

EECS 381

Objecty-Oriented and Advanced Programming



David Kieras (Maybe) - Fall 2015 (Maybe)

Contributors: Taku Rusike, Max Smith

Latest revision: June 27, 2015

Contents

1	Readings	1
1.1	C concepts: prototypes, headers, linkage	1
	Variable Names	1
	Data Types and Sizes	1
	Constants	2
	Declarations	2
	Arithmetic Operators	2

Abstract

This elective course introduces advanced concepts and techniques in practical C/C++ programming. We will start with a quick, but deep, introduction to important topics in C programming, and then the course will emphasize object-oriented programming with the use of single and multiple inheritance and polymorphism, and using the Standard Library algorithms and containers. Key ideas in object-oriented analysis and design and common design patterns will be introduced. Programming projects will focus on learning and using techniques that are valuable for professional practice in developing and extending large scale or high-performance software relatively easily. In addition to these content goals, an important function of the course is to help students develop good programming practices, style, and skill, with as much personalized coaching and critique of individual students code as possible. In short, the course is intended for those who want to start becoming outstanding programmers.

1 Readings

1.1 C concepts: prototypes, headers, linkage

Variable Names

- Internal variables
 - Internal variables are local to the function it is declared in.
 - They can have a can have a length of 31 characters and can only be comprised of integers and letters and numbers (The underscore counts as a letter).
- External variables
 - External variables (globals) are variables that exist for all functions.
 - External names can only have a length of 6 characters and one case because they may be used by assemblers and loaders.
- Variable naming good practice
 - You shouldn't start off variables with an underscore because library routines often use those names.
 - It is traditional practice to use lower case for variable names, all uppercase for symbolic constants and a capitalized name for object types.

Data Types and Sizes

char a single byte, 8 bits

int an integer, size depends on system

short an integer that is no larger than an integer (typically 16 bits)

long an integer, size is no smaller tahn an int (typically graeter than or equal to 32 bits)

float single-precision floating point

double double precision floating point

unsigned (data type) respective data type that can only be greater than or equal to 0. Because there are no negative numbers, one can have much more positive numbers ($maxSignedNumber * 2 + 1$)

Constants

- An integer constant can be represented like as just an int, hexadecimal, decimal or octal: 1234
- A long constant is represented as a short constant with a terminal L or l, or it is a number too big to fit in an int: 123456789l or 123456789L
- An unsigned integer constant is represented with a u or U at the end of an integer: 1234u or 1234U
- An unsigned long constant is represented as a long but with a ul or UL at the end of the constant: 1234ul or 1234UL
- A double constant is represented as a number with a decimal point or as an exponent: 43.1
- A float constant is the same representation as a double constant but with a f or F suffix: 43.1F or 43.1f
- A character constant is an integer written as one character within single quotes such as 'f'. Certain characters

can be represented by escape sequences. Below is the complete set of escape sequences:

a	alert(bell) character	?
b	backspace	?
f	formfeed	?
n	newline	?
r	carriage return	oc
t	horizontal tab	xh
v	vertical tab	C

- A constant expression is an expression that involves only constants. Such expressions are usually evaluated during compilation rather than at run time
- A string constant or string literal is a sequence of zero or more characters surrounded by double quotes: "Something like this"
- An enumeration is a list of constant integer values that can be specified, really similar to #define. two examples of this is:

```
1 enum boolean {NO, YES}; /*NO = 0, YES = 1*/
2 enum months {JAN = 1, FEB, MAR, APR, MAY} /*JAN = 1, FEB = 2, MAR = 3*/
```

Declarations

- All variables must be declared before use. Declarations specify a type and contains a list of one or more variables of that type
- Automatic (or internal) variables are initialized to garbage variables if not explicitly initialized and are initialized every time that block of code is run
- External and static variables are initialized to zero by default and are initialized once throughout the life of a program
- The qualifier const can be applied to a declared variable and it specifies that the variable will not be changed

Arithmetic Operators

- The binary operators are +, -, *, / and the modulus operator %. Integer division truncates any fractional part. One cannot use % for float or double types.