**C Programming**

Basics & Syntax

**Example 1**

#include <stdio.h> // part of the C standard library – input/output (printf(), scanf() etc…)

#include <stdlib.h> // part of the C standard library

#include “myConstants.h” // include a user-defined header file

#define TRUE 1 // a macro

#define FALSE 0 // another macro

Int main(void) {

Int aNumber;

Printf(“Please enter a number:\n”); // print to console

Scanf(“%d”, &aNumber); // get user to input to console

Printf(“You entered %d”, aNumber);

return 0;

}

**Overview**

* When creating your own library, you need a .h extension and a .c extension. The .h extension includes the blueprint and the .c extension includes the code related to the blueprint. The .h extension is the file you include.
* Every C programme requires a *main*() function, and only one! This is where the execution starts.
* Puts() prints strings to the terminal
* Printf() prints all datatypes to the terminal

**Variables**

*datatype varName = value*

float priceOfPetrol = 0.99;

int age = 28;

double radius = 2.34567; // a float but requires twice as much memory

*printf(“Price of petrol is %f\n”, priceOfPetrol);*

*// %f – float, %s – string, %d – integer/digit, %lf – double/long float, etc…*

**Comments**

// single line

/\* multiline \*/

**Date Types**

The type determines the amount of memory (bytes) a variable requires.

\* Use the built in function *sizeof()* to get the memory of each type, e.g. *sizeof(int)* \*

Integer

|  |  |  |
| --- | --- | --- |
| **Type** | **Size (bytes)** | **Range** |
| Char | 1 | -128 – 127 or 0 – 255 |
| Unsigned char | 1 | 0 - 255 |
| Signed char | 1 | -128 – 127 |
| Int | 2 or 4 | -32, 768 – 32,767 or  -2,147,483,648 – 2,147,483,647 |
| Unsigned int | 2 or 4 | 0 – 65,535 or 0 – 4,294,967,295 |
| Short | 2 | -32,768 – 32,767 |
| Unsigned short | 2 | 0 – 65,535 |
| Long | 4 | -2,147,483,648 – 2,147,483,647 |
| Unsigned long | 4 | 0 – 4,294,967,295 |

Floats

|  |  |  |
| --- | --- | --- |
| **Type** | **Size (bytes)** | **Range** |
| Float | 4 | 6 decimal places |
| Double | 8 | 15 decimal places |
| Long double | 16 | 19 decimal places |

Void

* No value (no data)
* Adds clarity to code and helps with debugging
* Void applies to functions, not variables, i.e. int main(void){} // no parameters

Boolean

Booleans are not defined in standard C, you can define them as 1 or 0 at the top of your files. E.g.

#define TRUE 1

#define FALSE 0

Int main(void) {

aBooleanVariable = TRUE;

printf(“Boolean is %d”, aBooleanVariable); // 1

return EXIT\_SUCCESS; // 1 – built in macro

}

Strings

There is also no string type in C, instead use:

Char[*lengthOfString*]

Char name[4] = “Ali”;

**Example 2**

#include <stdio.h>

#include <stdlib.h>

Int main(void) {

Char name[20];

Char c;

Printf(“How do you feel %s\n?”);

C = getChar(); // store user input to c variable, holds only 1 character

Printf(“Enter your full name…\n”);

Scanf(“%s”, &name);

Printf(“Your name is %s\n”, name);

Printf(“You feel: ”);

Putchar(c);

}

**Manipulating strings**

#include <string.h>

// one of the standard C libraries that provide many useful string manipulation functions

* Strcmp(str1, str2); // string compare
* Strcpy(dest, src); // copy string *src* to *dest*
* Memcpy(dest, src, n); // copy *n* chars from *src* to *dest*
* Strlen(str); // compute length of string

**Arithmetic operators**

Similar to JS except for increment/decrement. These can be used before as well as after a variable.

X = 10;

X++; // 10 – prints x as is, and then increments

++x; // 12 – increments first and then prints x with new value

#include <math.h> // library which includes further math functions

**Functions**

Without parameters

Void printName(void) { // void at the start is the return type, void as not returning

Char name[4] = “Ali”;

Printf(“%s\n”, name);

}

printName(); // invoke

With parameters

Void printName(charName[20]) {

Printf(“%s\n”, name);

}

printName(“Ali Issaee”);

Returning

Int getNum(int num) {

Return num;

}

Int randomNum = getNum(10);

Printf(“%d”, randomNum); // 10

**Typecasting**

Int num = 5;

Int numAsFloat = (float)num; // 5.000000

**Implement an external library**

Library.h

#include <stdio.h>

#include <stdlib.h>

// global variables

#ifndef TRANSACTIONS\_H\_ // if not defined already

#define TRANSACTIONS\_H\_

Float accountBalance, amount;

// global functions

Void initAmmount(void);

Void updateAccount(float);

#endif

Library.c

// create the full functions from library.c

Void initAmmount(void) {

// code here

}

Main.c

#include <stdio.h>

#include <stdlib.h>

#include “library.h”

**Conditionals**

If statements and switch statements are the same as in JS

**Loops**

For loops, while loops same as JS

When looping, its best to avoid recursion functions as with recursion, each time a function is called, a copy of the function is made therefore locking up memory

Do…while loop

Do {

//code

} while(*condition)*

**Single vs double quotes**

Single quotes identify a single character

Double quotes create a string literal

**Array**

Store a temporary list of items. Temporary as removed from memory when the programme ends.

