

# Notes

Introduction to Computer Science (CS50) on EdX

Sparsh Jain

November 24, 2020

# Contents

<b>1</b>	<b>Computational Thinking, Scratch</b>	<b>5</b>
1.1	Binary Number System . . . . .	5
1.2	Algorithms . . . . .	5
1.3	Time Complexity . . . . .	5
1.4	Pseudocode . . . . .	5
1.5	Scratch . . . . .	5
<b>2</b>	<b>C</b>	<b>6</b>
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

<b>3</b>	<b>Arrays</b>	<b>23</b>
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
<b>4</b>	<b>Algorithms</b>	<b>34</b>
4.1	Linear Search . . . . .	34
4.2	Binary Search . . . . .	34
4.3	Efficiency . . . . .	35
4.3.1	$\mathcal{O}$ Notation: . . . . .	35
4.3.2	$\Omega$ Notation: . . . . .	35
4.4	Examples . . . . .	36
4.4.1	Linear Search . . . . .	36
4.4.2	Bad Design . . . . .	37
4.4.3	Good Design - <code>typedef struct</code> . . . . .	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	$\Theta$ Notation . . . . .	43
<b>5</b>	<b>Memory</b>	<b>44</b>
5.1	Hexadecimal . . . . .	44
5.2	Addresses . . . . .	44
5.2.1	Operators . . . . .	45
5.3	Pointers . . . . .	46
5.4	Strings . . . . .	47
5.5	String Comparision . . . . .	49
5.6	String Copy . . . . .	51
5.7	Malloc and Free . . . . .	52
5.8	Buffer Overflow . . . . .	52
5.9	Swap . . . . .	53
5.10	scanf . . . . .	55

5.11 File I/O . . . . .	56
-------------------------	----

# 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 <code>typedef struct</code> . . . . .	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
5.1	integer . . . . .	44
5.2	address of an integer . . . . .	45
5.3	address2.c . . . . .	45
5.4	accessing an address . . . . .	46
5.5	pointers . . . . .	46
5.6	strings . . . . .	47
5.7	strings are pointers . . . . .	47
5.8	strings are <code>char []</code> addresses are consecutive in arrays . . . . .	48
5.9	accessing characters in a string . . . . .	48
5.10	accessing characters in a <code>char *</code> . . . . .	48
5.11	comparing integers . . . . .	49
5.12	attempting to compare strings directly . . . . .	50
5.13	comparing strings properly . . . . .	50
5.14	attempting to copying strings directly . . . . .	51
5.15	copy strings properly . . . . .	52
5.16	buffer overflow . . . . .	53
5.17	naive attempt at swap . . . . .	53
5.18	swap . . . . .	54
5.19	scanning an integer . . . . .	55
5.20	scanning a string in uninitialized . . . . .	55
5.21	scanning a long string in small array . . . . .	56
5.22	files in c . . . . .	57
5.23	phonebook.csv . . . . .	57
5.24	check jpeg or not . . . . .	58

# Chapter 5

## Memory

Removing the training wheels `#include <cs50.h>` from now!

### 5.1 Hexadecimal

**Digits:** {1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}

**Ambiguity:** Prefix the number with 0x

### 5.2 Addresses

```
1 // Prints an integer
2
3 #include <stdio.h>
4
5 int main(void)
6 {
7     int n = 50;
8     printf("%i\n", n);
9 }
```

Program 5.1: integer

```

1 // Prints an integer's address
2
3 #include <stdio.h>
4
5 int main(void)
6 {
7     int n = 50;
8     printf("%p\n", &n);
9 }

```

Program 5.2: address of an integer

```

1 // Prints an integer via its address
2
3 #include <stdio.h>
4
5 int main(void)
6 {
7     int n = 50;
8     printf("%i\n", *&n);
9 }

```

Program 5.3: address2.c

## 5.2.1 Operators

& = Get the address

\* = Go to the address



## 5.3 Pointers

```
1 // Stores and prints an integer's address
2
3 #include <stdio.h>
4
5 int main(void)
6 {
7     int n = 50;
8     int *p = &n;
9     printf("%p\n", p);
10 }
```

Program 5.4: accessing an address

```
1 // Stores and prints an integer via its address
2
3 #include <stdio.h>
4
5 int main(void)
6 {
7     int n = 50;
8     int *p = &n;
9     printf("%i\n", *p);
10 }
```

