

# Notes

## Introduction to Computer Science (CS50) on EdX

Sparsh Jain

November 16, 2020

# Contents

<b>1</b>	<b>Computational Thinking, Scratch</b>	<b>4</b>
1.1	Binary Number System . . . . .	4
1.2	Algorithms . . . . .	4
1.3	Time Complexity . . . . .	4
1.4	Pseudocode . . . . .	4
1.5	Scratch . . . . .	4
<b>2</b>	<b>C</b>	<b>5</b>
2.1	Hello World . . . . .	5
2.2	Input . . . . .	5
2.3	Initialization . . . . .	7
2.4	Increment . . . . .	7
2.5	Conditionals . . . . .	7
2.6	Loops . . . . .	7
2.6.1	While Loop . . . . .	7
2.6.2	For Loop . . . . .	8
2.7	Additional Info . . . . .	8
2.7.1	Datatypes . . . . .	8
2.7.2	Functions . . . . .	8
2.7.3	Placeholders . . . . .	9
2.7.4	Arithmetic Operations . . . . .	9
2.8	Examples . . . . .	9
2.8.1	Arithmetic . . . . .	9
2.8.2	Conditional . . . . .	12
2.8.3	Logical . . . . .	13
2.8.4	Loop . . . . .	14
2.8.5	Function . . . . .	15
2.9	Limitations . . . . .	20

<b>3</b>	<b>Arrays</b>	<b>22</b>
3.1	Compiling . . . . .	22
3.1.1	Preprocessing . . . . .	22
3.1.2	Compiling . . . . .	22
3.1.3	Assembling . . . . .	22
3.1.4	Linking . . . . .	22
3.2	Debugging . . . . .	22
3.3	Casting . . . . .	23
3.4	Array . . . . .	23
3.5	String . . . . .	26
3.6	Command Line Arguments . . . . .	31

# List of Programs

2.1	Hello World in C . . . . .	5
2.2	Hello User in C . . . . .	6
2.3	int.c . . . . .	10
2.4	float.c . . . . .	10
2.5	parity.c . . . . .	11
2.6	conditions.c . . . . .	12
2.7	agree.c . . . . .	13
2.8	cough0.c . . . . .	14
2.9	cough1.c . . . . .	14
2.10	cough2.c . . . . .	15
2.11	cough3.c . . . . .	16
2.12	positive.c . . . . .	17
2.13	mario0.c . . . . .	17
2.14	mario2.c . . . . .	18
2.15	mario8.c . . . . .	19
2.16	floats.c . . . . .	20
2.17	overflow.c . . . . .	20
3.1	casting . . . . .	23
3.2	scores0.c . . . . .	23
3.3	scores1.c . . . . .	24
3.4	scores2.c . . . . .	24
3.5	scores3.c . . . . .	25
3.6	names.c . . . . .	26
3.7	string0.c . . . . .	27
3.8	string1.c . . . . .	27
3.9	string2.c . . . . .	28
3.10	uppercase0.c . . . . .	29
3.11	uppercase1.c . . . . .	30
3.12	argv.c . . . . .	31
3.13	argv2.c . . . . .	32
3.14	exit.c . . . . .	32

# 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

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5      printf("Hello, World!\n");
6  }
```

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!

```
1  #include <cs50.h>
2  #include <stdio.h>
3
4  int main(void)
5  {
6      string answer = get_string("What's your name?\n");
7      printf("Hello, %s!\n", answer);
8  }
```

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

```

1  #include <cs50.h>
2  #include <stdio.h>
3
4  int main(void)
5  {
6      int age = get_int("What's your age?\n");
7      // int days = age * 365;
8      // printf("You are at least %i days old.\n", days);
9      printf("You are at least %i days old.\n", age * 365);
10 }

```

Program 2.3: int.c

```

1  #include <cs50.h>
2  #include <stdio.h>
3
4  int main(void)
5  {
6      float price = get_float("What's the price?\n");
7      // printf("Your total is %f.\n", price * 1.18);
8      printf("Your total is %.2f.\n", price * 1.18);
9  }

```

Program 2.4: float.c

