Programming Software Systems

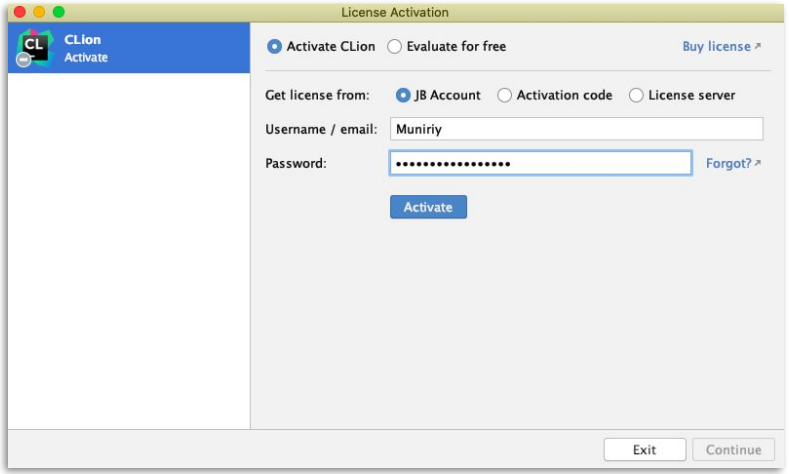
Lab 2

C is forever

Introducing **CLion**

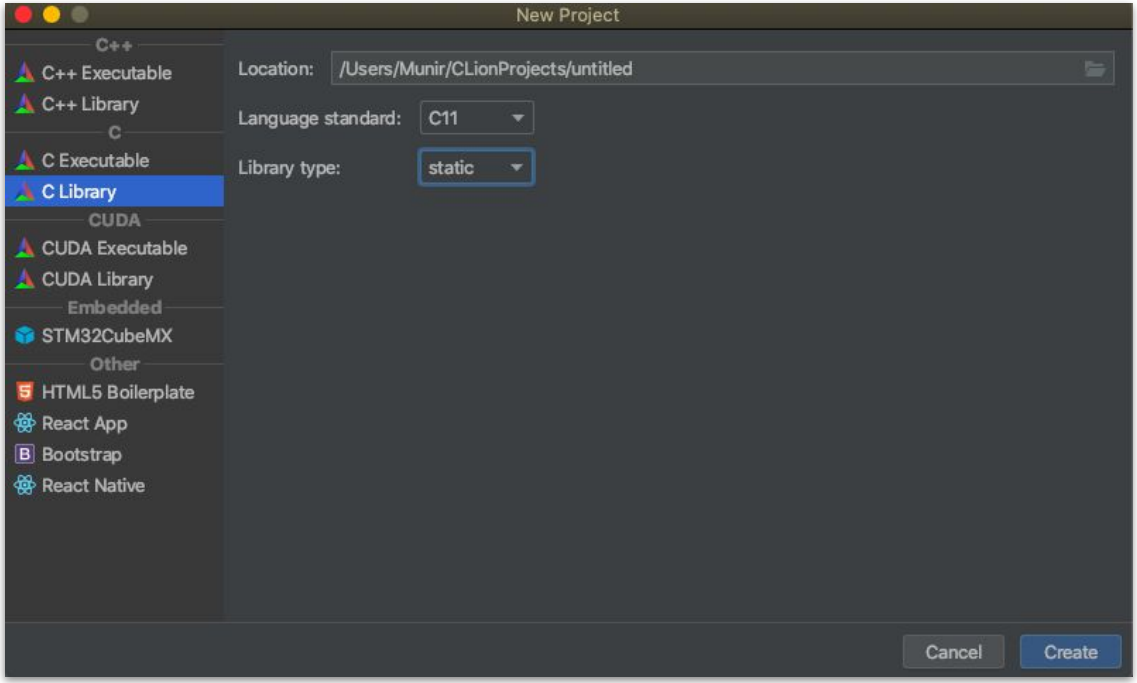
Download and Install the IDE

- CLion requires license to be used. As a student you can get free access to all JetBrains IDEs for personal use.
- Follow the next link to get the CLion IDE for free https://www.jetbrains.com/community/education/#students
- For application use your IU email, you should get a message
- Download and install CLion
- After installation to activate license choose JB account option and enter your username and password
- You may renew license if needed



New Project

For creation of a new project Choose C language and C11 standard



Hello World

Let’s start and run Hello World program from the previous lab

Pointes, **strings** and **arrays**

Pointers. Address of a variable

```c
#include <stdio.h>

int main () {

   int  var1;
   char var2[10];

   printf("Address of var1 variable: %x\n", &var1  );
   printf("Address of var2 variable: %x\n", &var2  );

   return 0;
}
```

```
Address of var1 variable: bff5a400
Address of var2 variable: bff5a3f6
```

