CH6

6.1

Modular Programming

Breaking program into smaller functions

Improves maintainability, simplifies, reuse, avoids redundancy

Function

Collection of statements that complete a task

6.2

Defining and calling functions

Definition

The statements

Include header

Call

Causes functions to execute

Definition includes

Header – one line

Return type – int void

Name – main (anything we want)

Parameter list – stuff in parenthesis (data types and names)

Body

Returns

Data type must be in header

Return must use data type specified

Calling

Does not require function type 🡪 evenOdd(val)

Sequential

Nested

Need: name, return type, parameter info (data type and order)

6.3

Function Prototypes / Function declarations

Similar to function definition without body

Prototype

Function type, function name, variable types (names recommended but not req)

int func1 (int, double); 🡨 just fine but names would be best

int func1 (int numOfMonths, double intRate);

6.4

Sending Data to a function

Values

Arguments passed into function

Actual Parameter

Actual Argument

Variables

Parameters that hold values passed in

Formal Parameter

Formal Argument

Arguments promoted demoted to match parameter

Value of argument is copied into parameter on function call

Parameters scope is function of call

6.5 / 6.13

Passing Data

Values

Copy value into parameter

Changes do not affect outside function

Reference

Alias

Place & before variable name in proto and definition

Space between data and & irrelevant

Int& x == int & x

6.6

Using functions in Menu-Driven Functions

6.7

The return statement

6.8

Returning value from a function

6.9

Returning a Boolean Value

6.10

Local and Global Variables

Local duration - as long as function in which they are declared executes

Global duration - as long as program executes, all functions defined afterwards can use these variables. Make software reuse more difficult.

6.11

Static local variable

- retain values through successive calls to a function

- static in front of normal declaration (static int)

- initialization rules (system will initialize them for us at 0)

- lifetime (as long as program runs)

- lifetime for a regular (as long as function runs)

- scope is same for regular

- make variable chart for static local, global and local

6.12

Default argument

* is an argument that is passed automatically to a parameter if the argument is missing on the function call.
* Must be declared last, if there is a mix

6.14

Overloading functions

- Two or more functions with same name, different parameters.

Signature

- unique id created by compiler based on function name as well as number, type and order of variables used.

- does not include return data type

6.15

Exit function

Cstdlib

Exit(EXIT\_SUCESS)

Exit(EXIT\_FAILURE)

6.16

Stub

Dummy function

Driver

Tests another function by calling it