Week 3

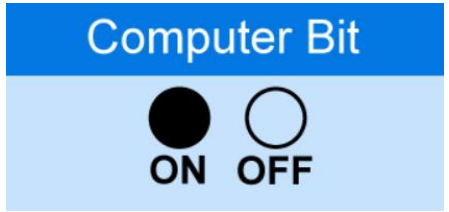
Hari Sethuraman

By the end of today

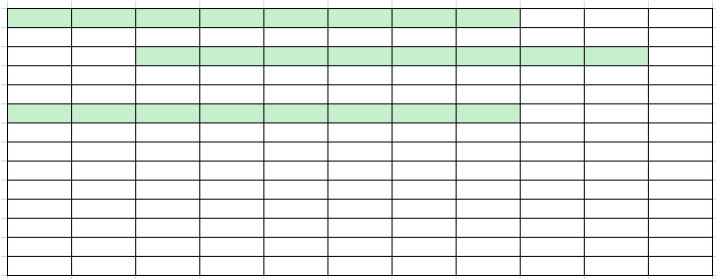
You should have a solid understanding of

- Datatypes
- Basics of C-Programming
  - ‘Hello World’ program
  - Variables
  - Get input from the user
  - Using this input to generate output
  - Program to add two numbers

Summary of last lesson



- Computers count using Base-2 notation
  - Powers of 2 rather than powers of 10
    - 0 or 1
  - 0 1 0 0 0 0 1 -> 65
- Each 0 or 1 is a bit
- 8 bits make up a byte
- The memory is a large grid of bits and stores groups of bits as bytes
- ASCII is used to convert Base-2 notation to characters and vice-versa.



Datatypes

- There are different types of information:
  - Numbers, Strings, Characters
- In order to let the computer know which type we are using, we use data types.

Char

Character: there are 256 different Characters. Represented using ‘ASCII’

dec	hex	oct	char	dec	hex	oct	char	dec	hex	oct	char	dec	hex	oct	char
0			NULL	32			space	64			@	96			`
1			SOH	33			!	65			A	97			a
2			STX	34			"	66			B	98			b
3			ETX	35			#	67			C	99			c
4			EOT	36			$	68			D	100			d
5			ENQ	37			%	69			E	101			e
6			ACK	38			&	70			F	102			f
7			BEL	39			'	71			G	103			g
8			BS	40			(	72			H	104			h
9			TAB	41			)	73			I	105			i
10			LF	42			*	74			J	106			j
11			VT	43			+	75			K	107			k
12			FF	44			,	76			L	108			l
13			CR	45			-	77			M	109			m
14			SO	46			.	78			N	110			n
15			SI	47			/	79			O	111			o
16			DLE	48			0	80			P	112			p
17			DC1	49			1	81			Q	113			q
18			DC2	50			2	82			R	114			r
19			DC3	51			3	83			S	115			s
20			DC4	52			4	84			T	116			t
21			NAK	53			5	85			U	117			u
22			SYN	54			6	86			V	118			v
23			ETB	55			7	87			W	119			w
24			CAN	56			8	88			X	120			x
25			EM	57			9	89			Y	121			y
26			SUB	58			:	90			Z	122			z
27			ESC	59			;	91			[	123			{
28			FS	60			<	92			\	124			|
29			GS	61			=	93			]	125			}
30			RS	62			>	94			^	126			~
31			US	63			?	95			_	127			DEL

www.overcoded.net

‘A’ -> 65 -> 0 1 0 0 0 0 0 1

‘!’ -> 33 -> 0 0 1 0 0 0 0 1

‘B’ -> ?

