Notes

Introduction to Computer Science (CS50) on EdX

Sparsh Jain

December 11, 2020

Contents

I General 6

1 Computational Thinking, Scratch 6
- 1.1 Binary Number System . . . 6
- 1.2 Algorithms . . . 6
- 1.3 Time Complexity . . . 6
- 1.4 Pseudocode . . . 6
- 1.5 Scratch . . . 6

2 C 7
- 2.1 Hello World . . . 7
- 2.2 Input . . . 7
- 2.3 Initialization . . . 9
- 2.4 Increment . . . 9
- 2.5 Conditionals . . . 9
- 2.6 Loops . . . 9
  - 2.6.1 While Loop . . . 9
  - 2.6.2 For Loop . . . 10
- 2.7 Additional Info . . . 10
  - 2.7.1 Datatypes . . . 10
  - 2.7.2 Functions . . . 10
  - 2.7.3 Placeholders . . . 11
  - 2.7.4 Arithmetic Operations . . . 11
- 2.8 Examples . . . 11
  - 2.8.1 Arithmetic . . . 11
  - 2.8.2 Conditional . . . 14
  - 2.8.3 Logical . . . 15
  - 2.8.4 Loop . . . 16
  - 2.8.5 Function . . . 17
- 2.9 Limitations . . . 22

3 Arrays 24

3.1 Compiling 24
3.1.1 Preprocessing 24
3.1.2 Compiling 24
3.1.3 Assembling 24
3.1.4 Linking 24
3.2 Debugging 24
3.3 Casting 25
3.4 Array 25
3.5 String 26
3.6 Command Line Arguments 34

4 Algorithms **36**
4.1 Linear Search 36
4.2 Binary Search 36
4.3 Efficiency 37
4.3.1 $\mathcal{O}$ Notation: 37
4.3.2 Ω Notation: 37
4.4 Examples 38
4.4.1 Linear Search 38
4.4.2 Bad Design 39
4.4.3 Good Design - `typedef struct` 40
4.5 Bubble Sort 41
4.6 Selection Sort 41
4.7 Better Bubble Sort 42
4.8 Recursion 42
4.9 Merge Sort 45
4.9.1 Θ Notation 45

5 Memory **46**
5.1 Hexadecimal 46
5.2 Addresses 46
5.2.1 Operators 47
5.3 Pointers 48
5.4 Strings 49
5.5 String Comparision 51
5.6 String Copy 53
5.7 Malloc and Free 54
5.8 Buffer Overflow 54
5.9 Swap 55
5.10 `scanf` 57
5.11 File I/O 58

6 Data Structures **61**
6.1 Arrays 61
6.2 Data Structures 64
6.3 Linked List 64
6.4 Tree 66
6.4.1 Binary Search Tree 66
6.5 Hash Table 67
6.6 Trie 67
6.7 Queue 68
6.8 Stack 68
6.9 Dictionary 68

7 Python **69**
7.1 Introduction 69
7.2 Datatypes 80
7.3 Previous assignments from C to python 81
7.4 Regular Expressions 82
7.5 Fancier stuff: Hardware usage 83

8 Database **86**
8.1 csv files 86
8.2 SQL 90
8.2.1 Example 90
8.2.2 Relational Database 90
8.2.3 Syntax 91
8.2.4 Huge Database 93
8.3 Problems 97
8.3.1 Race Conditions 97
8.3.2 SQL Injection Attacks 97

9 Where to? **98**
9.1 How far we have come! 98
9.2 Tracks 99
9.2.1 Web Programming 99
9.2.2 Mobile App Development 99
9.2.3 Game Development 99

II Web **100**

10 Introduction **101**
10.1 Protocols 101

10.1.1 IP addresses 101
10.1.2 Port Numbers 102
10.1.3 URL: Domain Name System 102
10.1.4 HTTP(S) 102
10.1.5 Status Codes 102

11 HTML **103**

12 CSS **108**

13 JavaScript **114**
13.1 Syntax 114
13.2 Document Object Model 115

14 Flask **123**
14.1 Hello World 123
14.2 Templates 124
14.3 Variables 124
14.3.1 String 124
14.3.2 Random Numbers 125
14.4 Conditions 126
14.4.1 Coin Flip 126
14.5 Interactive Webpage 127
14.5.1 Forms 127
14.6 Layouts 130
14.7 Tasks Application 131

15 Databases **134**
15.1 Sessions 134
15.2 SQL 135

III Android **138**

16 Java **139**
16.1 Introduction 139
16.2 Data Types 139
16.3 Examples 139
16.4 Generics 140
16.4.1 Lists 140
16.4.2 Maps 140
16.5 Classes 141

16.6 Static Methods 141
16.7 Inheritance 142
16.8 Interfaces 142
16.9 Packages 143
16.10Android 143

Appendices **146**

List of Programs **147**

Part I

General

Part II

Web

Part III

Android

Chapter 16

Java

16.1 Introduction

Use Android Studio (IDE) to build android apps. Convention for package name is your domain name in reverse followed by app name. Use androidx (newer version android libraries). Also need to create AVD (Android Virtual Device) to simulate an android device to run your app. Use Java to code.

