Week 11 - C and Python Programming

An Introduction to C

- A statically typed language
  (i.e variables must be declared to be of a particular type, and can only hold that type)
- A compiled language
- A functional language that supports structured programming.
- Developed by Dennis Richie 1971-73.

The creator of C - Dennis Richie

Richie was also heavily involved in the development of the Unix operating system.

Ritchie's death did not receive much news coverage as the media was focussed on Steve Jobs who died the week before.

Computer historian Ceruzzi stated that:

“Ritchie was under the radar. His name was not a household name at all, but [...] if you had a microscope and could look in a computer, you'd see his work everywhere inside. “

Sources:

Srinivasan, Rajeev (October 25, 2011). "Dennis Ritchie, a tech genius as great as Steve Jobs". Firstpost. Retrieved December 4, 2017.

Langer, Emily (October 14, 2011). "Dennis Ritchie, founder of Unix and C, dies at 70".

Washington Post. Retrieved November 3, 2011.

Learning Other Languages

The concepts you have already learned are largely the same for other languages.

- Variables and constants
- Loops and conditional statements
- Arrays to hold multiple values
- Library files for code that is already written and can be re-used.

Using C

C is a language that may be described as “closer to the machine”:

- It does less for you (so you can only have it do what you want)
- But it generates smaller code that therefore runs faster (tradeoff productivity for performance)
- It can appear cryptic, and gives more cryptic errors some other languages.

C and other languages

- C syntax is the basis for many modern languages
- So knowing C is useful just for syntax.
- We use the GNU C++ compiler (gcc)
- So in labs you will need to run mingw.

Some Differences between C and Ruby

C is case sensitive

You lose some features:

- There are no Strings in C - you have to use arrays of characters
- You can't pass by reference the way you do in Ruby (C++ can) - you have to pass pointers
- Functions / procedures can't be passed or return arrays by-value (i.e a copy of the array)
- To pass an array in you pass in a pointer to the 1st element

C String handling is lower level than Ruby

```
string = char* or char my_array[n]
```

- Arrays of characters, terminated with the null character
- Make sure you don't read/write beyond the end of the array
- In C you need to store the length of the array yourself (Ruby does it for you)

C Naming Conventions

There are a number of C language conventions... we use...

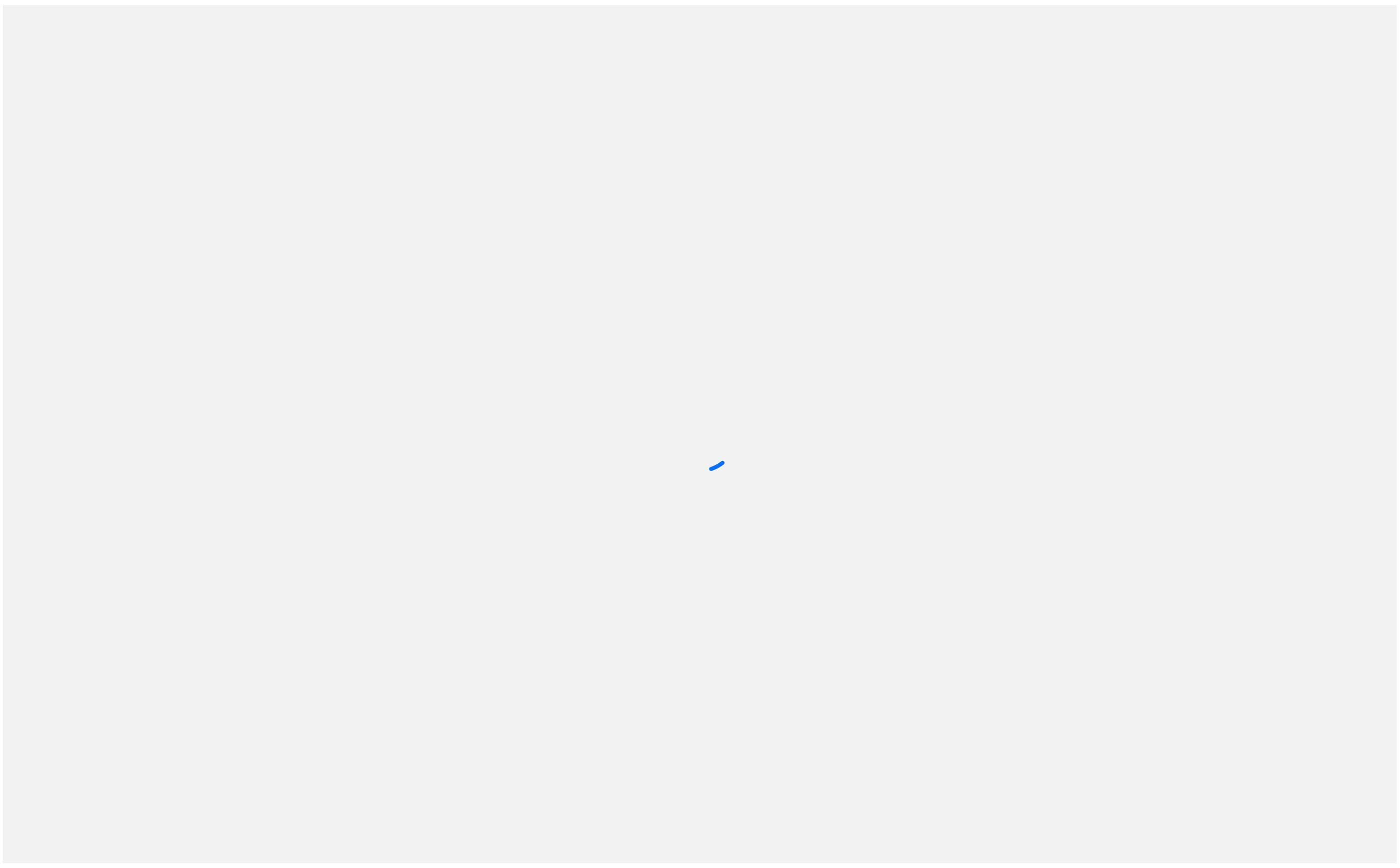
- lower_case = identifiers for functions/procedures, types, variables
- UPPER_CASE = constants
- Indent within { }, and control structures (as we will see)

