Notes

Introduction to Computer Science (CS50) on EdX

Sparsh Jain

November 18, 2020

Contents

1 Computational Thinking, Scratch **5**
- 1.1 Binary Number System . . . 5
- 1.2 Algorithms . . . 5
- 1.3 Time Complexity . . . 5
- 1.4 Pseudocode . . . 5
- 1.5 Scratch . . . 5

2 C **6**
- 2.1 Hello World . . . 6
- 2.2 Input . . . 6
- 2.3 Initialization . . . 8
- 2.4 Increment . . . 8
- 2.5 Conditionals . . . 8
- 2.6 Loops . . . 8
  - 2.6.1 While Loop . . . 8
  - 2.6.2 For Loop . . . 9
- 2.7 Additional Info . . . 9
  - 2.7.1 Datatypes . . . 9
  - 2.7.2 Functions . . . 9
  - 2.7.3 Placeholders . . . 10
  - 2.7.4 Arithmetic Operations . . . 10
- 2.8 Examples . . . 10
  - 2.8.1 Arithmetic . . . 10
  - 2.8.2 Conditional . . . 13
  - 2.8.3 Logical . . . 14
  - 2.8.4 Loop . . . 15
  - 2.8.5 Function . . . 16
- 2.9 Limitations . . . 21

3 Arrays **23**
3.1 Compiling . . . 23
3.1.1 Preprocessing . . . 23
3.1.2 Compiling . . . 23
3.1.3 Assembling . . . 23
3.1.4 Linking . . . 23
3.2 Debugging . . . 23
3.3 Casting . . . 24
3.4 Array . . . 24
3.5 String . . . 27
3.6 Command Line Arguments . . . 32

4 Algorithms **34**
4.1 Linear Search . . . 34
4.2 Binary Search . . . 34
4.3 Efficiency . . . 35
4.3.1 $\mathcal{O}$ Notation: . . . 35
4.3.2 Ω Notation: . . . 35
4.4 Examples . . . 36
4.4.1 Linear Search . . . 36
4.4.2 Bad Design . . . 37
4.4.3 Good Design - `typedef struct` . . . 38
4.5 Bubble Sort . . . 39
4.6 Selection Sort . . . 39
4.7 Better Bubble Sort . . . 40
4.8 Recursion . . . 40
4.9 Merge Sort . . . 43
4.9.1 Θ Notation . . . 43

List of Programs

2.1 Hello World in C 6
2.2 Hello User in C 7
2.3 int.c 11
2.4 float.c 11
2.5 parity.c 12
2.6 conditions.c 13
2.7 agree.c 14
2.8 cough0.c 15
2.9 cough1.c 15
2.10 cough2.c 16
2.11 cough3.c 17
2.12 positive.c 18
2.13 mario0.c 18
2.14 mario2.c 19
2.15 mario8.c 20
2.16 floats.c 21
2.17 overflow.c 21

3.1 casting 24
3.2 scores0.c 24
3.3 scores1.c 25
3.4 scores2.c 25
3.5 scores3.c 26
3.6 names.c 27
3.7 string0.c 28
3.8 string1.c 28
3.9 string2.c 29
3.10 uppercase0.c 30
3.11 uppercase1.c 31
3.12 argv.c 32
3.13 argv2.c 33
3.14 exit.c 33

4.1 Linear Search Pseudocode 34
4.2 Binary Search Pseudocode 34
4.3 Linear Search on numbers 36
4.4 Linear Search on names 37
4.5 Linear Search in a phonebook 38
4.6 Linear Search in phonebook with `typedef struct` 39
4.7 Iteration Pseudocode 40
4.8 Recursion Pseudocode 41
4.9 Iteration C code 41
4.10 Recursion C code 42
4.11 Merge Sort Pseudocode 43

Chapter 1

Computational Thinking, Scratch

1.1 Binary Number System

1.2 Algorithms

1.3 Time Complexity

1.4 Pseudocode

1.5 Scratch

This was only an introductory lecture. Click here for more details.