Program 5.5: pointers

## 5.4 Strings

There are no strings. Strings are just pointers.

```
1 // Prints a string
2
3 #include <cs50.h>
4 #include <stdio.h>
5
6 int main(void)
7 {
8     string s = "EMMA";
9     printf("%s\n", s);
10 }
```

Program 5.6: strings

```
1 // Prints a string's address
2
3 #include <cs50.h>
4 #include <stdio.h>
5
6 int main(void)
7 {
8     string s = "EMMA";
9     printf("%p\n", s);
10 }
```

Program 5.7: strings are pointers

```
1 // Prints a string's address as well the addresses of its
  ↳ chars
2
3 #include <cs50.h>
4 #include <stdio.h>
5
6 int main(void)
7 {
8     string s = "EMMA";
9     printf("%p\n", s);
10    printf("%p\n", &s[0]);
```

```

11     printf("%p\n", &s[1]);
12     printf("%p\n", &s[2]);
13     printf("%p\n", &s[3]);
14     printf("%p\n", &s[4]);
15 }

```

Program 5.8: strings are `char []`  
addresses are consecutive in arrays

```

1  // Prints a string's chars
2
3  #include <cs50.h>
4  #include <stdio.h>
5
6  int main(void)
7  {
8      string s = "EMMA";
9      printf("%c\n", s[0]);
10     printf("%c\n", s[1]);
11     printf("%c\n", s[2]);
12     printf("%c\n", s[3]);
13 }

```

Program 5.9: accessing characters in a string

```

1  // Stores and prints a string's address via pointer arithmetic
2
3  #include <stdio.h>
4
5  int main(void)
6  {
7      char *s = "EMMA";
8      printf("%c\n", *s);
9      printf("%c\n", *(s+1));
10     printf("%c\n", *(s+2));
11     printf("%c\n", *(s+3));
12 }

```

Program 5.10: accessing characters in a `char *`

## 5.5 String Comparision

```
1 // Compares two integers
2
3 #include <cs50.h>
4 #include <stdio.h>
5
6 int main(void)
7 {
8     // Get two integers
9     int i = get_int("i: ");
10    int j = get_int("j: ");
11
12    // Compare integers
13    if (i == j)
14    {
15        printf("Same\n");
16    }
17    else
18    {
19        printf("Different\n");
20    }
21 }
```

Program 5.11: comparing integers

```
1 // Compares two strings' addresses
2
3 #include <cs50.h>
4 #include <stdio.h>
5
6 int main(void)
7 {
8     // Get two strings
9     string s = get_string("s: ");
10    string t = get_string("t: ");
11
12    // Compare strings' addresses
13    if (s == t)
14    {
15        printf("Same\n");
```

```

16     }
17     else
18     {
19         printf("Different\n");
20     }
21 }

```

Program 5.12: attempting to compare strings directly

```

1  // Compares two strings using strcmp
2
3  #include <cs50.h>
4  #include <stdio.h>
5
6  int main(void)
7  {
8      // Get two strings
9      string s = get_string("s: ");
10     string t = get_string("t: ");
11
12     // Compare strings
13     if (strcmp(s, t) == 0)
14     {
15         printf("Same\n");
16     }
17     else
18     {
19         printf("Different\n");
20     }
21 }

```

Program 5.13: comparing strings properly

## 5.6 String Copy

```
1 // Capitalizes a string
2
3 #include <cs50.h>
4 #include <ctype.h>
5 #include <stdio.h>
6 #include <string.h>
7
8 int main(void)
9 {
10     // Get a string
11     string s = get_string("s: ");
12
13     // Copy string's address
14     string t = s;
15
16     // Capitalize first letter in string
17     if (strlen(t) > 0)
18     {
19         t[0] = toupper(t[0]);
20     }
21
22     // Print string twice
23     printf("s: %s\n", s);
24     printf("t: %s\n", t);
25 }
```

Program 5.14: attempting to copying strings directly

```
1 // Capitalizes a copy of a string
2
3 #include <cs50.h>
4 #include <ctype.h>
5 #include <stdio.h>
6 #include <stdlib.h>
7 #include <string.h>
8
9 int main(void)
10 {
11     // Get a string
```

```

12     char *s = get_string("s: ");
13
14     // Allocate memory for another string
15     char *t = malloc(strlen(s) + 1);
16
17     // Copy string into memory
18     for (int i = 0, n = strlen(s); i <= n; i++)
19     {
20         t[i] = s[i];
21     }
22
23     // Capitalize copy
24     t[0] = toupper(t[0]);
25
26     // Print strings
27     printf("s: %s\n", s);
28     printf("t: %s\n", t);
29 }

```

Program 5.15: copy strings properly  
Just use strcpy(target, source) to copy strings.