16.2 Data Types

- `boolean`
- `double`, `float`
- `char`
- `int`
- `List`
- `Map`
- `String`
- …

16.3 Examples

```
String title = "CS50";
int count = 50;
```

```java
count += 5;

String title = "iOS";
if (title.equals("iOS")) {
        System.out.println("Good Choice");
}
else {
        System.out.println("Maybe Next Time");
}

int[] values = new int[]{1, 2, 3};
for (int i = 0; i < values.length; i++){
        System.out.println(i);
}
```

Program 16.1: First few lines of java

16.4 Generics

16.4.1 Lists

```java
List<String> values = new ArrayList<>();
values.add("one");
values.add("two");
for (String value : values){
        System.out.println(value);
}
```

Program 16.2: Lists in java using Generics

16.4.2 Maps

```java
Map<String, String> airports = new HashMap<>();
airports.put("SFO", "San Francisco");
airports.put("BOS", "Boston");
for (Map.Entry<String, String> e : airports.entrySet()) {
        System.out.println(e.getKey() + ": " + e.getValue());
}
```

Program 16.3: Maps in java using Generics

16.5 Classes

structs + functions/methods = class

```java
public class Person {
        String name;

        Person(String name) {
                this.name = name;
        }

        public void sayHello() {
                System.out.println("I'm " + name);
        }
}

Person person = new Person("Tommy");
person.sayHello();
```

Program 16.4: Classes in java

16.6 Static Methods

Can be called from a class, without having an instance of it.

```java
public class Person {
        ...
        public static void wave() {
                System.out.println("Wave");
        }
}

Person.wave();
```

Program 16.5: Static Methods in java

16.7 Inheritance

```java
public class Vehicle {
        public int wheels() {
                return 4;
        }

        public void go() {
                System.out.println("zoom!");
        }
}

public class Motorcycle extends Vehicle {
        @Override
        public int wheels() {
                return 2;
        }
}
```

Program 16.6: Inheritance in Java Classes

16.8 Interfaces

Basically a list of methods to implement in classes. If we forget, compiler raises an error.

```java
public interface Teacher() {
        public void teach();
}

public class CS50Teacher implements Teacher {
        @Override
        public void teach() {
                ...
        }
}
```

Program 16.7: Interfaces in Java Classes

Remark. We can implement multiple interfaces but only extend one class.

16.9 Packages

Sort of a way to organise java code.

```java
package edu.harvard.cs50.example;

import java.util.List;
```

Program 16.8: Packages in Java

16.10 Android

```java
package com.example.javaexample;

public class House {
    private String name;
    private String head;

    House(String name, String head){
        this.name = name;
        this.head = head;
    }

    public String getName(){
        return name;
    }

    public String getHead(){
        return head;
    }
}
```

Program 16.9: House class in Java

```java
package com.example.javaexample;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.os.Trace;
import android.util.Log;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Random;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        List<House> houses = new ArrayList<>();
        houses.add(new House("Gryffindor", "McGonagall"));
        houses.add(new House("Hufflepuff", "Sprout"));
        houses.add(new House("Ravenclaw", "Flitwick"));
        houses.add(new House("Slytherin", "Snape"));

        List<String> students = Arrays.asList("Harry", "Ron",
         ↪  "Hermione", "Neville", "Draco", "Parvati", "Padma",
         ↪  "Cho", "Cedric");
        Map<String, House> assignments = new HashMap<>();

        Random random = new Random();
        for (String student : students) {
            int index = random.nextInt(houses.size());
            assignments.put(student, houses.get(index));
        }

        for (Map.Entry<String, House> entry :
         ↪  assignments.entrySet()) {
            House house = entry.getValue();
            Log.d("cs50", entry.getKey() + " got " +
             ↪  house.getName() + " under " + house.getHead());
        }
    }
}
```

Program 16.10: Example Android Application in Java

Appendices

List of Programs

4.1 Linear Search Pseudocode 36
4.2 Binary Search Pseudocode 36
4.3 Linear Search on numbers 38
4.4 Linear Search on names 39
4.5 Linear Search in a phonebook 40
4.6 Linear Search in phonebook with `typedef struct` 41
4.7 Iteration Pseudocode 42
4.8 Recursion Pseudocode 43
4.9 Iteration C code 43
4.10 Recursion C code 44
4.11 Merge Sort Pseudocode 45

5.1 integer 46
5.2 address of an integer 47
5.3 address2.c 47
5.4 accessing an address 48
5.5 pointers 48
5.6 strings 49
5.7 strings are pointers 49
5.8 strings are `char[]` addresses are consecutive in arrays 50
5.9 accessing characters in a string 50
5.10 accessing characters in a `char *` 50
5.11 comparing integers 51
5.12 attempting to compare strings directly 52
5.13 comparing strings properly 52
5.14 attempting to copying strings directly 53
5.15 copy strings properly 54
5.16 buffer overflow 55
5.17 naive attempt at swap 55
5.18 swap 56
5.19 scanning an integer 57
5.20 scanning a string in unintialized 57

