The C programming language

- Developed in the 1970's at Bell Labs
- For the purpose of building the Unix Operating system
  - Linux & MacOS are descendents of Unix.
- Allows access to the hardware
  - for "systems programming"
  - the vast majority of software that runs computing devices is written in C.

C is not object-oriented

- no classes
- no objects
- no "methods"
  - C has "functions"

Simple built-in types:

int - 4 byte signed integer (32 bits)

char - 1 byte signed integer (8 bits)

long int } 8 byte signed integer
long } (64 bits)

2^{64} possible integers, half of them are negative

Quick question:

If there are 2^{64} possible integers and half of them are negative, are there 2^{32} possible negative integers?

NO!

Since $2^{64} = 2 \times 2^{63}$:

There are 2^{63} negative numbers.

2^{32} is not half of 2^{64}, it is the square root of 2^{64}

$$2^{64} = 2^{32} \times 2^{32}$$

because $2^{(n+m)} = 2^n \times 2^m$

Continuing with C types:

short int
short
} 2 byte signed integer (16 bits)

Unsigned integers:
- always treated as non-negative.

unsigned char - 1 byte
unsigned short - 2 bytes
unsigned int - 4 bytes
unsigned long - 8 bytes

Floating point numbers:

float - 32 bit floating point number
- approximates a real number

double - 64 bit floating point number

No boolean type

- $\emptyset$ represents false
- any non-zero number represents true.

```
int x = 7;
 if (x)
    printf("yes");
else
   printf("No");
```

prints "yes"

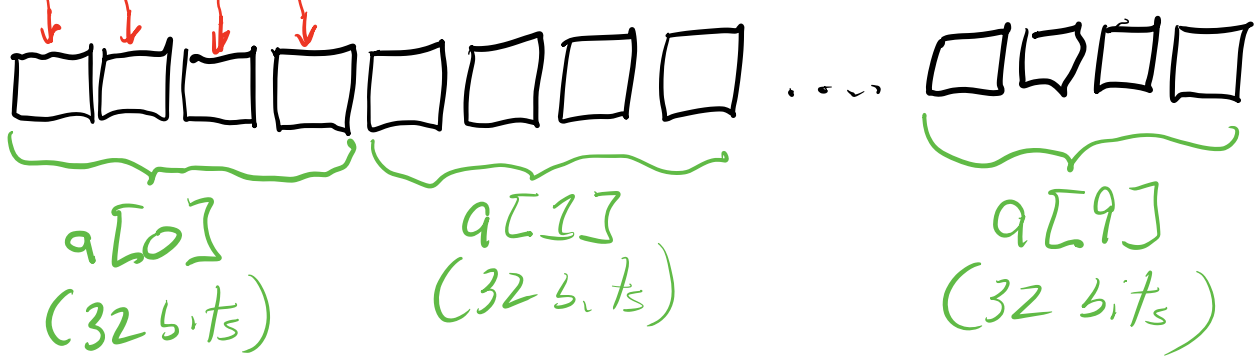
Arrays - sequence of contiguous locations in memory (block of memory)

int a[10];

allocates a block of memory big enough to hold 10 ints.

40 bytes: ten 4-byte ints.

each is 1 byte



C does not have "bounds checking"

- no checking to make sure you are accessing elements of the declared array
- you can go off the end of the array.

```
int a[10];
a[11]=50;   // allowed
            // but a bug.
            // Overwrites data
            // outside the array.

for(int i=0; i<12; i++)
    a[i]=i*2;
```

too many iterations!

There's no separate string type.

- a string is just an array of chars.

```
char s[10];  //enough space
             // for a string
             // of 9 characters
             // plus a termination
             // value of Ø.
```

A C program is a collection of functions.

- the first function to run is called "main".

```
int main()   //no params in its simplest form
{
  printf("Hello World\n");
}
```

- Variables are either <u>global</u> or <u>local</u>.
  - global: defined outside a function
    - visible everywhere in the file, starting from where they are declared.
  - local: defined inside a function
    - only visible inside the function.

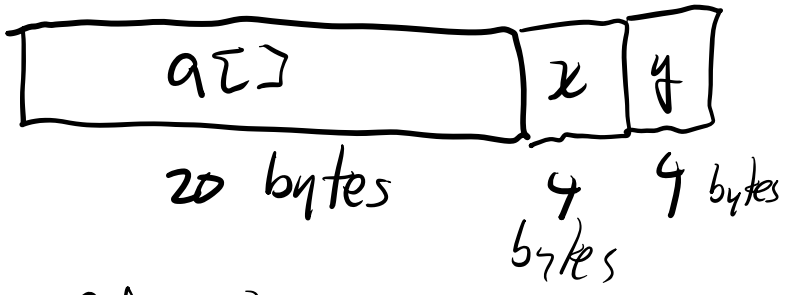
See "bad.c" in the programs written in class (on Brightspace)

- It declares global variables as follows:

```
int a[5] = {0, 1, 2, 3, 4};
int x = 7;
int y = 8;
```

- The compiler allocated them in contiguous memory locations:

memory:



So writing off the end of a[] causes x and y to be overwritten.