**Libraries**

#include <stdio.h> //standard input output  
#include <math.h> //math functions  
#include <stdbool.h> //standard boolean   
#include <string.h> //for string manipulation ( strlen() )  
#include <windows.h> //sleep function: Sleep(1000); in miliseconds  
#include <unistd.h> //sleep for linux/mac: sleep(1); in seconds  
#include <time.h> //for random int  
#include <stdlib.h> //for random int and malloc()

**Data types**

int a = 5; //max size of 4 bytes   
float b = 5.5; //4 bytes, 7 decimal points (single precision)  
double c = 3.14159265358979; //8bytes, 15 decimal points (double precision)  
char d = ‘A‘; //1 byte singe charakter  
char name[] = ¨Ahoj jak je¨ //array of characters (string) we use double quotes  
bool isTrue = true; //1 byte needs its own library and can be true or false

**Format specifier**

Special tokens that begin with % symbol followed by a charakter that specifies the data type and optional modifiers(width, precision, flags). They control how data is displayed and interpreted.

printf(“Je mi %d“, 18); //%d expects int after the string  
printf(“Cena %f“, 19.548); //%f expects float  
printf(¨Cena %lf¨, 19.54545454545) //%lf expects long float (double)  
printf(“Známka %c“, ¨A¨); //%c expects charakter (char)  
printf(“Text : %s“, ¨Ahoj¨); //%s expects string  
printf(“Text : %p“, &age); //%p expects pointer address

**modifiers**

printf(¨%4d¨, 10); //prints 10 but right justified (two spaces and 10)  
printf(¨%-4d¨, 10); //prints 10 but left justified (10 and two spaces)  
printf(¨%04d¨, 10); //prints 10 but right justified (two 0 and 10)  
printf(¨%+d¨, 10); //prints 10 but with + or – sign (depends on value of int 10 or -10)

printf(¨%.2f¨, 10.5481); //prints 10.54 so the 0.2 means 2 decimal places (automatically rounds numbers so 19.99 with %.1 would be 20.0)  
  
printf(¨%+7.2f¨, 10.5481); //prints +10.55 right justified

**Operators**

| **Operator** |  |  | **Name** | **Description** | **Example** |
| --- | --- | --- | --- | --- | --- |
| + |  |  | Addition | Adds two operands | a + b |
| - |  |  | Subtraction | Subtracts right from left operand | a - b |
| \* |  |  | Multiplication | Multiplies two operands | a \* b |
| / |  |  | Division | Divides left by right operand | a / b |
| % |  |  | Modulus | Remainder of division | a % b |
| ++ |  |  | Increment | Increases value by 1 | ++a, a++ |
| -- |  |  | Decrement | Decreases value by 1 | --a, a-- |
| && |  |  | AND | Bool | a && b |
| || |  |  | OR | Bool | a || b |
| ! |  |  | NOT | Bool | !a |

!!!! Be carefull with /  
float a = b / c //in this context c cant be int type must be float (integer division)

**Variables**

int a = 0; //sets value a to 0  
int b; //declares b as int so we can later use it and assign its value BUT this can lead to undefined behaviour because b can exits in memory with some value so its good to assing it beforehand like a or define its value before we use it  
  
float gpa = 0.0f; //tells the compiler that its float (0.0 would be double)  
char grade = ‘\0‘; //null terminator (clears variable only for char)  
char name[30] = ¨¨; //empty string with max size of 30 bytes (30 chars)  
const int A = 12; //creates a constant int variable A

**Type casting**

**User input**

int age = 0;  
printf(¨Enter age : ¨);  
scanf(¨%d¨, &age); //&age the & means address of so it returns memmory address  
scanf(¨ %c¨, &grade); //space before %c (without it the grade would be set to \n because of the first scanf (input buffer contains \n before we input our character))  
  
char name[30] = ‘ ‘;  
getchar(); //clears the \n symbol becaouse fgets cant do that  
fgets(name, sizeof(name) , stdin); // scanf stops at white spaces so we use fgets() with values of variable to input into, size of the vairable (we use function to get it), stdin = standard input

**Strings**

char name[] = “Dave“;  
name[strlen(name)-1] = “\0“; // strlen() vrátí délku stringu a poté nastavíme poslední index (e) na null terminator

**Math**

sqrt(9); //square root of 9  
pow(2,3); //2 raised to the power of 3 (8)  
round(3.14); //3  
ceil(3.14); //4  
floor(3.14); //3  
abs(-4); //4  
log(3); //1.098612 natural logarithm (e to the power of x = 3)  
sin(45); //0.850904 sin with input in radians  
cos(45); //0.525322 cos with input in radians  
tan(45); //1.617591 radians input

**If statements**

int a = 18;   
if (a >= 65){  
 printf(“Senior“);  
}else if (a >= 18){  
 printf(“Adult“)  
}else{  
 printf(“Kid“);  
}

**Switch Case**

int a = 1;  
switch(a){  
 case 1:  
 print(“Monday“);  
 break;  
 case 2:  
 print(“Tuesday“);  
 break;  
 default:  
 print(“Other day“);  
}

