Learning C

String: char string[] = “blablanbla”. Get stored as an array

[] for multiple, without for only 1 character.

Number: int – whole number, float, double – decimal number .. etc.

Print string: printf(“%s”, string)

Print number: printf(“%d”, number)

Print double: printf(“%f”, double)

Print char: printf(“%c”, char)

Datatypes: <https://en.wikipedia.org/wiki/C_data_types>

Print a memory address: printf(“%p”, &variable)

Math:

pow(2,3) = 2^3

sqrt(36) = 6

ceil(36.3456) = 37

floor(36.643) = 36

const int makes a constant, which can’t be changed down the line. Constants are written all caps.

Input from user: declare variable number int age, print text to ask, then scanf(“%d”, &age)

For a double, use scanf(“%lf”, &grade)

For a char, use scanf(“%c”, &grade)

For a name scanf("%s", name); no ampersand. But input Omar El Nahhas = output Omar. No spaces. Name is declared as char name[amount of characters];

Fgets(name, amount of characters, stdin) will assign a string to the name variable and you can limit the amount of characters. Stdin is standard input and is the panel where we give the information. This will input the whole Omar El Nahhas, BUT since you also click enter, the \n gets stored aswell.

Make an array: int numbers[] = {4, 8, 15, 16, 12};

Make a function: return value – void, int etc. Outside the int main() function.

Void sayHi(int variable)

{

}

Call function in main() function. sayHi(variable);

Make a prototype above the main(): int sayHi(int variable);

Struct: “object or class”, different datatypes.

Struct Student{

Char name [50];

Char major [50];

Int age;

Double gpa;

}

Now in the main() you can define as follows: struct Student student1;

This has created student1 as the special struct. Now you can give student1 information:

Student1.age = 22;

Student1.gpa = 3.2;

But for strings it’s different:

//Strcpy stands for stringcopy.

Strcpy(student1.name, “Omar”)

Strcpy(student1.major, “Engineering”)

And like this you can make all kinds of different “students” where the specific data can be assigned and retrieved. Unlike an array, which can only contain 1 datatype.

To accept command line arguments:

Int main(int argc, char \*argv[])

{

}

Argc is the number of arguments inside argv[]. And argv is an array which can be scrolled through to obtain values.

f.e.:

make test

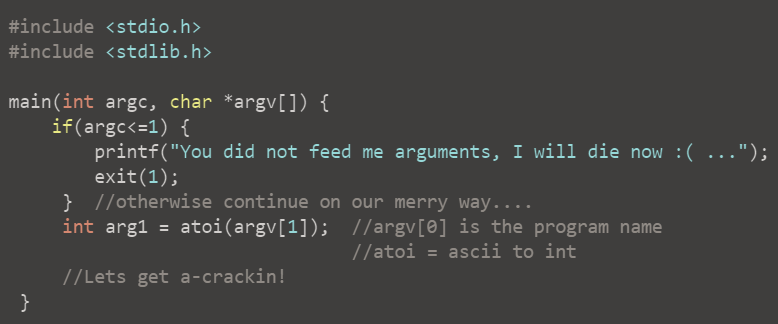
./test hallo bla

Argv[1] is “hallo”

Argv[2] is “bla”

Argv[3] is NULL

Accept number in command line argument:



String to number (ascii to int): use atoi function.

Command line:

Windows cmd, go to directory (cd) where the projectname.exe is:

c:\Users\Omar\Desktop\Tutorial\bin\Debug>tutorial 42



# Memory

Stored in RAM (Random access memory). This is the memory your computer is going to use when it’s running programs. C stores variables in physical memories, under memory addresses.

A string gets saved in memory with a \0 (NULL) at the end to indicate the end of a string.

## Pointers

Pointers is a type of data. Pointers store a physical memory address.

& in front of a variable will give the physical memory address of the variable.

A pointer variable is used to store the memory address of another variable in our program. Which is done as: (int, char, double) \*pAge = &age

You can print the address now: printf("age: %p\n", pAge);

Dereferencing the pointer can be done by: printf("age: %d\n", \*pAge);. This will give the actual value the address is holding. Datatype %p changes to %d because it is no longer an address. \*pAge is the same as \*&age

A string gets saved in memory with a \0 at the end to indicate the end of a string.

Iterate through the characters of a string:

How long it takes to run the program, use unix time program

**time** *./programname input*

# Files

FILE \*fpointer = fopen(“nameoffile.txt”, “r, w, a”);

Fclose(fpointer);

Fpointer creating a pointer to the physical file. FILE is a datatype. We create a file, and fpointer will show us the memory address where the file is stored.

Print to a file: fprintf(fpointer, “Whatever you want to type”)

Fpointer makes sure it gets written to the exact address the file is stored.

Read a file:

To read lines in a file, use:

Char line[255];

FILE \*fpointer = fopen(“nameoffile.txt”, “r”);

Fgets(variable to store the data it has read (line in this case), maximum size of the data (255), fpointer (address where our file is));

Reads the first line of the file, stores it in the variable.

If you call another fgets, it goes to line 2. You can do this for the whole file.

Length of an integer:

int counter;

for(counter = 0; number > 0; counter++)

{

number = number/10;

}

# digit in an integer:

Last digit is number%10 🡪 1234%10 is 123 rest 4. So the last digit is 4.

Second to last digit is (number/10)%10 🡪 (1234/10)%10 is 123,4%10 is 12 rest 3. Because it’s an integer, the decimals get cut off, hence you get the desired number!

String length:

<https://www.programmingsimplified.com/c-program-find-string-length>

#include <string.h>

Integer to ascii (itoa)

Ascii to integer (atoi)

If you do:

Int number = ascii + int 🡪 will return the integer value of the ascii letter + the int!

# Libraries

#include <stdio.h> //for gets

#include <stdlib.h> //for atoi, itoa

#include <math.h>

#include <string.h> //for strlen

#include <ctype.h> //for isalpha

# Other

Only in header files:

#ifndef checks whether the given token has been #defined earlier in the file or in an

included file; if not, it includes the code between it and the closing #else or, if no #else is present, #endif statement.

#ifndef is often used to make header files idempotent by defining a token once the file has been included

and checking that the token was not set at the top of that file.

///

#define is a C-directive which is also used to define the aliases for various data types similar to typedef

but with the following differences −

typedef is limited to giving symbolic names to types only where as #define can be used to define alias

for values as well, q., you can define 1 as ONE etc.

typedef interpretation is performed by the compiler whereas #define statements are processed by the pre-processor.

///

#if defined(CREDIT)

credit();

#elif defined(DEBIT)

debit();

#else

printerror();

#endif

The function call to credit is compiled if the identifier CREDIT is defined. If the identifier DEBIT is defined,

the function call to debit is compiled. If neither identifier is defined, the call to printerror is compiled.

Note that CREDIT and credit are distinct identifiers in C and C++ because their cases are different.

https://docs.microsoft.com/en-us/cpp/preprocessor/hash-if-hash-elif-hash-else-and-hash-endif-directives-c-cpp?view=vs-2019

# Operators

bitwise |= OR operator.

byte b = 0x0A;

b |= 0x50;

after this b = 0x5A

see <https://www.tutorialspoint.com/cprogramming/c_operators.htm>

//

Bit shift >> or <<

A = 60; //binary is 111100

A >> 1 // shift A 1 bit to the right (so 011110), a = 30

A >> 2 //shift A 2 bits to the right (so 001111), a = 15

A >> 3// Shift A 3 bits to the right (so 000111), a = 7