Pointers. Address of a variable

```c
#include <stdio.h>

int main () {

   int  var1;
   char var2[10];

   printf("Address of var1 variable: %x\n", &var1  );
   printf("Address of var2 variable: %x\n", &var2  );

   return 0;
}
```

```
Address of var1 variable: bff5a400
Address of var2 variable: bff5a3f6
```

Pointers

A **pointer** is a variable whose value is the address of another variable

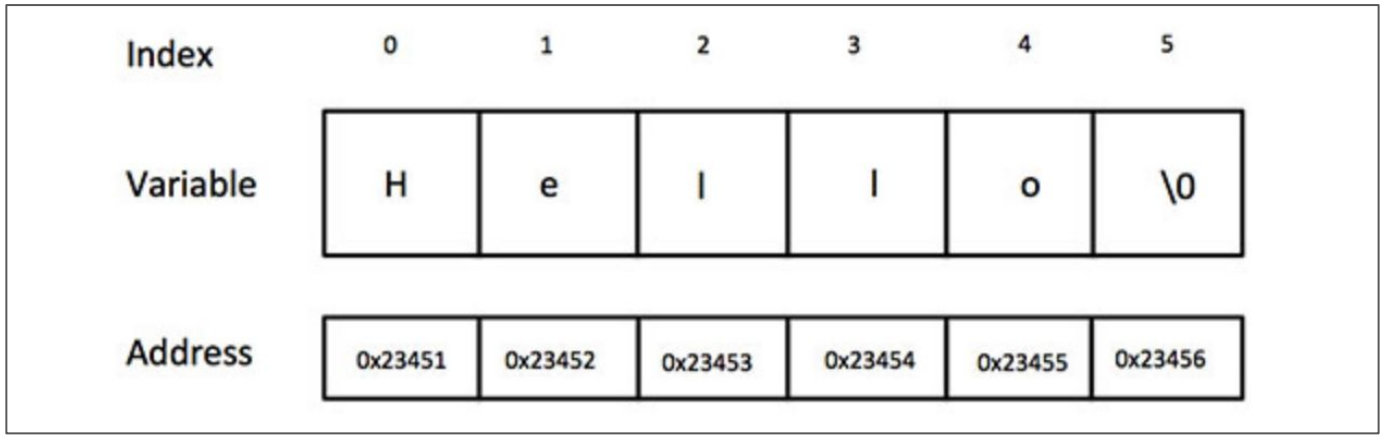
```c
#include <stdio.h>

const int MAX = 3;

int main () {

   int  var[] = {10, 100, 200};
   int  i, *ptr;

   /* let us have array address in pointer */
   ptr = &var[MAX-1];

   for ( i = MAX; i > 0; i--) {

      printf("Address of var[%d] = %x\n", i-1, ptr );
      printf("Value of var[%d] = %d\n", i-1, *ptr );

      /* move to the previous location */
      ptr--;
   }

   return 0;
}
```

```
Address of var[2] = bfedbcd8
Value of var[2] = 200
Address of var[1] = bfedbcd4
Value of var[1] = 100
Address of var[0] = bfedbcd0
Value of var[0] = 10
```

Strings

Strings are actually one-dimensional array of characters terminated by a **null** character '\0'.

```
1 char greeting1[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
2 char greeting2[] = "Hello";
```



Strings

Strings are actually one-dimensional array of characters terminated by a **null** character '\0'.

```
1 char greeting1[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
2 char greeting2[] = "Hello";
```

Do we really need to insert the **null** character **'\0'** at the end of the char array ???