Chapter 2

C

2.1 Hello World

```c
#include <stdio.h>

int main(void)
{
    printf("Hello, World!\n");
}
```

Program 2.1: Hello World in C

Remark. Need to compile using a compiler like `clang` or `gcc`.

2.2 Input

Remark. In case of errors in compiling, start by trying to *fix* the first one, and so on.

Remark. Use `-lcs50` to link `cs50.h` header.

Remark. Use `make` to ease your life compiling!

```c
#include <cs50.h>
#include <stdio.h>

int main(void)
{
    string answer = get_string("What's your name?\n");
    printf("Hello, %s!\n", answer);
}
```

Program 2.2: Hello User in C

2.3 Initialization

```
int counter = 0;
```

2.4 Increment

```
counter = counter + 1;
counter += 1;
counter++; // Syntactic Sugar
```

2.5 Conditionals

```
if (x < y)
{
        printf("x is less than y!\n");
}
else if (x > y)
{
        printf("x is greater than y!\n");
}
else // if (x == y)
{
        printf("x is equal to y!\n");
}
```

2.6 Loops

2.6.1 While Loop

Infinite Loop

```
while(true)
{

}
```

Repeat

```
int i = 0;
while(i < 50)
```

```
{
        printf("Hello World!\n");
        i = i+1;
}
```

2.6.2 For Loop

```
for(int i = 0; i < 50; i += 1)
{
        printf("Hello World!\n");
}
```

2.7 Additional Info

2.7.1 Datatypes

Some of these (like `string`) are implemented in `cs50.h` library.

- `bool`
- `char`
- `double`
- `float`
- `int`
- `long`
- `string`
- ...

2.7.2 Functions

They are implemented in `cs50.h` library.

- `get_char`
- `get_float`
- `get_double`

- `get_int`
- `get_long`
- `get_string`
- ...

2.7.3 Placeholders

- `%c` for `char`
- `%f` for `float`
- `%i` for `int`
- `%li` for `long`
- `%s` for `string`

2.7.4 Arithmetic Operations

- `+`
- `-`
- `*`
- `/`
- `%`

2.8 Examples

2.8.1 Arithmetic

```c
#include <cs50.h>
#include <stdio.h>

int main(void)
{
    int age = get_int("What's your age?\n");
    // int days = age * 365;
    // printf("You are atleast %i days old.\n", days);
    printf("You are atleast %i days old.\n", age * 365);
}
```

Program 2.3: int.c

```c
#include <cs50.h>
#include <stdio.h>

int main(void)
{
    float price = get_float("What's the price?\n");
    // printf("Your total is %f.\n", price * 1.18);
    printf("Your total is %.2f.\n", price * 1.18);
}
```

Program 2.4: float.c

```c
#include <cs50.h>
#include <stdio.h>

int main(void)
{
    int n = get_int("n: ");

    if (n % 2 == 0)
    {
        printf("even.\n");
    }
    else
    {
        printf("odd.\n");
    }
}
```

Program 2.5: parity.c

2.8.2 Conditional

```c
// Conditions and relational operators

#include <cs50.h>
#include <stdio.h>

int main(void)
{
    // Prompt user for x
    int x = get_int("x: ");

    // Prompt user for y
    int y = get_int("y: ");

    // Compare x and y
    if (x < y)
    {
        printf("x is less than y\n");
    }
    else if (x > y)
    {
        printf("x is greater than y\n");
    }
    else
    {
        printf("x is equal to y\n");
    }
}
```

Program 2.6: conditions.c

2.8.3 Logical

```c
// Logical operators
#include <cs50.h>
#include <stdio.h>
int main(void)
{
    // Prompt user to agree
    char c = get_char("Do you agree?\n");
    // Check whether agreed
    if (c == 'Y' || c == 'y')
    {
        printf("Agreed.\n");
    }
    else if (c == 'N' || c == 'n')
    {
        printf("Not agreed.\n");
    }
}
```