```

1  #include <cs50.h>
2  #include <stdio.h>
3
4  int main(void)
5  {
6      int n = get_int("n: ");
7
8      if (n % 2 == 0)
9      {
10         printf("even.\n");
11     }
12     else
13     {
14         printf("odd.\n");
15     }
16 }

```

Program 2.5: parity.c

## 2.8.2 Conditional

```
1 // Conditions and relational operators
2
3 #include <cs50.h>
4 #include <stdio.h>
5
6 int main(void)
7 {
8     // Prompt user for x
9     int x = get_int("x: ");
10
11     // Prompt user for y
12     int y = get_int("y: ");
13
14     // Compare x and y
15     if (x < y)
16     {
17         printf("x is less than y\n");
18     }
19     else if (x > y)
20     {
21         printf("x is greater than y\n");
22     }
23     else
24     {
25         printf("x is equal to y\n");
26     }
27 }
```

Program 2.6: conditions.c

### 2.8.3 Logical

```
1 // Logical operators
2 #include <cs50.h>
3 #include <stdio.h>
4 int main(void)
5 {
6     // Prompt user to agree
7     char c = get_char("Do you agree?\n");
8     // Check whether agreed
9     if (c == 'Y' || c == 'y')
10    {
11        printf("Agreed.\n");
12    }
13    else if (c == 'N' || c == 'n')
14    {
15        printf("Not agreed.\n");
16    }
17 }
```

Program 2.7: agree.c

## 2.8.4 Loop

```
1 // Opportunity for better design
2
3 #include <stdio.h>
4
5 int main(void)
6 {
7     printf("cough\n");
8     printf("cough\n");
9     printf("cough\n");
10 }
```

Program 2.8: cough0.c

```
1 // Better design
2
3 #include <stdio.h>
4
5 int main(void)
6 {
7     for (int i = 0; i < 3; i++)
8     {
9         printf("cough\n");
10    }
11 }
```

Program 2.9: cough1.c

## 2.8.5 Function

```
1  // Abstraction
2
3  #include <stdio.h>
4
5  void cough(void);
6
7  int main(void)
8  {
9      for (int i = 0; i < 3; i++)
10     {
11         cough();
12     }
13 }
14
15 // Cough once
16 void cough(void)
17 {
18     printf("cough\n");
19 }
```

Program 2.10: cough2.c



```

1  // Abstraction with parameterization
2
3  #include <stdio.h>
4
5  void cough(int n);
6
7  int main(void)
8  {
9      cough(3);
10 }
11
12 // Cough some number of times
13 void cough(int n)
14 {
15     for (int i = 0; i < n; i++)
16     {
17         printf("cough\n");
18     }
19 }

```

Program 2.11: cough3.c

```

1  // Abstraction and scope
2
3  #include <cs50.h>
4  #include <stdio.h>
5
6  int get_positive_int(void);
7
8  int main(void)
9  {
10     int i = get_positive_int();
11     printf("%i\n", i);
12 }
13
14 // Prompt user for positive integer
15 int get_positive_int(void)
16 {
17     int n;
18     do
19     {
20         n = get_int("Positive Integer: ");
21     }
22     while (n < 1);
23     return n;
24 }

```

Program 2.12: positive.c

```

1  // Prints a row of 4 question marks
2
3  #include <stdio.h>
4
5  int main(void)
6  {
7     printf("????\n");
8 }

```

Program 2.13: mario0.c

```

1  // Prints a row of n question marks with a loop
2
3  #include <cs50.h>
4  #include <stdio.h>
5
6  int main(void)
7  {
8      int n;
9      do
10     {
11         n = get_int("Width: ");
12     }
13     while (n < 1);
14     for (int i = 0; i < n; i++)
15     {
16         printf("?");
17     }
18     printf("\n");
19 }

```

Program 2.14: mario2.c

