Lecture

Quick Introduction to the C Language

What do we need to code in C?

- A working computer :-)

- A C compiler
    - Windows: Microsoft Visual Studio 2017 Community Edition
    - Linux: gcc

- Text editor
    - gedit, emacs, vi, eclipse, etc.

Theres a lifetime of studies about C beyond this lecture!!

- Internet is your friend! (https://www.tutorialspoint.com/cprogramming/)

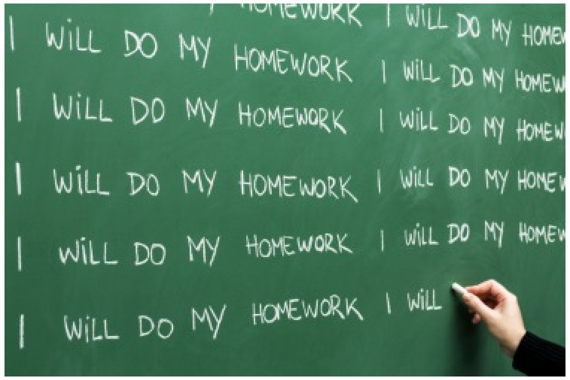
Our First Code in C

```c
//this says that we are using the input/output
//built-in functions
#include <stdio.h>

//this is line comment.

/*
   This is a multiple line
   comment
*/

//This is the main function, the one that will be run
int main(){
   printf("Hello World!\n");
}
```

Making decisions

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char** argv){
  if(argc <= 1){
   return -1;
  }
  int n = atoi(argv[1]);

  if(n > 10){
   printf("n is greater than 10\n");
   }
  else if(n == 10){
   printf("n is actually 10\n");
  }
  else{
   printf("n is lower and it's not 10\n");
  }

}
```

Making repetitions



```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char** argv){
  if(argc <= 1){
    return -1;
  }
  int n = atoi(argv[1]);

//Let's increase n until it reaches the value 10
  while(n < 10){
    printf("n = %d\n", n);
    n = n + 1;
  }

printf("n = %d\n", n);

//Let's increase n 10 times
  int i;
  for (i = 0; i < 10; i = i + 1){
    printf("n = %d. number of times increased = %d\n", n, i + 1);
    n = n + 1;
  }
}
```

Simple Variables/Pointers

- Variable: Contains a value
- Pointer: Contains a memory address (which may contain a value)

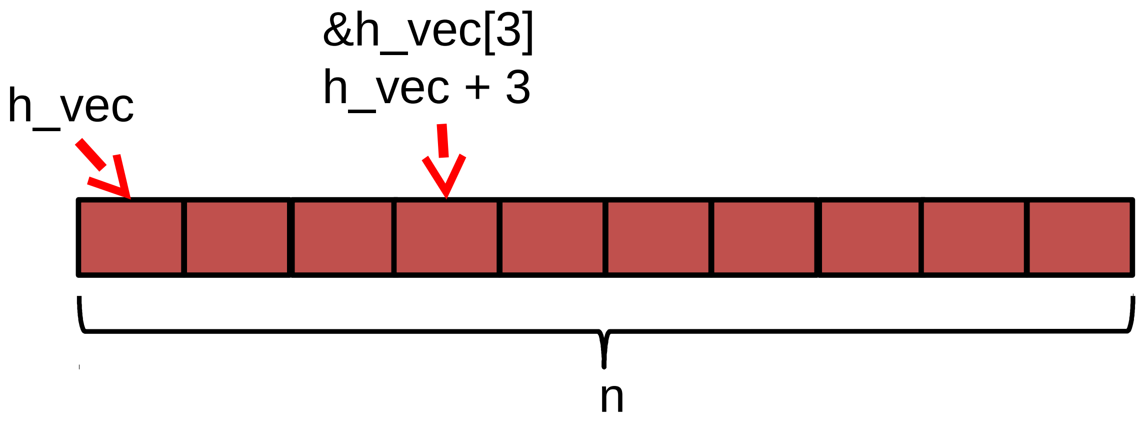
```
#include <stdio.h>

int main(){
  int ducks = 12;
  int *quacks = &ducks;
  printf("ducks=%d, quacks=%d\n", ducks, *quacks);
  ducks = 45;
  printf("ducks=%d, quacks=%p\n", ducks, quacks);
  *quacks = 67;
  printf("ducks=%p, quacks=%d\n", &ducks, *quacks);
}
```

Why pointers? Variable Passing! (copy/reference)

```
#include <stdio.h>

void sum(int number){
  number = number + 1;
}

void _sum(int* number){
  *number = *number + 1;
}

int main(){
  int n = 2;
  sum(n);
  printf("Is n equals to 3 now? n = %d\n", n);
  _sum(&n);
  printf("Is n equals to 3 now? n = %d\n", n);
}
```

Why pointers? Dynamic memory allocation!

```
#include <stdio.h>
#include <stdlib.h>

int main(){
  //n_max 10000000;
  int n = 20;

  printf("Allocating memory for vector s_vec.\n");
  int s_vec[n];
  printf("Done allocating memory!\n");

  printf("Dynamically allocating memory for vector h_vec.\n");
  int* h_vec = (int*) malloc(n * sizeof(int));
  printf("Done allocating memory!\n");

  int i;
  for(i = 0; i < n; i++){
      h_vec[i] = i / 4;
  }

  for(i = 0; i < n; i++){
    printf("%d ", h_vec[i]);
  }
  printf("\n");
  free(h_vec);
}
```



Practical Section - 1

1) Write the code snippet below
2) Replace “//PUT YOUR CODE HERE” with your code
3) Your code must decide if n is positive, negative, or zero

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char** argv){
  if(argc <= 1){
    return -1;
  }
  int n = atoi(argv[1]);
//YOUR CODE STARTS HERE

//PUT YOUR CODE HERE

//YOUR CODE ENDS HERE
  return 0;
}
```

Practical Section - 2

1) Write the code snippet below
2) Replace “//PUT YOUR CODE HERE” with your code
3) Your code must compute the factorial of n

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char** argv){
  if(argc <= 1){
    return -1;
  }
  int n = atoi(argv[1]);
//YOUR CODE STARTS HERE

//PUT YOUR CODE HERE

//YOUR CODE ENDS HERE
  return 0;
}
```

Practical Section - 3

1) Write the code snippet below
2) Replace “//PUT YOUR CODE HERE” with your code
3) Your code must compute the maximum and minimum values inside the vector `vec`

```
#include <stdio.h>
#include <stdlib.h>

int main(){
  int n = 50;
  int magnum = 32768;
  srand(magnum);
  int vec[n];

  int i;
  for(i = 0; i < n; i++){
     vec[i] = rand() % magnum;
  }
//YOUR CODE STARTS HERE
//PUT YOUR CODE HERE
//YOUR CODE ENDS HERE
  return 0;
}
```

Practical Section - 4

1) Write the code snippet below
2) Replace “//PUT YOUR CODE HERE” with your code
3) Your code must compute the sum of all values inside `vec`
4) Challenge: do it **without** any loop.

```
#include <stdio.h>
#include <stdlib.h>

int main(){
  int n = 5000;
  int i;
  int* vec = (int*) malloc(n * sizeof(int));

  for(i = 0; i < n; i++){
     vec[i] = i + 1;
  }
//YOUR CODE STARTS HERE
//PUT YOUR CODE HERE
//YOUR CODE ENDS HERE
  free(vec);
  return 0;
}
```