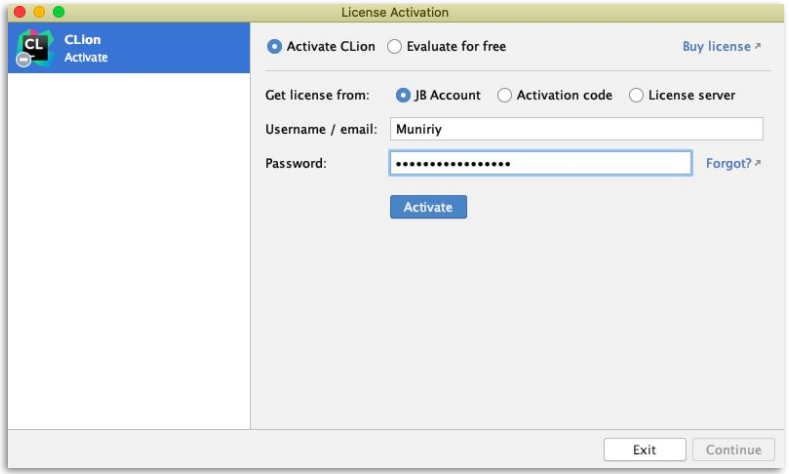
Introduction to Programming I

Lab 2

Introducing **CLion**

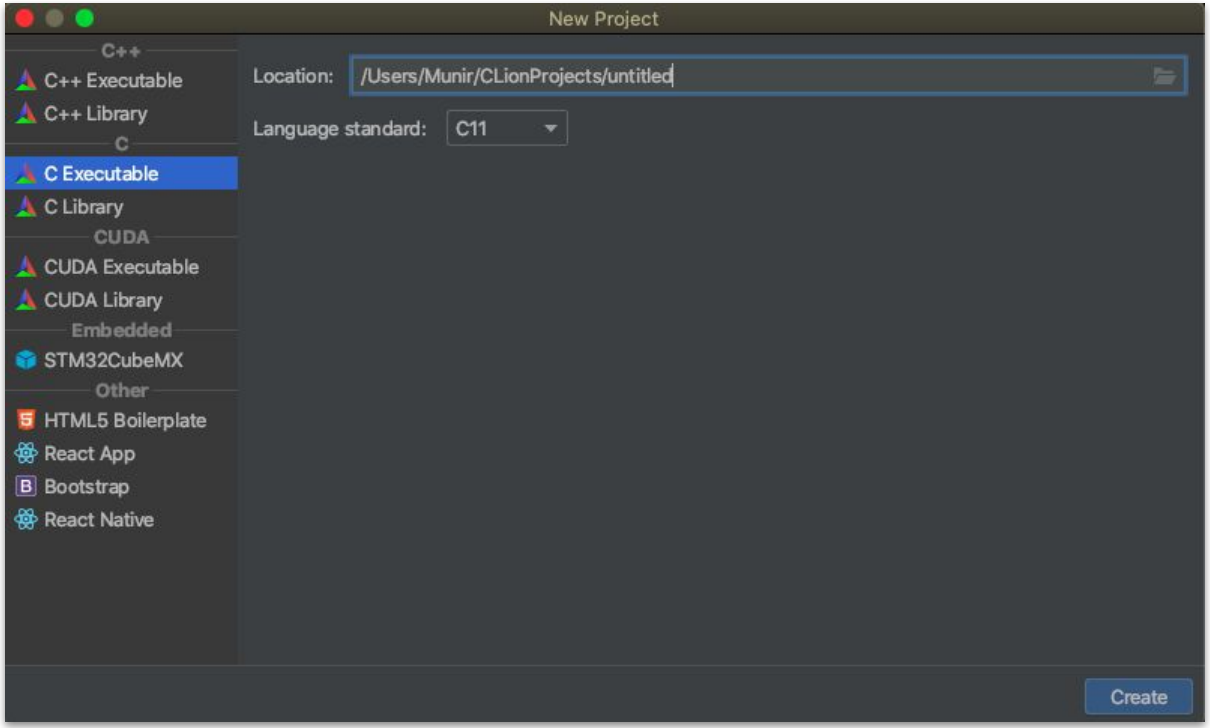
Download and Install the IDE

- CLion requires license to be used. As a student you can get a 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

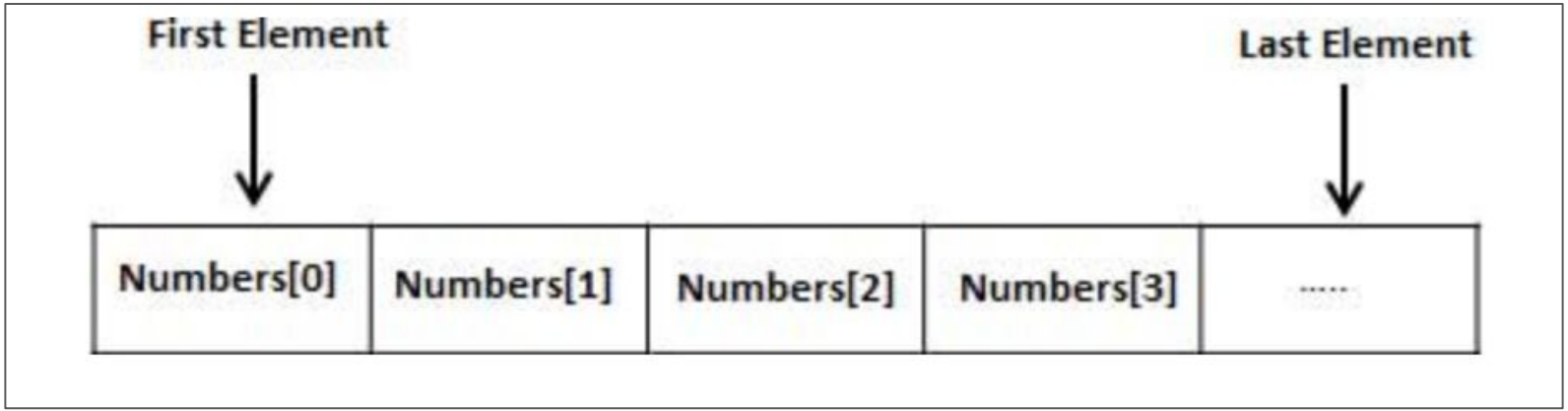
Let’s start and run Hello World program from the previous lab using the IDE

