CIS 314

Introduction to C

Prof. Michel A. Kinsky
Computing: The art of abstraction

Application
Algorithm
Programming Language
Operating System/Virtual Machine
Instruction Set Architecture (ISA)
Microarchitecture
Register-Transfer Level (RTL)
Circuits
Devices
Physics
Computing: The art of abstraction

The layers that we will touch in this class:

1. Application
2. Algorithm
3. Programming Language
4. Operating System/Virtual Machine
5. Instruction Set Architecture (ISA)
6. Microarchitecture
7. Register-Transfer Level (RTL)
8. Circuits
9. Devices
10. Physics
Programming Languages

- There are many programming languages available: Pascal, C, C++, Java, Ada, Perl and Python
- All of these languages share core concepts
- By focusing on these concepts, you are better able to learn any programming language
Programming Languages

• There are many programming languages available: Pascal, C, C++, Java, Ada, Perl and Python

• All of these languages share core concepts

• Hence, by learning C, you are poised to learn other languages, such as Java or Python

  ‣ In this class, we will learn core programming concepts through the powerful C language

  ‣ Why C? It runs nearly as fast as assembly language code
Programming Process

1. Write Program
2. Compile Program
3. Run Program
4. Debug Program

Flowchart: Write Program → Compile Program → Run Program → Debug Program → Compile Program → Write Program
Introduction to C

• Developed in 1972 by Dennis Ritchie at Bell Labs
• It is imperative programming language
• It provides:
  - Efficiency, high performance and high quality software
  - Flexibility and power
  - Many high-level and low-level operations
Introduction to C

• Developed in 1972 by Dennis Ritchie at Bell Labs
• It is imperative programming language
• It provides:
  ‣ Stability and small size code
  ‣ Provide functionality through rich set of function libraries
  ‣ Gateway for other professional languages like C++ and Java
Introduction to C

- It is used:
  - System software, Compilers, Editors, embedded systems, application programs
  - Data compression, graphics and computational geometry, utility programs
  - Databases, operating systems, device drivers, system level routines

- The real world still runs on C
  - Most of legacy code in use are in C
  - Many other programming languages are based on C
Introduction to C


Original comic is available here: http://xkcd.com/519/
Basic C variable types

- There are five basic data types in C
  - Char: ‘a’
    - A single byte capable of holding one character in the local character set
  - Int: 3
    - An integer of unspecified size
  - Float: 3.14
    - Single-precision floating point
  - Double: 3.1415926
    - Double-precision floating point
  - Void: Valueless special purpose type
## Basic C variable types

<table>
<thead>
<tr>
<th>Type (32 bit)</th>
<th>Smallest Value</th>
<th>Largest Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>short int</td>
<td>-32,768(-2^{15})</td>
<td>32,767(2^{15}-1)</td>
</tr>
<tr>
<td>unsigned short int</td>
<td>0</td>
<td>65,535(2^{16}-1)</td>
</tr>
<tr>
<td>Int</td>
<td>-2,147,483,648(-2^{31})</td>
<td>2,147,483,648(2^{31}-1)</td>
</tr>
<tr>
<td>unsigned int</td>
<td>0</td>
<td>4,294,967,295</td>
</tr>
<tr>
<td>long int</td>
<td>-2,147,483,648(-2^{31})</td>
<td>2,147,483,648(2^{31}-1)</td>
</tr>
<tr>
<td>unsigned long int</td>
<td>0</td>
<td>4,294,967,295</td>
</tr>
</tbody>
</table>
Variable assignment

- In C variables must be declared.
- They are given values through assignments.
- Assignment is done with the '==' operator.

