C OVERVIEW

02

AE1PGA 2023 - 2024

PROGRAM STRUCTURE

- Procedural programming
- Object-oriented programming
- Functional programming
- Data-flow programming
- Logic programming
- ...

DATA ABSTRACTIONS

Types

Variables

TYPES

Types describe what a value is.

Some type systems are more than others.

Some languages require you to give the type, others try to work it out automatically.

TYPE EXAMPLE: NUMBERS

- A number
- A whole number (integer)
- A non-negative whole number (natural number)
- A natural number between 0..21
- An odd natural number between 0..21
- An odd natural number between 0..21 that is the age of at least 1 person in this room.

VARIABLES

Variables are the abstraction used to store values in the machine.

Language design questions:

- Can their value be changed after the first time?
- Can they be used through the entire program or only in certain places?

INSTRUCTION ABSTRACTIONS

- Sequencing
- Selection (choice)
- Iteration (repetition)
- Recursion (repetition)

DEALING WITH THE OUTSIDE WORLD

- How can a program take input and give output?
- How can we validate the input is correct?
- What do we do if it is not?

ORGANISING THE SOURCE CODE

- A program is broken down into which perform individual tasks. Related functions are put in the same source code files.
- Complete programs can have many source code files.
- You indicate which external functions are required using preprocessor directives.
- Your program will start running at the main function.

WE WILL COME BACK TO FUNCTIONS IN A WHILE...

DATA

- Variables allow us to have data in our programs.
- A variable has a name.
- A variable has a type.
- A variable has a value.
- A variable has an address in memory.

VARIABLE NAMES

- Variable names are identifiers and have some rules on what characters are allowed to be in the name.
- For now, it is enough to know they are a non-digit followed by other letters or digits or underscore _
- Spaces are not allowed in identifiers.
- Variables that never change value are called constants.

VARIABLE TYPES

Whole numbers int (+ short, long)
Real numbers float, double
Characters char
Array types (more than 1 of the same thing) [ ] after the type

MATHS OPERATORS

- Usual mathematical operators are available.

  + - * / % ( )
- Be careful that some are not written the same as in maths (eg, ^ is not raising a number to a power)
- Assignment of a value is done with a single equals =
- Operations involving multiple types have explicit and implicit conversions.
- The types involved can influence the result of the expression.

RELATIONAL OPERATORS

> < >= <= == !=

- Comparison of two values is done with double equals ==
- Not equals is !=
- For greater/less than or equals to, the equals sign must be the second character >= <=

OPERATOR PRECEDENCE

- Textbook Page 83

- Online materials

https://en.cppreference.com/w/c/language/operator_precedence

SEQUENCING INSTRUCTIONS

- Instructions are executed one after another
- Simply, this means one line at a time going down the function
- The program starts at the start of main()
- Calling a function jumps to the start of that function
- When the function is finished it jumps back (returns) to the line after where it was called from.

SELECTION STATEMENTS

- if statement, do it if true
- if ... else... statement, do one or the other
- switch statement, do 1 of many things
- In C, 0 (zero) is false, non-zero is true!

REPETITION STATEMENTS

- for loop, do something x times
- while loop, do something an unknown number of times
- do...while loop, variation of while loop
- Any loop has a condition when they stop

FUNCTIONS

- Individual, stand-alone units of functionality
- Sequences of instructions that can act on different data
- Can take parameters (or arguments) as input
- Can return a value

BLOCKS AND SCOPE

- { } define the start and end of a block.
- A variable declared in a block is only inside that block.
- Each function defines a new block.
- Most instruction statements define new blocks.
- You can start a new block almost anywhere you can have an individual instruction.

EXTERNAL FUNCTIONS

Sometimes a function:

- is used in the same file it is written (we wrote it),
- is used in a different file (we wrote it),
- is part of the standard library (we didn't write it),
- is part of a 3rd party library (we didn't write it)

Use #include or extern to indicate where functions are defined.

Example: Variables Definition

Write a program which defines an integer and a real number, and then prints their values on the screen.

Example: Input/Output

- Input with scanf()
- Output with printf()
- Standard library functions in stdio.h.

Example: C Math Functions

Write a program which computes 2^16.

1)	ceil(number)	rounds up the given number. It returns the integer value which is greater than or equal to given number.
2)	floor(number)	rounds down the given number. It returns the integer value which is less than or equal to given number.
3)	sqrt(number)	returns the square root of given number.
4)	pow(base, exponent)	returns the power of given number.
5)	abs(number)	returns the absolute value of given number.

Example: Hypotenuse

Write a program which reads from the keyboard two integers to describe the lengths of two short sides of a right triangle and then compute the hypotenuse (the longest side of a right triangle).

Example: RIGHT-ANGLED TRIANGLE CHECKER

Write a program which reads in the lengths of three sides of a triangle and checks if they form a right-angled triangle. Print an appropriate message if they do or do not.

Example: TYPHOON WIND STRENGTH

Using the JTWC Tropical Typhoon intensity scale, write a program which takes wind-speed as input and outputs the classification of intensity (eg, typhoon, violent typhoon, etc).

Intensity Class	Wind speed (km/h)
Tropical Depression	- 62
Typhoon	63 - 118
Strong Typhoon	119 - 156
Very Strong Typhoon	157 - 192
Violent Typhoon	193 -