Int scores[20]; // an array that holds a maximum of 20 integers

Char word[10]; // an array of characters (i.e. string)

Double data[50];

Another way to declare an array:

#define MAX\_ITEMS 20

Int numbers[MAX\_ITEMS];

Initializing arrays:

Int classrooms[5] = {15, 18, 10, 25, 26};

Float numbers[] = {5.5, 6.2, 2.7};

// note above size of array is unspecified, this is set to the length of the array itself, however it uses up more memory. It is more efficient to specify an amount, even if the amount is slightly higher than the values you give it, i.e. char name[6] = “Ali”;

**Example 3**

#define NUMBER\_OF\_CLASSROOMS 20

// function prototypes – these allow you to use functions before they are defined in full

Void printArray(char dataname[], int dataset[], int datalength);

Int main(void) {

Int classrooms[NUMBER\_OF\_CLASSROOMS] = {15, 18, 10, 23, 15};

Classrooms[3] += 2; // 23 now 25 as index 3 altered

printArray(“classrooms”, classrooms, NUMBER\_OF\_CLASSROOMS);

return EXIT\_SUCCESS;

}

Void printArray(char dataname[], int dataset[], int datalength) {

For (int I = 0; I < datalength; i++) {

Printf(“%s[%d] : %d\n”, dataname, I, dataset[i]);

}

}

Before array values are set, they are initially given the value 0. Because only 5 items where added in the array above, and the number was set to 20 (MAX\_NUMBER\_OF\_CLASSROOMS), the index[4] to [19] will have the values 0 when printing out to the terminal

**Two-dimensional arrays**

Int table[2][3] = {

{132, 147, 23},

{0, 76, 872}

};

Printf(“Row 1 column 2 contains %d\n”, table[1][2]); // 872

#define MAX\_ROWS 10

Void printable(int dataset[][MAX\_COLUMNS]) {

For (int I = 0; I < MAX\_ROWS; i++) {

For (int j = 0; j < MAX\_COLUMNS; j++) {

Printf(“%d”, dataset[i][j]);

}

Printf(“\n”);

}

}

All extra dimensions other than the first must have set values,

i.e. int array[*doesn’t need value*][MUST\_SET\_VALUE]

\* if you try to access an element in an array, the outcome is completely unpredictable, e.g. int ages[3] = {2, 7, 12}; printf(“%d\n”, ages[20]); << this will not break the system but the output could be any random integer \*

**Sorting data**

Bubble sort

In a two-dimension for loop:

1. Store the *i* element in a temporary variable
2. Assign the initial *i* variable as the *j* value
3. Put the tempo value into the *j* variable

Void bubbleSort(float dataset[], int lenght) {

Int swapped;

Float temp;

Int loopCounter = 0;

Do {

Swapped = FALSE;

For (int I = 0; I < length; i++) {

If (dataset[i-1] > dataset[i]) {

Temp = dataset[i];

Dataset[i-1] = dataset[i];

Dataset[i] = temp;

Swapped = TRUE;

}

}

} while (swapped);

}

**Example 4**

#include <time.h>

#define DATA\_SIZE 10

Void showBarChart(void);

Void showOneBar(int barLength);

Void setDataValues(void);

Int dataset[DATA\_SIZE];

Int main(void) {

setDataValues();

showBarChart();

return EXIT\_SUCCESS;

}

Void setDataValues(void) {

Srand(time(NULL)); // used to generate random number further down

For (int I = 0; I < DATA\_SIZE; i++) {

Dataset[i] = rand() % 50; // 0-49

}

}

Void showBarChart(void) {

For (int I = 0; I < DATA\_SIZE; i++) {

showOneBar(dataset[i]);

}

}

Void showOneBar(int barLength) {

Printf(“%d”, barLength);

For (int I = 0; I < barLength; i++) {

Putchar(‘o’);

}

Putchar(‘\n’);

}

**Quiz**

**Part 1**

1. Create a TRUE and FALSE macro
2. Include a user-defined header
3. What is the size of:
   1. Char
   2. Int
   3. Short
   4. Long
   5. Float
   6. Double
4. Create a string
5. Print one character to the console
6. Get a character from user and store in a variable
7. Get the name of a user, assign it to a variable and print it in the console

**Part 2**

1. Get the length of a string
2. Typecast an integer as a float
3. Create a function with arguments that explicitly returns a value
4. Use a do while loop in an example
5. Single vs double quotes
6. Create an array with float values
7. Use a function with its prototype defined before you use it

**Part 3**

1. Create a two-dimensional array and prints its values in a table-like format
2. Bubble sort an array from high to low
3. Create a bar chart in the console of different lengths using two-dimensional arrays and loops