**Declarations**

```c
int number_of_students;
float average_gpa;
```

**Assignments**

```c
number_of_students = 12;
average_gpa = 3.9;
```
Variable assignment

- In C variables must be declared
- They are given values through assignments
- Assignment is done with the '=' operator

**Declarations**

```
int number_of_students;
float average_gpa;
```

**Assignments**

```
number_of_students = 12;
average_gpa = 3.9;
```
#include <stdio.h> /* Header files */

int main(void) {
    printf ("Hello World!\n") ;
    return 0;
}
C Program compilation

Compile: gcc – o myhello
hello.c

Run: ./myhello
C Program Analysis

- `#include <stdio.h> /* Header files */`
  - It is a preprocessor directive
  - It tells computer to load contents of the file
  - It allows standard input/output operations

- Comments are used to describe program
  - Text surrounded by `/*` and `*/` is ignored by computer
  - Lines starting with `//` are also ignored
C Program Analysis

• int main (void)
  ‣ C programs contain one or more functions, exactly one of which must be main
  ‣ Parenthesis used to indicate a function
  ‣ int means that main "returns" an integer value

• Braces ({ and }) indicate a block
  ‣ Bodies of all functions must be contained in braces

• printf ("Hello World!\n")
  ‣ printf and scanf functions
C Program Analysis

• `printf`
  ‣ Sends output to standard output
  ‣ General form
    • `printf(format descriptor, var1, var2, ...);`
  ‣ `printf("%s\n", "Hello world");`
    • Translation: Print hello world as a string followed by a newline character
  ‣ `printf("%d\t%f\n", j, k);`
    • Translation: Print the value of the variable j as an integer followed by a tab followed by the value of floating point variable k followed by a new line
C Program Analysis

• scanf
  ‣ Gets inputs from user
  ‣ General form
    • scanf(format descriptor, &var1, &var2, …);
    • scanf(“%f”, &i);
  ‣ Translation: Get floating point input i from user
    • scanf(“%d %f\n”, &j, &k);
    • Translation: Get the value of the variable j as an integer followed by the value of floating point variable k from user
    • Blocks program until user enters input
C Program Analysis

• Some special characters are not visible directly in the output stream
• These begin with an escape character (\);
  ‣ \n    newline
  ‣ \t    horizontal tab
  ‣ \a    alert bell
  ‣ \v    vertical tab
C Program Operations

• Arithmetic operators
  ‣ + “plus”
  ‣ - “minus”
  ‣ * “times”
  ‣ / “divided by”

```c
#include <stdio.h> /* Header files */
int number1, number2, number3;

int main(void) {
    scanf("Enter number1: %d", &number1);
    scanf("Enter number2: %d", &number2);
    number3 = number1 + number2;
    printf ("Number1 + number 2 = %d\n", number3 ) ;

    number3 = number1 - number2;
    printf ("Number1 - number 2 = %d\n", number3 ) ;

    number3 = number1 * number2;
    printf ("Number1 * number 2 = %d\n", number3 ) ;

    number3 = number1 / number2;
    printf ("Number1 / number 2 = %d\n", number3 ) ;
    return 0;
}
```
C Program Comparators

• Relational operators:
  ‣ == “is equal to”
  ‣ != “is not equal to”
  ‣ > “greater than”
  ‣ < “less than”
  ‣ >= “greater than or equal to”
  ‣ <= “less than or equal to”
There are two logical operators in C

- `||` “logical or”
  - An expression formed with `||` evaluates to true if any one of its components is true

- `&&` “logical and”
  - An expression formed with `&&` evaluates to true if all of its components are true
Advance Data types

• In C
  ‣ Arrays (a list of data (all of the Same Data Type!))
    • int grades[ ] = {94, 78, 88, 90, 93, 87, 59};
  ‣ Structures (a collection of named data referring to a single entity)

```c
struct Student {
    char Name[50];
    int id;
    float GPA;
    char major[25];
};
```
Advance Data types

• Pointers in C
  ‣ Pointers are memory addresses
  ‣ Every variable has a memory address
  ‣ Symbol & means “take the address of” e.g., &x
  ‣ Symbol * means “take the value of” e.g., *p
  ‣ Symbol * is also used to denote a pointer type e.g., int *q;
Advance Data types

• Pointers in C
  ‣ Declaration of integer pointers and an integer number
    • int * pointer1, * pointer2;
    • int number1;
  ‣ Setting pointer1 equal to the address of number1
    • pointer1 = &number1;
  ‣ Setting pointer2 equal to pointer1
    • pointer2 = pointer1;
Next Class

• Conditional, Functions, Recursions