## 10.6.2022

主函数首先被调用（别的函数放在主函数里）

句后加分号

函数有大括号

printf

#include <stdio.h>

Void -> 空

gcc (-o 新名字) xxx.c （很多其他用法在gcc的url，moodle Week1 Slides2）

.\hello.exe

变量和函数的type

缩进用空格，不是tab！

## 10.7.2022

.h file专门写declaration

int 有31bits 代表数字 超了的话，就只算能算的

## 10.10.2022

用printf的时候最好加\n, 要不然需要打印的东西容易存在buffer里，打印不出来，下一个执行命令行也会挤在一起

没给变量的initial value的话，random value

|| or && and

(not) a !a

小数不要直接3.123==（3.0+0.123）

对两个要比较的小数求差，差<0.0001（这个取决于需要的精确度）

这么比而不是直接==

++x （先加再比） x++（先比再加）

## 10.17.2022

#define SIZE 35

const int size = 35;

变量能用constant就constant，别人改不了

printf(“Enter xxx :”);

scanf(“%d”, &a); 可以一次扫多个，中间空格即可

Pointer of a

如果数据类型不对，scanf会直接返回，存0，东西呆在buffer （后续可clean input buffer）

对于scanf来说：

%d integer （也可以用%i）

%lf double

%ld long

用scanf以后，printf的占位符也是这个！

Scanf引号里的东西代表它要找的pattern

返回值是成功扫描的数量

clean input buffer:

char ch;

while ((ch = getchar()) != '\n' && ch != EOF);

char message[] 可接受一串字符串

## 10.20.2022 (slides on arrays week 3)

array初始值为0

可传到function当参数，但是function可以直接改原array，不用pointer

不能return local array，可以function外写一个，用function更新

2D array （可以更多）

int m[3][4] = {

{1, 2, 3, 4},

{4, 5, 6, 7},

{8, 9, 10, 11}

};

m[1][3]

string内容后面总有一个标记到此为止的\0，这属于一个character

string function，<string.h>, slides里找具体

char name[50];

scanf(“%49s”, name);

或者fgets(name, sizeof(name), stdin);

能减少重复代码就尽量减少

正常是先定义function，后使用。但是如果有declaration，只要后面有定义就可以了，顺序没那么重要

## 10.27.2022 Week 4 slides

Scope, lifetime. 当不运行那块的时候，那块的变量memory deallocated. （但其实在被重新利用之前，原来的东西还写在那里）

嵌套循环，里面的循环可以用外面循环的变量

int \*p

int \*\*p, pointer to pointer

int (\*functionPointer) (int) pointer to function

int n = 10;

int \*p = &n; （变量就叫p）

int t = \*p;

\*p = 20

int \*p = myArray 自然指向array的第一项

\*(p+1) = 30; array第二项，只能加减，不能乘除

比如Array, string length, string copy

Dynamic memory allocate之后要记得free掉

Calloc， malloc， realloc

struct 有点像class

Typedef

robot->x

atoi 字符串转整数 （在stdlib.h里）

在commend line输入的整数也是字符串类型

## 11.3.2022 Week 5 slides

Bitwise operators, printing value in binary

Virtual memory 等定义

Memory分成segments，c只能看见连续的一串virtual memory, MMU (hardware，所以快) map the virtual memory to physical memory

一个程序是一个process，可以多个一起跑，互相memory是分开的，看不到别的process的memory

因为virtual pages 多，所以暂时不用的page swap到disk里去，但是会导致变慢

Root, rings

Threads

Global variable - dependency

File scope - static keyword

Code里用其他source file的变量，加extern，它在原来那里不能是static

Abstraction (kind of encapsulation)

API，.h declaration

.c 具体function

library