```

1  // Prints an n-by-n grid of bricks with a loop
2
3  #include <cs50.h>
4  #include <stdio.h>
5
6  int main(void)
7  {
8      int n;
9      do
10     {
11         n = get_int("Size: ");
12     }
13     while (n < 1);
14     for (int i = 0; i < n; i++)
15     {
16         for (int j = 0; j < n; j++)
17         {
18             printf("#");
19         }
20         printf("\n");
21     }
22 }

```

Program 2.15: mario8.c

## 2.9 Limitations

```
1  // Floating-point arithmetic with float
2
3  #include <cs50.h>
4  #include <stdio.h>
5
6  int main(void)
7  {
8      // Prompt user for x
9      float x = get_float("x: ");
10
11     // Prompt user for y
12     float y = get_float("y: ");
13
14     // Perform division
15     printf("x / y = %.50f\n", x / y);
16 }
```

Program 2.16: floats.c

```
1  // Integer overflow
2
3  #include <stdio.h>
4  #include <unistd.h>
5
6  int main(void)
7  {
8      // Iteratively double i
9      for (int i = 1; ; i *= 2)
10     {
11         printf("%i\n", i);
12         sleep(1);
13     }
14 }
```

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  $\rightarrow$  Assembly code.

#### 3.1.3 Assembling

Assembly code  $\rightarrow$  Machine code.

#### 3.1.4 Linking

Linking all relevant 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

```
1 // Prints ASCII codes
2
3 #include <stdio.h>
4
5 int main(void)
6 {
7     char c1 = 'H';
8     char c2 = 'I';
9     char c3 = '!';
10    printf("%i %i %i\n", c1, c2, c3);
11 }
```

Program 3.1: casting

## 3.4 Array

Follow through the following examples:

```
1 // Averages three numbers
2
3 #include <cs50.h>
4 #include <stdio.h>
5
6 int main(void)
7 {
8     // Scores
9     int score1 = 72;
10    int score2 = 73;
11    int score3 = 33;
12
13    // Print average
14    printf("Average: %i\n", (score1 + score2 + score3) / 3);
15 }
```

Program 3.2: scores0.c



```

1  // Averages three numbers using an array
2
3  #include <cs50.h>
4  #include <stdio.h>
5
6  int main(void)
7  {
8      // Scores
9      int scores[3];
10     scores[0] = 72;
11     scores[1] = 73;
12     scores[2] = 33;
13
14     // Print average
15     printf("Average: %i\n", (scores[0] + scores[1] + scores[2]) / 3);
16 }

```

Program 3.3: scores1.c

```

1  // Averages three numbers using an array and a constant
2
3  #include <cs50.h>
4  #include <stdio.h>
5
6  const int N = 3;
7
8  int main(void)
9  {
10     // Scores
11     int scores[N];
12     scores[0] = 72;
13     scores[1] = 73;
14     scores[2] = 33;
15
16     // Print average
17     printf("Average: %i\n", (scores[0] + scores[1] + scores[2]) / N);
18 }

```

Program 3.4: scores2.c

```

1  // Averages numbers using a helper function
2
3  #include <cs50.h>
4  #include <stdio.h>
5
6  float average(int length, int array[]);
7
8  int main(void)
9  {
10     // Get number of scores
11     int n = get_int("Scores: ");
12
13     // Get scores
14     int scores[n];
15     for (int i = 0; i < n; i++)
16     {
17         scores[i] = get_int("Score %i: ", i + 1);
18     }
19
20     // Print average
21     printf("Average: %.1f\n", average(n, scores));
22 }
23
24 float average(int length, int array[])
25 {
26     int sum = 0;
27     for (int i = 0; i < length; i++)
28     {
29         sum += array[i];
30     }
31     return (float) sum / (float) length;
32 }

```

Program 3.5: scores3.c

## 3.5 String

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

