C -- overview

COEN 10

C -- Lecture 1

Overview

★C

- ◎Created by Kernighan and Ritchie in 1972 to write the Unix operating system
- ◎Created as a language designed for programmers

Overview

★Why C

- ◎Design Features
- ◎Efficiency
- ◎Portability
- ◎Power and Flexibility
- ◎Programmer Oriented
- ◎Shortcomings

Overview

★Where C is used

- ◎Operating Systems -- UNIX
- ◎Computer languages
- ◎Embedded systems
- ◎Computer games
- ◎Robot factories
- ◎Scientific applications
- ◎Special effects in movies
- ◎Many others…

Overview

★What computers do

- ◎CPU
- ◎Registers
- ◎RAM
- ◎Permanent memory

Overview

★What computers do

- ◎Everything is represented as a number
  - ✧Numbers
  - ✧Characters
  - ✧Instructions - machine language

Overview

★Better to program with a High Level Programming language

- ◎No numeric codes
- ◎High Level instructions closer to the way we think
- ◎But the computer does not understand them

Overview

★Compilers

- ◎Program that translates the high-level language program into machine language
- ◎Make programs portable

Overview

★Using C
- ◎7 steps
  - ✧Define Program objectives
  - ✧Design Program
  - ✧Write the code
  - ✧Compile
  - ✧Run
  - ✧Test and Debug
  - ✧Maintain and modify

Overview

★Programming Mechanics
- ◎source code
- ◎libraries
- ◎object code
- ◎executable code

Overview

★Programming Mechanics
- ◎Compiler
- ◎Linker
- ◎Loader

Overview

★Programming Mechanics
- ◎In Linux or MAC OS
  - ✧Edit with vi or vim
    - –Generate a .c
  - ✧Compile (and link) with gcc
    - –Create an a.out file
  - ✧Execute by entering ./a.out in the command line

Overview

★The C99 Standard

◎Preserve essential nature of the language defined in C90

◎Main changes

✧Internationalization

✧Correction of deficiencies

✧Improvement of computational usefulness

Intro to C

★A simple example

```
#include <stdio.h>
int main (void)
{
   int num;                          /* define variable num */
   num = 1;                          /* assign a value to num */

   printf ("I am a simple computer.\n");      // use printf
   printf ("My favorite number is %d because it is first.\n", num);

   return 0;
}
```

Intro to C

★C programs may contain preprocessor instructions

◎Include another file

◎Define constants

Intro to C

★C programs consists of one or more functions, identified by the parenthesis

★There is always a main function, the entry point of the program

Intro to C

- ★Functions contain statements
  - ◎Declarations
  - ◎Assignments
  - ◎Function calls
  - ◎Control
    - ✧Conditionals and loops
  - ◎Null

Program Details

#include

- ★Directive for the preprocessor to include a .h header file
- ★Different functions require different definitions
- ★Example

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

Program Details

- ★The main() Function

```
int main (void)
```

- ★int - main returns an integer
- ★void - no arguments

Program Details

- ★Comments

```
/* comment across lines */
// comment at the end of a line
```

Program Details

★Braces, Bodies, Blocks

{
…
}

★To delimit
- ◎a function body
- ◎blocks

Program Details

★Declarations
- ◎Connects an identifier with a data type and a location in memory
- ◎All variables need to be declared
- ◎Good practice
  - ✧Declare all variables in the beginning of each function

Program Details

★Data Types
- ◎Integer
- ◎Character
- ◎floating point

Program Details

★Name Choice
- ◎Use meaningful names
- ◎Rules
  - ✧Case sensitive
  - ✧Uppercase, lowercase, digit, and _
  - ✧1st character: letter or _
  - ✧Avoid start identifiers with _, to avoid conflicts with the libraries

Program Details

★Assignment

- ◎= operator
- ◎Always from right to left
- ◎Always ends with a ;

Program Details

★The printf() function

printf ("string", arg1, arg2, …);

- ◎Outputs a string
- ◎The string may contain placeholders to be substituted by the values of the other arguments

Program Details

★Return Statement

- ◎Specify the end of the function
- ◎Should return the same type specified as the function type

The Structure of a Simple Program

```
#include <stdio.h>

int main (void)
{
      statements
      return 0;
}
```

The Structure of a Simple Program

- ★Make it Readable
  - ◎Variables with meaningful names
  - ◎Use comments
  - ◎Blank lines to separate conceptual sections of the program
  - ◎One line per statement
  - ◎Alignment and indentation

Debugging

- ★Syntax errors
- ★Semantic errors

Debugging

- ★Tracing
  - ◎Follow the program state step-by-step by hand
  - ◎Program state - set of values of all variable at a point

Debugging

- ★printfs
  - ◎Sprinkle printf statements throughout the program
    - ✧to monitor the program flow
    - ✧to monitor the values of selected values at key points
  - ◎Remove them when you are done!

Debugging

- ★Debuggers
  - ◎gdb

Keywords and Reserved Identifiers

- ★Cannot use keywords as identifiers
  - ◎Table on page 44