**Functions**

**Declaring function return**

void a(){} //expected to return nothing (doesnt have return statement)  
int a(){} //expected to return int  
float a(){} // expected to return float  
double a(){} // expected to return double  
bool a(){} // expected to return bool  
char a(){} //expected to return char

void greet (char name[], int age) {  
 printf(“Heyyyyy %s you are %d“, name, age);  
}  
greet(“Dave“, 25);

**Function prototype**

By prototyping a fuction we can use it and define it later (under the main code)

void greet (char name[], int age);  
int main(){  
 greet(“Dave“, 18);  
 return 0;  
}  
void greet (char name[], int age) {  
 printf(“Heyyyyy %s you are %d“, name, age);  
}

**Do-While loop**

int a = 0;  
do{  
 printf(“%d“, a);  
 a++;  
}while(number <= 10);

**While loop**

int a = 0;  
while(a<=10){  
 printf(“%d“, a);  
 a++;  
}

**For loop**

We can use break; and continue; like in python

for (int i = 0 ; i < 10 ; i++){  
 printf(“%d“,i);  
}

**Pseudo Random**

srand(time(NULL)); //creates a base seed for rand with current time  
rand()  
  
int max = 100;  
int min = 1;  
int randomNum = (rand() % (max – min + 1 )) + min; //formula for rand range  
  
RAND\_MAX //this is a constant with tahe maximum int rand can return

**Array**

int number[] = {10,15,20,54,81};

if we want the len of an element we can use

int len = sizeof(number) / sizeof(number[0]); // 20/4 = 5  
  
int a[5] = {0}; //this declares a array of 5 elements all being int = 0

**2D array**

int number[][2] = {{1,2,3}, {4,5,6}};   
//you must always declare all sizes (except the first one)

**Array of strings**

char fruits[][10] = {“Apple“, “Banana“, “Coconut“};  
//just line 2D array because strings are already arrays of chars

**Ternary operator**

Shorted if else statements (?)  
  
int x = 5;  
int y = 6;  
int max = (x > y) ? x : y //if x>y max=x else max=y

**Typedef**

Typedef int Number; //we change int to Number (definition)  
int main(){  
 Number x = 3;  
 Number y = 4;  
 Number z = x+y;  
}

**Enums**

enum Day{SUNDAY, MONDAY, TUESDAY};  
 //we can set its values by SUNDAY=6 (default starts values on 0)   
int main(){  
 Day today = SUNDAY;   
} //we create a variable today with value from Day[SUNDAY] so 0

**Structs**

Just like OOP in Python

struct Student {  
 char name[50];  
 int age;  
 float gpa;  
 bool isFullTime;  
};  
int main(){  
 struct Student student1 = {“Dave“, 18, 2.5, true};  
 printf(“%s“, student1.name);  
 strcpy(student1.name, “Tom“); //we change the name to Tom  
}

**Array of Structs**

struct Car {  
 char model[50];  
 int year;  
 int price;  
};

int main(){  
 struct Car cars[] = {  
 {“Mustang“, 2025, 32000};  
 {“Corvette“, 2026, 68000};  
 {“Challenger“, 2024, 29000};  
 };  
 printf(“%s %d $%d“, car[0].model, car[0].year, car[0].price);

}

**Pointers**

Stores address of another variable

int age = 25;  
int \*pAge = &age; //creates a pointer (type of reference variable)  
(\*pAge)++; //we turn the pointer back into a variable (on the pointer address)  
 and increase its value by one

**File management**

**Create a file and write**char text[] = “Ahoj jak \nJe“;   
  
FILE \*pFile = fopen(“output.txt“, “w“);  
if(pFile == NULL){  
 printf(“Error opening file“);  
 return 1;  
}  
  
fprintf(pFile, “%s“, text);  
  
fclose(pFile);

**Open a file and read**char buffer[1024] = {0};  
FILE \*pFile = fopen(“output.txt“, “r“);  
if(pFile == NULL){  
 printf(“Error opening file“);  
 return 1;  
}

while(fgets(buffer, sizeof(buffer), pFile) != NULL){  
 printf(“%s“, buffer);  
}  
  
fclose(pFile);

**MALLOC()**

int number = 5;  
char \*grades = malloc( number\*sizeof(char) ); //we allocate a pointer grades with the  
 size calculated from number (in bytes)  
free(grades); //return rented space from heap  
grades = NULL; //avoids dangling pointers (so we dont use it accidentally)

**CALLOC()**

int number = 4;  
int \*scores = calloc(number, sizeof(int)); //like malloc but sets all values to 0  
free(scores);  
scores = NULL;

**REALLOC()**

**Reallocate (extend) pointer to array**

int number = 5;  
float \*prices = calloc( number, sizeof(float) );

int new\_number = 7;  
float \*temp = realloc( prices, (new\_number\*sizeof(float)) );  
prices = temp;  
temp = NULL;

free(prices);  
prices = NULL;