## 5.7 Malloc and Free

**malloc:** Allocate Memory and return its address.

**free:** Free Memory (prevent leaking).

## 5.8 Buffer Overflow

```

1 // http://valgrind.org/docs/manual/quick-start.html
  ↳ #quick-start.prepare
2
3 #include <stdlib.h>
4
5 void f(void)
6 {
7     int *x = malloc(10 * sizeof(int));
8     x[10] = 0;

```

```

9   }
10
11  int main(void)
12  {
13      f();
14      return 0;
15  }

```

Program 5.16: buffer overflow

## 5.9 Swap

Pass by *value* vs pass by *reference*

```

1  // Fails to swap two integers
2
3  #include <stdio.h>
4
5  void swap(int a, int b);
6
7  int main(void)
8  {
9      int x = 1;
10     int y = 2;
11
12     printf("x is %i, y is %i\n", x, y);
13     swap(x, y);
14     printf("x is %i, y is %i\n", x, y);
15 }
16
17 void swap(int a, int b)
18 {
19     int tmp = a;
20     a = b;
21     b = tmp;
22 }

```

Program 5.17: naive attempt at swap



```

1  // Swaps two integers using pointers
2
3  #include <stdio.h>
4
5  void swap(int *a, int *b);
6
7  int main(void)
8  {
9      int x = 1;
10     int y = 2;
11
12     printf("x is %i, y is %i\n", x, y);
13     swap(&x, &y);
14     printf("x is %i, y is %i\n", x, y);
15 }
16
17 void swap(int *a, int *b)
18 {
19     int tmp = *a;
20     *a = *b;
21     *b = tmp;
22 }

```

Program 5.18: swap

## 5.10 scanf

```
1  // Gets an int from user using scanf
2
3  #include <stdio.h>
4
5  int main(void)
6  {
7      int x;
8      printf("x: ");
9      scanf("%i", &x);
10     printf("x: %i\n", x);
11 }
```

Program 5.19: scanning an integer

```
1  // Incorrectly gets a string from user using scanf
2
3  #include <stdio.h>
4
5  int main(void)
6  {
7      char *s;
8      printf("s: ");
9      scanf("%s", s);
10     printf("s: %s\n", s);
11 }
```

Program 5.20: scanning a string in uninitialized

```

1 // Dangerously gets a string from user using scanf
2
3 #include <stdio.h>
4
5 int main(void)
6 {
7     char s[5];
8     printf("s: ");
9     scanf("%s", s);
10    printf("s: %s\n", s);
11 }

```

Program 5.21: scanning a long string in small array

## 5.11 File I/O

```

1 // Saves names and numbers to a CSV file
2
3 #include <cs50.h>
4 #include <stdio.h>
5 #include <string.h>
6
7 int main(void)
8 {
9     // Open CSV file
10    FILE *file = fopen("phonebook.csv", "a");
11    if (!file)
12    {
13        return 1;
14    }
15
16    // Get name and number
17    string name = get_string("Name: ");
18    string number = get_string("Number: ");
19
20    // Print to file
21    fprintf(file, "%s,%s\n", name, number);
22

```

```

23     // Close file
24     fclose(file);
25 }

```

#### Program 5.22: files in c

```

1  Sparsh,6238-098-518

```

#### Program 5.23: phonebook.csv

```

1  // Detects if a file is a JPEG
2
3  #include <stdio.h>
4
5  int main(int argc, char *argv[])
6  {
7      // Check usage
8      if (argc != 2)
9      {
10         return 1;
11     }
12
13     // Open file
14     FILE *file = fopen(argv[1], "r");
15     if (!file)
16     {
17         return 1;
18     }
19
20     // Read first three bytes
21     unsigned char bytes[3];
22     fread(bytes, 3, 1, file);
23
24     // Check first three bytes
25     if (bytes[0] == 0xff && bytes[1] == 0xd8 && bytes[2] ==
26         0xff)
27     {
28         printf("Maybe\n");
29     }
30     else

```

```
30     {  
31         printf("No\n");  
32     }  
33  
34     // Close file  
35     fclose(file);  
36 }
```

Program 5.24: check jpeg or not