Lab 2

Pointers & Templates

Pointers

- The variables store the address of another variables.

Example 1 :

```
# include <stdio.h>
void fun(int x)
{
    x = 30;
}

int main()
{
  int y = 20;
  fun(y);
  printf("%d", y);
  return 0;
}
```

Pointers(Cont.)

```
# include <stdio.h>
void fun(int *ptr)
{
    *ptr = 30;
}
int main()
{
  int y = 20;
  fun(&y);
  printf("%d", y);
  return 0;
}
```

Pointers (Cont.)

```
#include <stdio.h>
int main()
{
    int arri[] = {1, 2 ,3};
    int *ptri = arri;
    char arrc[] = {1, 2 ,3};
    char *ptrc = arrc;
    printf("sizeof arri[] = %d ", sizeof(arri));
    printf("sizeof ptri = %d ", sizeof(ptri));
    printf("sizeof arrc[] = %d ", sizeof(arrc));
    printf("sizeof ptrc = %d ", sizeof(ptrc));
    return 0;
}
```

Difference between pointers & reference

- Pointers can be assigned as NULL whereas the reference can't.
- Pointers can be iterated over the array. Arithmetic operations like increment and decrement can be done over the pointers.
- Pointer is the variable which has own memory address which is difdifferent from the item address it stores. Reference has the same memory address to the item it reference to.

Use references when you can, and pointers where you have to

Templates

- A tool that allows passing the different data types to the function so that the user doesn't need to write the functionality again for different data types.
- A template is not a class or a function. A template is a “pattern” that the compiler uses to generate a family of classes or functions.
- Templates are much more powerful than generics. It's like macro which is not restricted to the data type only.