Program 2.7: agree.c

2.8.4 Loop

```c
// Opportunity for better design

#include <stdio.h>

int main(void)
{
    printf("cough\n");
    printf("cough\n");
    printf("cough\n");
}
```

Program 2.8: cough0.c

```c
// Better design

#include <stdio.h>

int main(void)
{
    for (int i = 0; i < 3; i++)
    {
        printf("cough\n");
    }
}
```

Program 2.9: cough1.c

2.8.5 Function

```c
// Abstraction

#include <stdio.h>

void cough(void);

int main(void)
{
    for (int i = 0; i < 3; i++)
    {
        cough();
    }
}

// Cough once
void cough(void)
{
    printf("cough\n");
}
```

Program 2.10: cough2.c

```c
// Abstraction with parameterization

#include <stdio.h>

void cough(int n);

int main(void)
{
    cough(3);
}

// Cough some number of times
void cough(int n)
{
    for (int i = 0; i < n; i++)
    {
        printf("cough\n");
    }
}
```

Program 2.11: cough3.c

```c
// Abstraction and scope

#include <cs50.h>
#include <stdio.h>

int get_positive_int(void);

int main(void)
{
    int i = get_positive_int();
    printf("%i\n", i);
}

// Prompt user for positive integer
int get_positive_int(void)
{
    int n;
    do
    {
        n = get_int("Positive Integer: ");
    }
    while (n < 1);
    return n;
}
```

Program 2.12: positive.c

```c
// Prints a row of 4 question marks

#include <stdio.h>

int main(void)
{
    printf("????\n");
}
```

Program 2.13: mario0.c

```c
// Prints a row of n question marks with a loop

#include <cs50.h>
#include <stdio.h>

int main(void)
{
    int n;
    do
    {
        n = get_int("Width: ");
    }
    while (n < 1);
    for (int i = 0; i < n; i++)
    {
        printf("?");
    }
    printf("\n");
}
```

Program 2.14: mario2.c

```c
// Prints an n-by-n grid of bricks with a loop

#include <cs50.h>
#include <stdio.h>

int main(void)
{
    int n;
    do
    {
        n = get_int("Size: ");
    }
    while (n < 1);
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            printf("#");
        }
        printf("\n");
    }
}
```

Program 2.15: mario8.c

2.9 Limitations

```c
// Floating-point arithmetic with float

#include <cs50.h>
#include <stdio.h>

int main(void)
{
    // Prompt user for x
    float x = get_float("x: ");

    // Prompt user for y
    float y = get_float("y: ");

    // Perform division
    printf("x / y = %.50f\n", x / y);
}
```

Program 2.16: floats.c

```c
// Integer overflow

#include <stdio.h>
#include <unistd.h>

int main(void)
{
    // Iteratively double i
    for (int i = 1; ; i *= 2)
    {
        printf("%i\n", i);
        sleep(1);
    }
}
```

Program 2.17: overflow.c

Click here for more examples.

Chapter 3

Arrays

3.1 Compiling

3.1.1 Preprocessing

Expansion/Inclusion of header files, macros, etc.

3.1.2 Compiling

C code → Assembly code.

3.1.3 Assembling

Assembly code → Machine code.

3.1.4 Linking

Linking all relevent files.

3.2 Debugging

- Can use `help50` to understand error msgs in this course.
- Can use (poor man's) `printf`.
- Can use `debug50` for proper debugging (in this course).

Remark. Use `style50` for styling your code.

3.3 Casting

```c
// Prints ASCII codes

#include <stdio.h>

int main(void)
{
    char c1 = 'H';
    char c2 = 'I';
    char c3 = '!';
    printf("%i %i %i\n", c1, c2, c3);
}
```

Program 3.1: casting

3.4 Array

Follow through the following examples:

```c
// Averages three numbers

#include <cs50.h>
#include <stdio.h>

int main(void)
{
    // Scores
    int score1 = 72;
    int score2 = 73;
    int score3 = 33;

    // Print average
    printf("Average: %i\n", (score1 + score2 + score3) / 3);
}
```