Languages have similar statements to perform actions

Variables and Constants are used to store values

Code is divided into Functions and Procedures, with Parameters and Local Variables

Use procedure calls to run procedures



Create Custom Data Types

Arrays are used to store multiple values

Use assignment statements to store values in variables

C uses pointers to access memory

Branch with if and case statements

Loop with while, for, and repeat or do … while loops

Return or Exit from functions and procedures

Strings

Not all languages have the same level of type support

string.h library

In C you need to use a function to compare strings (as they just an array of characters – the strcmp function checks each character to see if it is the same in both arrays)
Eg:

If the arrays have the same content, strcmp will return 0, otherwise it will return a positive integer.

You will need the above for this week’s tutorial task.

Terminal and File Output

Use printf() to write to the terminal *(and use fprintf() to write to file)*

printf() is defined in the library <stdio.h>
(<somelib.h> means a system library)

Terminal and File Input

Read from the user with our functions for read string etc.

- Use scanf() of gets() to read values from the user
- Both printf and scanf use format strings to indicate the format of the output/input data
- You could use the supplied terminal user input code — (access the string characters using the .str field of the record/struct)

Reading from the terminal in C

Testing scanf()

Using libraries

In C you need to #include libraries (like require in Ruby)

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

OR:

```
#include "terminal_user_input.h"
```

Ruby equivalent:

```
require './input_functions'
```

(although the base Ruby language has gets() puts() and printf() etc – which C does not)

A C library program

This is library program in C that allows us to reuse code.

It has a public interface and private (?) implementation (demonstrating information hiding and to some degree, encapsulation):

String functions

strcmp()

Header Files

C has header files that contain function prototypes:
Eg: terminal_user_input.h:

The actual functions are then contained in the matching .c program file Eg: an extract from terminal_user_input.c:

Example C program

Here is a C program similar to the task you did in Ruby for Test 1.

A Ruby Library file

This code slide does not have a description.

Summary

- We have looked at the C language
- The concepts should be the same, although the syntax and implementation of somethings (eg: Strings) can be different
- Use the example code above to help with this week’s tutorial task.

Python Programming

Python is more similar to Ruby than C is. The main differences for the types of tasks we are doing is in relation to syntax and program language constructs. Here are some differences:

1. Defining functions and procedures

2. The form of conditional statements and loops

Note, that indentation in Python is critical to the code being interpreted correctly. There is no `end` keyword as in Ruby, instead indentation is used to indicate where blocks of code finish.

Python Programming - Defining functions and procedures:

1. Defining functions and procedures in Ruby, then in Python

Run RUBY

```ruby
def procedure()
    print("this is a Ruby procedure\n")
end

def function()
    value = 5
    return value
end

def main()
    procedure()
    print(function())
end

```

Run PYTHON

```python
def procedure():
    print("this is a Python procedure") # note: it prints a newline

def function():
    value = 5
    return value

def main():
    procedure()
    print(function())

main()
```

Python Programming - Conditionals and Loops

IF statements in Python:

Run PYTHON

```python

def main():
    a = 5
    b = 10
    if (a < 4) or (b > 6):
        print("Consequent")
    else:
        print("Alternative")

main()

```

Run RUBY

```ruby
def main()
    a = 5
    b = 10
    if (a < 4) or (b > 6)
        print("Consequent")
    else
        print("Alternative")
    end
end

main()
```

Loops in Python:

Run PYTHON

```python
def main():
    i = 0
    while (i < 5):
        print(str(i) + " loop")
        i += 1

main()
```

Loop in Ruby:

```ruby
def main()
    i = 0
    while (i < 5)
        print(i.to_s + " loop\n")
        i += 1
    end
end

main()
```