```

Concatenating strings

- Concatenating string literals is simple:

```
#include <stdio.h>
#include <string.h>
#define HELLO "Hello”
#define WORLD "world”

int main (void){
   char str[100];

   strncpy(str, HELLO " " WORLD ".\n", sizeof(str));
   printf(str);
   return 0;

}
```

Concatenating strings

Solution:

```
#include <stdio.h>
#include <string.h>

int main(void){
    char str[100];
    char hello[] = "Hello";
    char world[] = "world";

    strncpy(str, hello, sizeof(str));
    strcat(str, " ");
    strcat(str, world);
    strcat(str, ".\n");
    printf("%s", str);

    return 0;
}
```

Comparing strings

```
#include <stdio.h>
#include <string.h>

int main(void){
    char s1[] = "World";
    char s2[] = "world";
    char s3[] = "world";

    int r1 = strcmp(s1, s2);
    int r2 = strcmp(s2, s1);
    int r3 = strcmp(s2, s3);

    printf("r1 = %d\n", r1);
    printf("r2 = %d\n", r2);
    printf("r3 = %d\n", r3);

    return 0;
}
```

Strings in Standard C++

- Can use functions using **cstring**
  - strcpy()
  - strcat()
  - strlen()
  - strcmp()

- C++ provides its own **string** class

Use of **cstring** functions example

```
#include <iostream>
#include <cstring>

int main () {
    char str1[10] = "Hello";
    char str2[10] = "World";
    char str3[10];
    int  len ;

    // copy str1 into str3
   strcpy( str3, str1);
   std::cout << "strcpy( str3, str1) : " << str3 << std::endl;

   // concatenates str1 and str2   std::strcat( str1, str2);
   std::cout << "strcat( str1, str2): " << str1 << std::endl;

   // total length of str1 after concatenation
   len = strlen(str1);
   std::cout << "strlen(str1) : " << len << std::endl;

   return 0;
}
```

C++ String Class

- C++ has a **string** class type that implements a programmer defined string datatype
- Similar to Java String class
- There are multiple constructors to instantiate a string object
- A wide range of operators and member functions are available for variables declared as **string** type

C++ Constructors Examples

- std::string myName("Kirita"); // default constructor
- std::string copyOfMyName(myName); //copy constructor
- std::string copyFromIndex(myName, 3); //copy from index 3
- std::string copyWithSize(myName, 3, 4); // copy from index with length 4

More C++ Constructor Examples

- const char *sourceChar("my string");
- std::string output(sourceChar);   //copy from C-style string
- std::string output(sourceChar, 2);   //copy from C-style string with index
- std::string multiCharacter(4, 'Q');   // output the char, n times

C++ String Functions

- length()/size()
- empty()
- max_size()
- capacity()
- empty()

Length and Capacity Example

```
#include <iostream>
#include <string>

int main () {

// length() and capacity() functions
std::string source("012345678");
std::cout << source.length() << std::endl;
std::cout << source.max_size() << std::endl;

std::string sString1("Not Empty");
std::cout << (sString1.empty() ? "true" : "false") << std::endl;

std::string sString2; // empty
std::cout << (sString2.empty() ? "true" : "false")  << std::endl;

return 0;
}
```