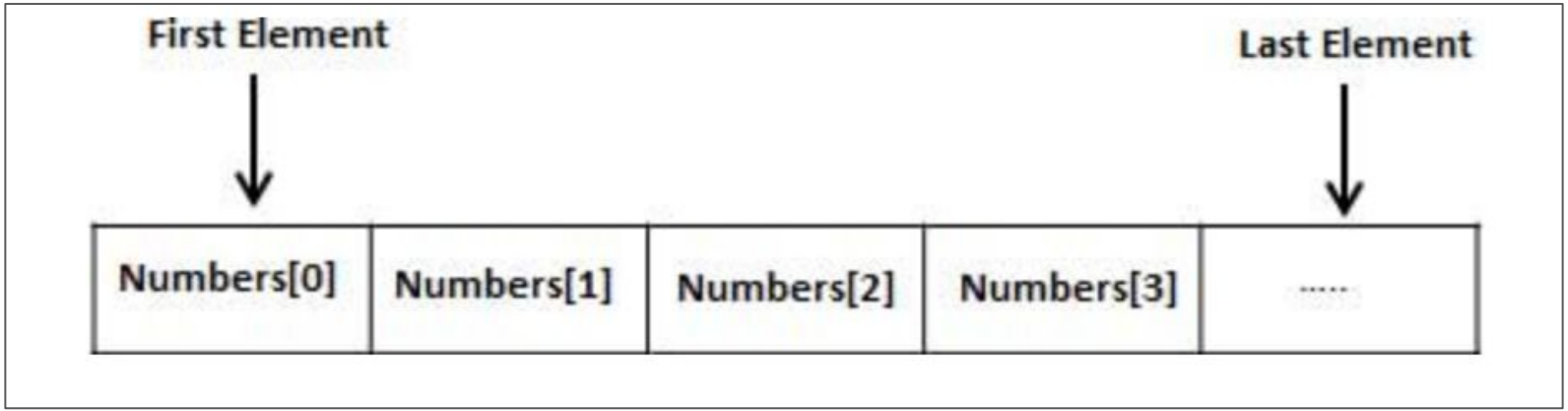
Strings

```c
#include <stdio.h>

int main () {

   char greeting[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
   printf("Greeting message: %s\n", greeting );
   return 0;
}
```

```
Greeting message: Hello
```

Arrays

Arrays a kind of data structure that can store a fixed-size sequential collection of elements of the same type.



Arrays. Declaration. Initialization.

```
1 type arrayName [ arraySize ];
2
3 double balance[10];
4
5 double balance[5] = {1000.0, 2.0, 3.4, 7.0, 50.0};
6
7 double balance[] = {1000.0, 2.0, 3.4, 7.0, 50.0};
8
9 balance[4] = 50.0;
```

Arrays. Passing Arrays as Function Arguments.

```
 1 double getAverage(int arr[], int size) {
 2
 3    int i;
 4    double avg;
 5    double sum = 0;
 6
 7    for (i = 0; i < size; ++i) {
 8       sum += arr[i];
 9    }
10
11    avg = sum / size;
12
13    return avg;
14 }
```

```
 1 #include <stdio.h>
 2
 3 /* function declaration */
 4 double getAverage(int arr[], int size);
 5
 6 int main () {
 7    /* an int array with 5 elements */
 8    int balance[5] = {1000, 2, 3, 17, 50};
 9    double avg;
10
11    /* pass pointer to the array as an argument */
12    avg = getAverage( balance, 5 ) ;
13
14    /* output the returned value */
15    printf( "Average value is: %f ", avg );
16
17    return 0;
18 }
```

Arrays. Pointer to an Array.

```c
#include <stdio.h>

int main () {
   /* an array with 5 elements */
   double balance[5] = {1000.0, 2.0, 3.4, 17.0, 50.0};
   double *p;
   int i;

   p = balance;

   /* output each array element's value */
   printf( "Array values using pointer\n");

   for ( i = 0; i < 5; i++ ) {
      printf("*(p + %d) : %f\n",  i, *(p + i) );
   }

   printf( "Array values using balance as address\n");

   for ( i = 0; i < 5; i++ ) {
      printf("*(balance + %d) : %f\n",  i, *(balance + i) );
   }

   return 0;
}
```

Exercises (1)

1) Write a program that prompts the user for a string, and prints its reverse
2) Write a function that outputs a right-side-up triangle of height n and width 2n-1. Your program must accept n as a command line parameter; the output for n = 6 would be:

```
     *
    ***
   *****
  *******
 *********
***********
```

Exercises (2)

1. Add several functions to your previous solution, so user could print different figures on his/her choice; examples are:

2. Write a program that asks user to input two integers and swaps them using a separate function
    - Hint: you will need to pass parameters by reference

Exercises (3)

1. Write an implementation of **linked_list**
2. Extend your **linked_list** to a **double_linked_list**
3. Write a program that asks user to input a text and writes the input into a text file
    - Hint: you will need to use `fopen, fgets, fputs`

Homework: (to be submitted to)