‘$’ ->

Integer

- Typically 2 or 4 bytes in C (represented using *int*)
  - If 2 bytes: value can be between -32,768 and 32,767 (Why?)
    - If we have 2 bytes or 16-bits, then the largest number we can represent is $2^{15} + 2^{14} + \ldots + 2^{1} + 2^{0} = 65533$. (we stack the bytes on top of each other)
    - However, because we want to represent both positive and negative integers, we split this number across the number line
      - Therefore, the minimum is -32768 and maximum is 32767
  - If 4 bytes: value can be between -2,147,483,649 and 2,147,483,649.
- Variations:
  - Unsigned-int: only positive integers (use when you don't need –ve numbers)
  - Long: int but with longer numbers between -9223372036854775808 and 9223372036854775807

Float

- Used to store decimal values:
  - Usually takes up 4 bytes.
  - Stores up to 6 decimal places
- Variations:
  - Double:
    - Takes up 8 bytes
    - Stores up to 15 decimal places
  - Long Double:
    - Takes up 10 bytes
    - Stores up to 19 decimal places

Boolean

- Typically takes up 1 byte
  - Stores a value of ‘True’ or ‘False’
  - Why not 1-bit? Because computers store info in bytes as a whole

How a C-program looks like

```
#include <stdio.h>
int main ()
{

}
```

Header

Main function

Insert code here
(between curly brackets)

Header and main function

- The Header (The <stdio.h>) imports a lot of useful features and functions into the program.
  - There are other headers that bring other functions, but <stdio.h> will be in every program
- The main function is a function that is called by default when the program starts
  - All the logical code you write must be inside a function
  - The main function always looks like: *int main() {…}*

Printf

- printf is a function, that prints out (or displays) whatever is passed to it:
  - Example:
    - printf(“Hi”);
      - Will print Hi
  - Including a ‘\n’ at the end of the string we want to print skips a line

Hello world

```
#include <stdio.h>

Int b = 3;
int main ()
{
        printf(“hello world\n”);
        int i = 0;
        I = I + 1
        b = b + 1
}

Int a()
{
        b = b + 2
}
```

REMEMBER

ALMOST ALL LINES OF CODE IN A FUNCTION END WITH A SEMI-COLON (;)

Variables

- We use Variables to store a value or piece of information
- In C, we need to specify the data-type of the variable
  - int for integer
    - Example: *int i = 5;*
  - *char* for character
    - *char c = 'e';*
    - Remember to use single quotations around characters (NOT double quotations)
  - *float* for float
    - *float f = 10.9141;*
- There are two types of variables:
  - Local variables: can only be used and referred to within the function they were created.
  - Global variables: are created outside a function and can be used and referred to within and outside all functions
  - We rarely use global variables, to avoid confusion in naming.

Printf + placeholder

- If we want to print the value of a variable using a printf statement, we need to use placeholders:
  - *printf("hello, %d", i);*

    i is the variable

    placeholder
  - *%d* for integer variables
  - *%c* for chars
  - *%f* for floats
  - …
- Multiple place holders:
  - *printf("hello, %d, %c", i, c);*
  - List the variables in order of the placeholder
  - Make sure the datatypes match or else the program will crash!

scanf

- To get the input from the user and store it in a variable, we use *scanf*
  - Scanf consists of two parts
    - Scanf(“ ”,);

&variable name

Placeholder with matching data-type

  - Example: get an integer input and store it in the variable *I*
    - *int i = 0;* **initializing the variable value**
    - *scanf(“%d”, &i);* **get input from the user**
  - The computer stores the input once the user pressers the enter button
  - *Why the ‘&’? We use this symbol in order to tell the computer the location in memory of *i*. if we just used *i*, we would pass the value of *i*, which is currently *0*.
  - This is not very useful, therefore, by using the ‘&’ before the variable name, we pass the location of the variable in memory rather than the value, so that the computer can directly write the input value there

Example

- (see the ide)
- We want to get a character from the user, and print the character out

Add two numbers

```
# include <stdio.h>

int main()
{
        int i1 = 0;
        int i2 = 0;
        scanf(“%d”, &i1);
        scanf(“%d”, &i2);

        int sum = i1 + i2;
        printf(“sum: %d\n”, sum);
}
```

Sources

- C - Data Types – Tutorialspoint
- In C how much space does a bool (boolean) take up? Is it 1 bit, 1 byte or something else? - Stack Overflow
- C library function - scanf() - Tutorialspoint