Program 3.2: scores0.c

```c
// Averages three numbers using an array

#include <cs50.h>
#include <stdio.h>

int main(void)
{
    // Scores
    int scores[3];
    scores[0] = 72;
    scores[1] = 73;
    scores[2] = 33;

    // Print average
    printf("Average: %i\n", (scores[0] + scores[1] + scores[2]) / 3);
}
```

Program 3.3: scores1.c

```c
// Averages three numbers using an array and a constant

#include <cs50.h>
#include <stdio.h>

const int N = 3;

int main(void)
{
    // Scores
    int scores[N];
    scores[0] = 72;
    scores[1] = 73;
    scores[2] = 33;

    // Print average
    printf("Average: %i\n", (scores[0] + scores[1] + scores[2]) / N);
}
```

Program 3.4: scores2.c

```c
// Averages numbers using a helper function

#include <cs50.h>
#include <stdio.h>

float average(int length, int array[]);

int main(void)
{
    // Get number of scores
    int n = get_int("Scores:  ");

    // Get scores
    int scores[n];
    for (int i = 0; i < n; i++)
    {
        scores[i] = get_int("Score %i: ", i + 1);
    }

    // Print average
    printf("Average: %.1f\n", average(n, scores));
}

float average(int length, int array[])
{
    int sum = 0;
    for (int i = 0; i < length; i++)
    {
        sum += array[i];
    }
    return (float) sum / (float) length;
}
```

Program 3.5: scores3.c

3.5 String

`string` is just (or a little more) than an array of `char`s.

```c
// Stores names using an array

#include <cs50.h>
#include <stdio.h>
#include <string.h>

int main(void)
{
    // Names
    string names[4];
    names[0] = "EMMA";
    names[1] = "RODRIGO";
    names[2] = "BRIAN";
    names[3] = "DAVID";

    // Print Emma's name
    printf("%s\n", names[0]);
    printf("%c%c%c%c\n", names[0][0], names[0][1], names[0][2], names[0][3]);
}
```

Program 3.6: names.c

```c
// Prints string char by char, one per line

#include <cs50.h>
#include <stdio.h>

int main(void)
{
    string s = get_string("Input:  ");
    printf("Output: ");
    for (int i = 0; s[i] != '\0'; i++)
    {
        printf("%c", s[i]);
    }
    printf("\n");
}
```

Program 3.7: string0.c

```c
// Prints string char by char, one per line, using strlen

#include <cs50.h>
#include <stdio.h>
#include <string.h>

int main(void)
{
    string s = get_string("Input:  ");
    printf("Output: ");
    for (int i = 0; i < strlen(s); i++)
    {
        printf("%c", s[i]);
    }
    printf("\n");
}
```

Program 3.8: string1.c

```c
// Prints string char by char, one per line, using strlen, remembering string's

#include <cs50.h>
#include <stdio.h>
#include <string.h>

int main(void)
{
    string s = get_string("Input: ");
    printf("Output: ");
    for (int i = 0, n = strlen(s); i < n; i++)
    {
        printf("%c", s[i]);
    }
    printf("\n");
}
```

Program 3.9: string2.c

```c
// Uppercases a string

#include <cs50.h>
#include <stdio.h>
#include <string.h>

int main(void)
{
    string s = get_string("Before: ");
    printf("After:  ");
    for (int i = 0, n = strlen(s); i < n; i++)
    {
        if (s[i] >= 'a' && s[i] <= 'z')
        {
            printf("%c", s[i] - 32);
        }
        else
        {
            printf("%c", s[i]);
        }
    }
    printf("\n");
}
```

Program 3.10: uppercase0.c

```c
// Uppercases string using ctype library (and an unnecessary condition)

#include <cs50.h>
#include <ctype.h>
#include <stdio.h>
#include <string.h>

int main(void)
{
    string s = get_string("Before: ");
    printf("After:  ");
    for (int i = 0, n = strlen(s); i < n; i++)
    {
        if (islower(s[i]))
        {
            printf("%c", toupper(s[i]));
        }
        else
        {
            printf("%c", s[i]);
        }
    }
    printf("\n");
}
```