Pointers, **strings** and **arrays**

DATA TYPE	MEMORY (BYTES)	RANGE	I/O FORMAT SPECIFIER
short int	2	-32,768 to 32,767	%hd
unsigned short int	2	0 to 65,535	%hu
unsigned int	4	0 to 4,294,967,295	%u
int	4	-2,147,483,648 to 2,147,483,647	%d
long int	4	-2,147,483,648 to 2,147,483,647	%ld
unsigned long int	4	0 to 4,294,967,295	%lu
long long int	8	-(2^63) to (2^63)-1	%lld
unsigned long long int	8	0 to 18,446,744,073,709,551,615	%llu
signed char	1	-128 to 127	%c
unsigned char	1	0 to 255	%c
float	4	3.4e-38 to 3.4e+38	%f
double	8	1.7e-308 to 1.7e+308	%lf
long double	16	3.4e-4932 to 1.1e+4932	%Lf

Arrays

Arrays are 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.

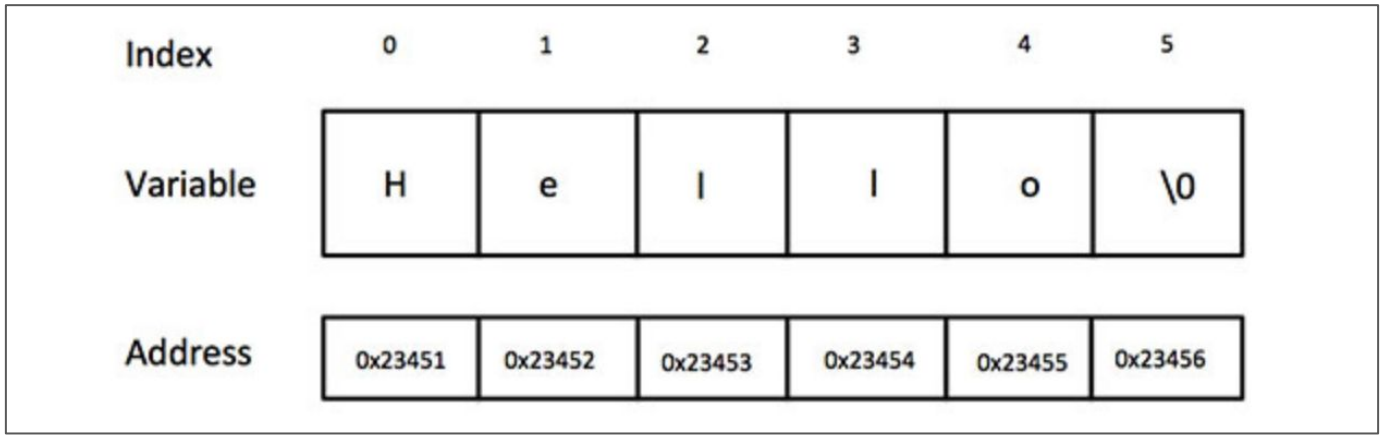
```
1 double getAverage(int arr[], int size) {
2
3    int i;
4    double avg;
5    double sum = 0;
6
7    for (i = 0; i < size; ++i) {
8       sum += arr[i];
9    }
10
11   avg = sum / size;
12
13   return avg;
14 }
```

```
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 }
```

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
```

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
```

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 function that outputs a isosceles 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, 3:

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

4. Write a program that asks user to input two integers and swaps them using a separate function *(you may need to pass parameters by reference)*

Exercises 4:

1. Write a program that prompts the user for a string, reverse it using recursion and pointers and prints the result.

Exercice4: Solution

1. Write a program that prompts the user for a string, reverse it using recursion and pointers and prints the result.

```c
#include <stdio.h>
#include <string.h>
void reverse_string(char*, int, int);

int main()
{
    //This array would hold the string upto 150 char
    char string_array[150];
    printf("Enter any string:");
    scanf("%s", &string_array);

    //Calling our user defined function
    reverse_string(string_array, 0, strlen(string_array)-1);
    printf("\nReversed String is: %s",string_array);

    return 0;
}

void reverse_string(char *x, int start, int end)
{
    char ch;
    if (start >= end)
       return;

    ch = *(x+start);
    *(x+start) = *(x+end);
    *(x+end) = ch;

    //Function calling itself: Recursion
    reverse_string(x, ++start, --end);
}
```

References

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