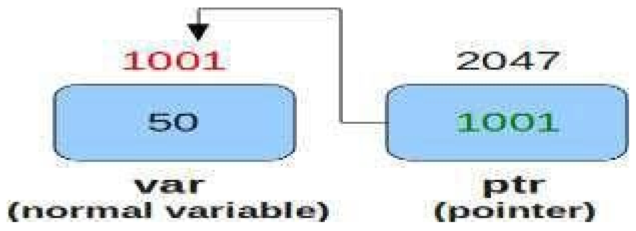
# CSE 115L: Programming Language I Lab (Section: 06)

Spring 2020

Lab-08 (2D Arrays)

---

**Pointers:** C Pointer is a variable that stores/points the address of another variable. C Pointer is used to allocate memory dynamically i.e. at run time. The pointer variable might be belonging to any of the data types such as int, float, char, double, short etc.



Example 1:

```
#include <stdio.h>

int main()
{
        int c;
        int* pc;
        c = 22;
        printf("Address of c: %d\n",&c);
        printf("Value of c:%d\n\n",c);

        pc = &c;
        printf("Content of pointer pc:
%d\n", pc);
        printf("content of the address %d:
%d\n\n", pc, *pc);

        c = 11;
        printf("Content of pointer pc:
%d\n", pc);
        printf("content of the address %d:
%d\n\n", pc, *pc);

        *pc = 2;
        printf("Content of pointer pc:
%d\n", pc);
        printf("content of the address %d:
%d\n\n", pc, *pc);
        return 0;}
```

Example 2: pointer to pointer

```
#include <stdio.h>

int main ()
{
        int var;
        int *ptr, **pptr;
        var = 3000;
        /* take the address of var */
        ptr = &var;
        /* take the address of ptr using
address of operator & */
        pptr = &ptr;
        /* take the value using pptr */
        printf("Value of var = %d\n", var );
        printf("Value available at *ptr =
%d\n", *ptr );
        printf("Value available at **pptr =
%d\n", **pptr);

        return 0;
}
```

**Array and pointer:** The name of an array holds the memory address of the first element of that array.

```
int Array[3];
int *ptr = &Array;
Array[0] = 2;
Array[1] = 7;
Array[2] = 5;
```

| Pointer | ptr | ptr+1 | ptr+2 |
|---|---|---|---|
| Memory address | 1000 | 1004 | 1008 |
| Element | Array[0] | Array[1] | Array[2] |
| Value | 2 | 7 | 5 |

**Each integer variable takes 4 bytes. So if an integer is stored at location n, the next one will be stored at n+4.

i) passing array to function using pointer

```
#include<stdio.h>

void printNum(int *ptr, int len);

int main()
{
        int a[4]={4,10,1,5};
        printNum(a,4);
        return 0;
}

void printNum(int *ptr,int len)
{
        int i;
        for(i=0; i <len ; i++ )
        {
        printf("*(ptr+%d) = %d
\n",i,*(ptr+i),i));
        printf("ptr[%d]= %d \n",i, ptr[i]);
        }
}
```

ii) returning pointer from functions

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

int* getRandom();

int main ()
{
        int *p;
        p = getRandom();
        printf("address of %d\n",p);
        printf("value at *p : %d\n", *p );
        return 0;
}

int* getRandom( )
{
        static int j;
        j=rand()%10;
        printf("value of j: %d\n",j);
        printf("Address of j: %d\n", &j);
        return &j;
}
```

Key points to remember about pointers in C

1. Normal variable stores the value whereas pointer variable stores the address of the variable.
2. The content of the C pointer always be a whole number i.e. address.
3. Always C pointer is initialized to null, i.e. **int *p = NULL**. The value of the null pointer is 0.
4. & symbol is used to get the address of the variable.
5. * symbol is used to get the value of the variable that the pointer is pointing to.
6. If NULL is assigned to a pointer, it means it is pointing to nothing.

| **Some Common Mistakes** | **Equivalent Expression** |
|---|---|
| int c, *pc=c;<br>//pc is address whereas, c is not an address.<br>*pc=&c;<br>//&c is address whereas, *pc is not an address. | list[2]=5;<br>*(p+2)=5;<br>p[2]=5;<br>*(list+2)=5;<br>All of the above are the same. |

**Task:**

1. Write a function which will display the sum of an array using pointers.