Program 3.11: uppercase1.c

3.6 Command Line Arguments

```c
// Printing a command-line argument

#include <cs50.h>
#include <stdio.h>

int main(int argc, string argv[])
{
    if (argc == 2)
    {
        printf("hello, %s\n", argv[1]);
    }
    else
    {
        printf("hello, world\n");
    }
}
```

Program 3.12: argv.c

```c
// Printing characters in an array of strings

#include <cs50.h>
#include <stdio.h>
#include <string.h>

int main(int argc, string argv[])
{
    for (int i = 0; i < argc; i++)
    {
        for (int j = 0, n = strlen(argv[i]); j < n; j++)
        {
            printf("%c\n", argv[i][j]);
        }
        printf("\n");
    }
}
```

Program 3.13: argv2.c

```c
// Returns explicit value from main

#include <cs50.h>
#include <stdio.h>

int main(int argc, string argv[])
{
    if (argc != 2)
    {
        printf("missing command-line argument\n");
        return 1;
    }
    printf("hello, %s\n", argv[1]);
    return 0;
}
```

Program 3.14: exit.c

Chapter 4

Algorithms

4.1 Linear Search

```
for i from 0 to n-1
        if ith element is 50
                return true;
return false;
```

Program 4.1: Linear Search Pseudocode

4.2 Binary Search

```
if no items
        return false;
if middle item is 50
        return true;
else if 50 < middle item
        search left half
else if 50 > middle item
        search right half
```

Program 4.2: Binary Search Pseudocode

4.3 Efficiency

4.3.1 $\mathcal{O}$ Notation:

Worst case scenario

$$\begin{aligned}
n^2 &: \mathcal{O}(n^2) \\
n \log_n n &: \mathcal{O}(n \log n) \\
n &: \mathcal{O}(n)\ (LinearSearch) \\
n/2 &: \mathcal{O}(n) \\
\log_2 n &: \mathcal{O}(\log n)\ (BinarySearch) \\
constant &: \mathcal{O}(1)
\end{aligned}$$

4.3.2 Ω Notation:

Best case scenario

$$\begin{aligned}
&\Omega(n^2) \\
&\Omega(n \log n) \\
&\Omega(n) \\
&\Omega(n) \\
&\Omega(\log n) \\
&\Omega(1)
\end{aligned}$$

Q: Better to have a really good $\mathcal{O}$ value or a really good Ω value?

A: $\mathcal{O}$, or even *average* case.

4.4 Examples

4.4.1 Linear Search

Numbers

```c
// Implements linear search for numbers

#include <cs50.h>
#include <stdio.h>

int main(void)
{
    // An array of numbers
    int numbers[] = {4, 8, 15, 16, 23, 42};

    // Search for 50
    for (int i = 0; i < 6; i++)
    {
        if (numbers[i] == 50)
        {
            printf("Found\n");
            return 0;
        }
    }
    printf("Not found\n");
    return 1;
}
```

Program 4.3: Linear Search on numbers

Names

```c
// Implements linear search for names

#include <cs50.h>
#include <stdio.h>
#include <string.h>

int main(void)
{
    // An array of names
    string names[] = {"EMMA", "RODRIGO", "BRIAN", "DAVID"};

    // Search for EMMA
    for (int i = 0; i < 4; i++)
    {
        if (strcmp(names[i], "EMMA") == 0)
        {
            printf("Found\n");
            return 0;
        }
    }
    printf("Not found\n");
    return 1;
}
```

Program 4.4: Linear Search on names

4.4.2 Bad Design

Correct/Working code but bad design!

```c
// Implements a phone book without structs

#include <cs50.h>
#include <stdio.h>
#include <string.h>

int main(void)
{
    string names[] = {"EMMA", "RODRIGO", "BRIAN", "DAVID"};
    string numbers[] = {"617-555-0100", "617-555-0101",
     ↪  "617-555-0102", "617-555-0103"};

    for (int i = 0; i < 4; i++)
    {
        if (!strcmp(names[i], "EMMA"))
        {
            printf("Found %s\n", numbers[i]);
            return 0;
        }
    }
    printf("Not found\n");
    return 1;
}
```

