**C EXTRAS**

**Arguments in main()**

Int main(int argc, char\*\* argv) {}

Argc = argument count

Argv= array of strings typed at the command prompt

By default argc is 1, and the argument is the name of the programme.

Int main(int argc, char\*\* argv) {

If (argc < 2) {

Puts(“Please specify a filename”);

Return (1);

}

Printf(“Examining file %s…\n”, argv[1]);#

Return (0);

}

**Static variable**

Use static variables to retain values used in side scopes.

Void print() {

Static int x = 1;

Printf(“%d\n”, x);

}

Int main() {

Print(); // 1

Print(); // 2

Print(); //3

}

**Bubble sort string**

Void bubbleSort(char\* str) {

Int len = strlen(str);

For (int i = 0; i < len -1; i++) {

For (int j = 0; j < len -1; j++) {

If (str[j] > str[j]) {

Char tmp = str[j];

Str[j] = str[j+1];

Str[j+1] = tmp;

}

}

}

Printf(”%s\n”, str);

}

Int main() {

Char name[10] = “bdac”;

bubbleSort(name);

return (0);

}

**Pointer tips**

Ampersand operator

& = address/memory location

Int i = 1;

Printf(“%p\n”, &i); // hex address

Pointer variables hold memory addresses without specifying the ampersand (like above):

Int i = 1, \*ptri;

Ptri = &i;

Printf(“%p == %p”, ptri, &i);

When working with arrays, you do not need to use ampersand when assigning a pointer. Arrays behave similar to pointers:

Int a[10], \*ptra;

Ptra = a; // no ampersand

Printf(“%p == %p == %p\n”, a, &a, ptra);

Arrays are treated like pointers. However, elements within an array are not, so you must specify ‘&’ if referencing individual elements in an array.

1. Ptra = &a[0];
2. Ptra = &a[2];
3. Ptra = a;

// 1 and 3 will be the same address as when an array is initialized, the address points to the first element in an array

Unary operator

\* unary operator

When \* is used with a pointer variable, it obtains the value of the address it points to. Without the \* operator, it refers to the address.

Int x = 0;

Int \*ptrx = &x;

Printf(“%d lives at %p”, \*ptrx, ptrx);

**Other**

Streams

Stderr, stdin, stdout macros are pointers to FILE types which correspond to the standard error, standard input, standard output streams.

Fprintf(stdout, “Hello World”); // print to the standard output (i.e. terminal/monitor)

Fflush(FILE \*stream) – flushes the output buffer of a stream. (fflush(stdout))

getc() & ungetc()

getc() – get the next character in a stream

ungetc() – retain the last input in a stream

*e.g.1*

FILE \*fp;

Fp = fopen(“test.json”, “r”); // where test.json is {“name”: “Ali”}

Char c = getc(fp);

Printf(“%c”, c); // “

C = getc(fp);

Printf(“%c”, c); // n

Ungetc(c, fp);

C = getc(fp);

Printf(“%c”, c); // n (not ‘a’ from name)

*e.g.2*

int main() {

char c = peek();

c = peek();

printf(“%c”, c); // new line

}

// if ungetc(c, stdin) above printf(), then the printf output would be the last user inputted character, not the newline.

Char peek(void) {

Char c = getchar();

Return c;

}

Fprintf

Sends formatted output to a stream

A stream could be a file or the standard output

*e.g.1*

Fprintf(stderr, “An error occurred”);

FILE \*fp;

Fp = fopen(“test.txt”, “w”);

Fprintf(fp, “hi”);

*e.g.2*

int age;

fscanf(stdin, “%d”, &age);

fprintf(stdout, “%d\n”, age);

**Strings**

Avoid scanf() and other functions that do not have buffer overflow protection, unless you know for certain that the input will always be of a specific format. Instead, use *fgets(char \*str, int buffer, FILE \*stream)*

Char name[20];

Fgets(name, 20, stdin);

The null terminator

The null character \0 is a character with the value 0. It is a reserved character used to signify the end of a string. Often called the null terminated string.

Char str1[] = “Hello”;

Char str2[] = {‘H’, ‘e’, ‘l’, ‘l’, ‘’o’, ‘\0’};

// these 2 strings above are the same

Strcmp

Strcmp(str1, str2); // compare 2 strings

If the strings match, then 0 is returned, i.e. no error.

You cannot compare 2 strings using ==

Example 1

Char \*removeBlanks(char\* input) {

Int I, j;

Char\* output = input;

For (i = 0, j = 0; i < strlen(input); i++, j++) {

If (input[i] != ‘ ’) {

Output[j] = input[i];

} else {

j--;

}

}

Output[j] = 0; // null terminator

Return output;

}

Example 2

Int main() {

Fputs(“> “, stdout);

For(;;) {

Char name[20];

Fscanf(stdin, “%s”, name);

If (strcmp(name, “Ali”) == 0) {

Fprintf(stdout, “Great name”);

}

Printf(“\t= %s\n> ”, name);

}

}

**Quiz**

Part 1

1. Name the 2 arguments used in the main function
2. Pass a filename to executable and check its existence, log to the error stream if filename has not been passed
3. Give an example of a static variable
4. Sort a string alphabetically using bubble sort
5. What are the 2 operators used with pointers
6. When do you not user an ampersand when assigning a value to a pointer
7. Print the memory location of a specific element in an array
8. Give an example of:
   1. Getc()
   2. Ungetc()
   3. Fprintf()
   4. Fscanf()
   5. Fputs()
   6. Fgets()
9. Log some text to an error file
10. What is the null terminator
11. Write an array equivalent string
12. Use strcmp() in an example, if a match is made then print success output

Part 2

1. Create a function which removes spaces in a string
2. Create a function where user can output to a screen until ctrl^c, everything inputted is outputted to the screen on carriage return
3. Create a fuzzbuzz function
4. Loop through a pointer array displaying the value and address of each element
5. Create a function to check if a number is odd or even
6. Create a function which changes the first letter of each word in a string with an underscore