5.21 scanning a long string in small array 58
5.22 files in c 59
5.23 phonebook.csv 59
5.24 check jpeg or not 60

6.1 array with hardcoded size 62
6.2 array with dynamic size using `malloc` 63
6.3 array with dynamic size using `realloc` 64
6.4 linked list 66
6.5 node for a binary tree 66
6.6 search in a binary-search-tree 67

7.1 Hello Python 69
7.2 strings in python 69
7.3 print function in python 69
7.4 format strings 70
7.5 integers in python 70
7.6 comparisions in python 70
7.7 logical operators in python 71
7.8 convert string to lowercase in python 71
7.9 while loop in python 71
7.10 for loop and `range` in python 72
7.11 functions in python 72
7.12 arguments to functions in python 72
7.13 scopes in python 73
7.14 named arguments in python 73
7.15 multiplying a string: pythonic 73
7.16 nested loops in python 74
7.17 input strings in python 74
7.18 input integers in python 74
7.19 overflow in python? 74
7.20 lists in python 75
7.21 directly using lists in python 75
7.22 access characters of a string in python 75
7.23 accessing characters of a string directly in python 76
7.24 changing to uppercase in python 76
7.25 command line arguments in python 76
7.26 directly accessing command line arguments in python 77
7.27 exiting on error in python 77
7.28 searching in a list in python 77
7.29 dictionary in python 78

7.30 string comparision in python 78
7.31 swapping values in python 79
7.32 files in python 79
7.33 `with` in python 80
7.34 blur.py: blur an image 81
7.35 dictionary.py: implement a dictionary 82
7.36 regex in python 82
7.37 extremely simple AI 83
7.38 speach recognition in python 83
7.39 reply with speach recognition in python 84
7.40 iteractive speach recognition in python 85

8.1 Read a csv file in python 86
8.2 Use a dictionary to count in python 87
8.3 Print sorted dictionary by ’keys’ in python 88
8.4 Print sorted dictionary by ’values’ in python 89
8.5 lambda function in python 90
8.6 load a csv to a db in sqlite3 90
8.7 SQL querries in sqlite3 90
8.8 SQL Syntax 92
8.9 filtering the database in python 94
8.10 searching the database in python 94
8.11 using SQL in python 96
8.12 import to multiple tables in SQL using python 97
8.13 query with multiple tables in SQL 97
8.14 indexing in sql 97

9.1 brute-forcing 4-digit pins in python 98
9.2 brute-forcing dictionary words in python 98

11.1 hello html 103
11.2 image in html 104
11.3 link in html 104
11.4 paragraphs in html 105
11.5 headings in html 106
11.6 table in html 106
11.7 form in html 107

12.1 inline styling in html 108
12.2 multiple styles within an html element 109
12.3 css classes in html 110
12.4 multiple css classes in an html element 110

12.5 separate css file . . . 111
12.6 styled table in html . . . 112
12.7 using bootstrap css library . . . 113

13.1 JavaScript syntax . . . 115
13.2 Alert using JavaScript . . . 116
13.3 Updating webpage using JavaScript . . . 116
13.4 Variables in a webpage using JavaScript . . . 117
13.5 Changing background using JavaScript . . . 118
13.6 Updating font size using JavaScript . . . 119
13.7 Blinking a content using JavaScript . . . 120
13.8 JavaScript code in a separate file . . . 121
13.9 HTML using external JavaScript file . . . 121
13.10 Getting location of the user via JavaScript . . . 122

14.1 Hello World in Flask . . . 123
14.2 Templates in Flask . . . 124
14.3 Variables in Flask . . . 124
14.4 Jinja syntax for (flask) variables in HTML . . . 125
14.5 Passing Random Numbers from Flask . . . 125
14.6 Displaying random numbers in HTML . . . 126
14.7 Coin Flipping in flask . . . 126
14.8 Requesting arguments in flask . . . 127
14.9 Requesting name in HTML . . . 128
14.10 Hello (name) in HTML . . . 128
14.11 Failure page in HTML . . . 129
14.12 Layout HTML . . . 130
14.13 Requesting name in HTML that extends layout . . . 130
14.14 Displaying name in HTML that extends layout . . . 130
14.15 Failure message in HTML that extends layout . . . 131
14.16 Tasks Application using Flask . . . 131
14.17 Layout for the tasks application . . . 132
14.18 Default page for the tasks application . . . 132
14.19 Add page for the tasks application . . . 133

15.1 Sessions in tasks application in Flask . . . 135
15.2 Register App using SQL database in Flask . . . 135
15.3 Layout of Register App . . . 136
15.4 Default page of Register App . . . 136
15.5 Registration page of Register App . . . 137
15.6 Apology page of Register App . . . 137

16.1 First few lines of java 140
16.2 Lists in java using Generics 140
16.3 Maps in java using Generics 140
16.4 Classes in java 141
16.5 Static Methods in java 141
16.6 Inheritance in Java Classes 142
16.7 Interfaces in Java Classes 142
16.8 Packages in Java 143
16.9 House class in Java 143
16.10 Example Android Application in Java 145