```
1  // Stores names using an array
2
3  #include <cs50.h>
4  #include <stdio.h>
5  #include <string.h>
6
7  int main(void)
8  {
9      // Names
10     string names[4];
11     names[0] = "EMMA";
12     names[1] = "RODRIGO";
13     names[2] = "BRIAN";
14     names[3] = "DAVID";
15
16     // Print Emma's name
17     printf("%s\n", names[0]);
18     printf("%c%c%c%c\n", names[0][0], names[0][1], names[0][2], names[0][3]);
19 }
```

Program 3.6: names.c

```

1  // Prints string char by char, one per line
2
3  #include <cs50.h>
4  #include <stdio.h>
5
6  int main(void)
7  {
8      string s = get_string("Input: ");
9      printf("Output: ");
10     for (int i = 0; s[i] != '\0'; i++)
11     {
12         printf("%c", s[i]);
13     }
14     printf("\n");
15 }

```

Program 3.7: string0.c

```

1  // Prints string char by char, one per line, using strlen
2
3  #include <cs50.h>
4  #include <stdio.h>
5  #include <string.h>
6
7  int main(void)
8  {
9      string s = get_string("Input: ");
10     printf("Output: ");
11     for (int i = 0; i < strlen(s); i++)
12     {
13         printf("%c", s[i]);
14     }
15     printf("\n");
16 }

```

Program 3.8: string1.c

```

1  // Prints string char by char, one per line, using strlen, remembering string's
2
3  #include <cs50.h>
4  #include <stdio.h>
5  #include <string.h>
6
7  int main(void)
8  {
9      string s = get_string("Input: ");
10     printf("Output: ");
11     for (int i = 0, n = strlen(s); i < n; i++)
12     {
13         printf("%c", s[i]);
14     }
15     printf("\n");
16 }

```

Program 3.9: string2.c

```

1  // Uppercases a string
2
3  #include <cs50.h>
4  #include <stdio.h>
5  #include <string.h>
6
7  int main(void)
8  {
9      string s = get_string("Before: ");
10     printf("After: ");
11     for (int i = 0, n = strlen(s); i < n; i++)
12     {
13         if (s[i] >= 'a' && s[i] <= 'z')
14         {
15             printf("%c", s[i] - 32);
16         }
17         else
18         {
19             printf("%c", s[i]);
20         }
21     }
22     printf("\n");
23 }

```

Program 3.10: uppercase0.c

```

1  // Uppercases string using ctype library (and an unnecessary condition)
2
3  #include <cs50.h>
4  #include <ctype.h>
5  #include <stdio.h>
6  #include <string.h>
7
8  int main(void)
9  {
10     string s = get_string("Before: ");
11     printf("After: ");
12     for (int i = 0, n = strlen(s); i < n; i++)
13     {
14         if (islower(s[i]))
15         {
16             printf("%c", toupper(s[i]));
17         }
18         else
19         {
20             printf("%c", s[i]);
21         }
22     }
23     printf("\n");
24 }

```

Program 3.11: uppercase1.c

## 3.6 Command Line Arguments

```
1  // Printing a command-line argument
2
3  #include <cs50.h>
4  #include <stdio.h>
5
6  int main(int argc, string argv[])
7  {
8      if (argc == 2)
9      {
10         printf("hello, %s\n", argv[1]);
11     }
12     else
13     {
14         printf("hello, world\n");
15     }
16 }
```

Program 3.12: argv.c



```

1  // Printing characters in an array of strings
2
3  #include <cs50.h>
4  #include <stdio.h>
5  #include <string.h>
6
7  int main(int argc, string argv[])
8  {
9      for (int i = 0; i < argc; i++)
10     {
11         for (int j = 0, n = strlen(argv[i]); j < n; j++)
12         {
13             printf("%c\n", argv[i][j]);
14         }
15         printf("\n");
16     }
17 }

```

Program 3.13: argv2.c

```

1  // Returns explicit value from main
2
3  #include <cs50.h>
4  #include <stdio.h>
5
6  int main(int argc, string argv[])
7  {
8      if (argc != 2)
9      {
10         printf("missing command-line argument\n");
11         return 1;
12     }
13     printf("hello, %s\n", argv[1]);
14     return 0;
15 }

```

Program 3.14: exit.c