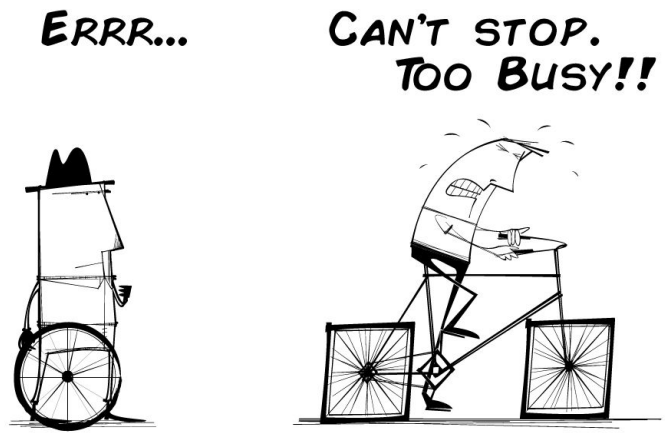
Write a program that reads the data from a text file and writes it into a text file

- Hint: you will need to use `fopen, fgetc and putchar`

PSP

How to Improve the Quality of your work?

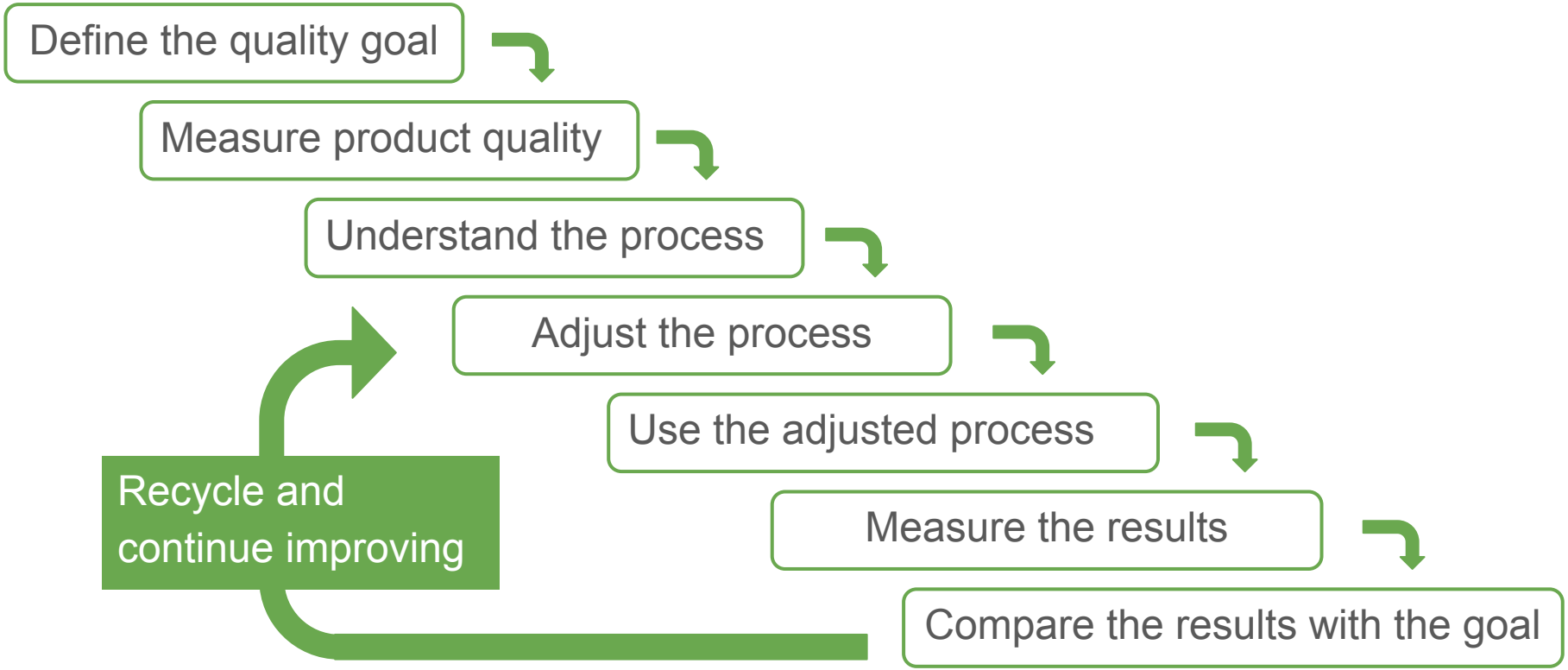
1) Collect data
2) Analyze
3) Improve



TOO BUSY TO IMPROVE?

WorkCompass

The Improvement Process



References

1. https://www.tutorialspoint.com/cprogramming/index.htm
2. http://hilite.me/