CSI33 Data Structures

Sharon Persinger

Fall 2019

Day 18 November 6

Topics

Types and type casting

More on function parameters

Header files, inline function definitions

Testing, assert

Scope and lifetime, static variables

Introduction to C++ string class

Types and type casting

C++ variables have type, but the compiler doesn’t enforce types in numerical assignment.

Arithmetic operations – division in particular – are type specific.

Types and values of variables and constants can be changed using can be changed using type casting

Examples

Parameter passing in C++

Pass by value –

- The value in the actual parameter (used when the function is called) is copied to the formal parameter, the variable used it the function definition.
- This is the default way parameters are passed in C/C++

Pass by reference – &param in the prototype

- The formal parameter, the variable used in the function definition, receives a reference to the actual parameter, the variable used when the function is called.
- The formal parameter has access to the memory location of the actual parameter.
- By using pass-by-reference, a function can change the parameter used when the function was called.

- Look at the program reference.cpp

Array parameters

Array parameters are always passed by reference.

This means a function with an array parameter can change its actual parameter when the function is called.

Common practice to include the size of the array as an additional parameter to the function that takes an array parameter.

Look at the program selection .cpp

Example: Write a function that gets input from the console into an array of integers. The array is declared before the function. Is called.

Const parameter

You can mark a function parameter with the keyword const to indication that the function should not change the parameter.

This is important when the function takes a parameter that is passed by reference because of its size but that should not be modified.

```
void output_array(const in a[], int size);
```

Header files

Declare functions, classes, non-local variables.

Look at sort.cpp

Preprocessor commands - #ifndef checks if the symbol __SORT_H has been defined. If not, define it

Function declarations next. The code is copied into the file.

If __SORT__H has already been defined, nothing is copied into the file

This construction prevents code from being copied more than one time. Header files can be used over and over in many other files.

To include a header file sort.h that you wrote and stored with your project, use #include “sort.h”

Inline function definitions

Short functions can be defined in the header file - four/five line definitions.

```
inline void fnname(double param1, double param2)
{
//function definition, a few lines
}
```

Notice – no semicolon at end

Testing with Assert statements

C++ does not have a standard unit testing framework.

You can write simple tests using the C++ assert statement. Look at test_sort2.cpp

Scope and lifetime of variables

The scope of a variable is the section of code where the variable can be accessed.

The lifetime of a variable is the execution time period starting when the memory for the variable is allocated and ending when the memory is deallocated.

The lifetime of an automatic variable starts when the function that contains the variable declaration begins execution and ends when the function completes execution.

Scope and lifetime of variables

You can declare a static variable in a function. The lifetime of that variable is the entire execution time of the program. The variable will remember its value from call to call.

Write a function that counts how many times the function is called.

```
int count()
{    static int c = 0;
c++;
return c;}
```

Common C++ errors made by Python programmers

Semi-colon errors –

- forgetting one at end of statement
- placing one at end of Boolean expression in for, while

Omitting braces

Omitting parentheses around Boolean expressions in if, while

Omitting data types in variable definitions