Program 4.5: Linear Search in a phonebook

4.4.3 Good Design - `typedef struct`

Using `typedef struct` for better design!

```c
// Implements a phone book with structs

#include <cs50.h>
#include <stdio.h>
#include <string.h>

typedef struct
{
    string name;
    string number;
}
person;

int main(void)
{
    person people[4];

    people[0].name = "EMMA";
    people[0].number = "617-555-0100";

    people[1].name = "RODRIGO";
    people[1].number = "617-555-0101";

    people[2].name = "BRIAN";
    people[2].number = "617-555-0102";

    people[3].name = "DAVID";
    people[3].number = "617-555-0103";

    // Search for EMMA
    for (int i = 0; i < 4; i++)
    {
        if (strcmp(people[i].name, "EMMA") == 0)
        {
            printf("Found %s\n", people[i].number);
            return 0;
        }
    }
    printf("Not found\n");
    return 1;
}
```

Program 4.6: Linear Search in phonebook with `typedef struct`

4.5 Bubble Sort

```
repeat n-1 times
        for i = 0 to n-2
                if ith and i+1th elements out of order
                        swap them
```

$\mathcal{O}(n^2)$

$\Omega(n^2)$

4.6 Selection Sort

```
for i from 0 to n-1
        find smallest item between ith item and last item
        swap smallest item and ith item
```

$\mathcal{O}(n^2)$

$\Omega(n^2)$

4.7 Better Bubble Sort

```
repeat until swap
        for i = 0 to n-2
                if ith and i+1th elements out of order
                        swap them
```

$\mathcal{O}(n^2)$

$\Omega(n)$

4.8 Recursion

```
1  Pick up phone book
2  Open to middle of phone book
3  Look at page
4  if Smith is on page
5          Call Mike
6  else if Smith is earlier in book
7          Open to middle of left half of book
8          Go back to line 3
9  else if Smith is later in book
10         Open to middle of right half of book
11         Go back to line 3
12 else
13         Quit
```

Program 4.7: Iteration Pseudocode

Can we do a better design?

```
Pick up phone book
Open to middle of phone book
Look at page
if Smith is on page
        Call Mike
else if Smith is earlier in book
        Search left half of book
else if Smith is later in book
        Search right half of book
```

```
else
        Quit
```

Program 4.8: Recursion Pseudocode

```c
// Draws a pyramid using iteration

#include <cs50.h>
#include <stdio.h>

void draw(int h);

int main(void)
{
    // Get height of pyramid
    int height = get_int("Height: ");

    // Draw pyramid
    draw(height);
}

void draw(int h)
{
    // Draw pyramid of height h
    for (int i = 1; i <= h; i++)
    {
        for (int j = 1; j <= i; j++)
        {
            printf("#");
        }
        printf("\n");
    }
}
```

Program 4.9: Iteration C code

```c
// Draws a pyramid using recursion

#include <cs50.h>
#include <stdio.h>
```

```c

void draw(int h);

int main(void)
{
    // Get height of pyramid
    int height = get_int("Height: ");

    // Draw pyramid
    draw(height);
}

void draw(int h)
{
    // If nothing to draw
    if (h == 0)
    {
        return;
    }

    // Draw pyramid of height h - 1
    draw(h - 1);

    // Draw one more row of width h
    for (int i = 0; i < h; i++)
    {
        printf("#");
    }
    printf("\n");
}
```

Program 4.10: Recursion C code

4.9 Merge Sort

```
if only 1 item
        return
else
        sort left half of items
        sort right half of items
        merge sorted halves
```

Program 4.11: Merge Sort Pseudocode

$\mathcal{O}(n \log n)$

$\Omega(n \log n)$

4.9.1 Θ Notation

When $\mathcal{O} = \Omega$!