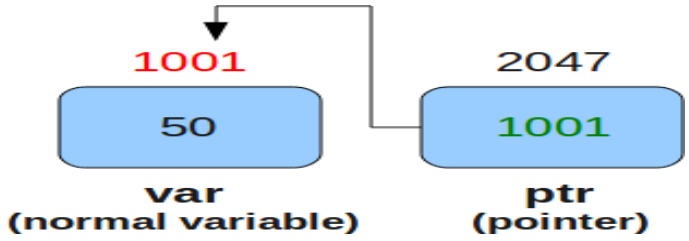
CSE115L – Programming Language I Lab

Lab - 27

Pointers

In this lab, we will learn how to use pointers. A pointer is a variable that stores the address of another variable. By storing the address of a variable, a pointer points to that variable. The pointer variable can be of any data type, such as int, float, char or double, depending on the type of the variable it stores the address of.



In C programming, the name of an array is actually a pointer that always points to address of the first element of the array. For instance, consider the following program snippet.

```
int a[5] = {1,2,3,4,5};
int *p = a;
```

Elements	a[0]	a[1]	a[2]	a[3]	a[4]
Value/content	1	2	3	4	5
Pointer	p	p+1	p+2	p+3	p+4
Memory address	1000	1004	1008	1012	1016

Key points to remember:

- Pointer variables store the memory addresses of other variables.
- An address is always an integer number.
- Always initialize pointer with NULL, i.e. `int *p = NULL`. NULL means 0.
- If NULL is assigned to a pointer, it means it does not points to anything.
- The **&** symbol is used to get the address of variables.
- The ***** symbol is used to get the value at the address a pointer points to.
- Integer arithmetic operations can be performed on pointer variables.

Example 1: Declaration and initialization of pointers

```
#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("Address of c:%d\n",pc);
    printf("Value of c:%d\n",*pc);
    c=11;
    printf("Address of c:%d\n",pc);
    printf("Value of c:%d\n",*pc);
    *pc=2;
    printf("Address of c:%d\n",&c);
    printf("Value of c:%d\n\n",c);
    return 0;
}
```

Example 2: Passing pointers to functions

```
#include<stdio.h>

void multiplyByTwo(int *n)
{
    *n = *n * 2;
}

int main()
{
    int n = 10;
    multiplyByTwo(&n);
    printf("n = %d",
    n); return 0;
}
```

Example 3: Passing pointers to functions

```
#include <stdio.h>

void swap(int *p, int *q);

int main ()
{
    int a=2, b=3;
    swap(&a,&b);
    printf("a= %d b= %d",a,b);
    return 0;
}

void swap(int *p, int *q)
{
    int temp=*p;
    *p=*q;
    *q=temp;
}
```

```
#include <stdio.h>

void print(char *s);

int main ()
{
    char str[]="simple";
    print(str);
    return 0;
}

void print(char *s)
{
    while(*s!='\0')
    {
        printf("%c",*s); s++;
    }
}
```

Example 4: Manipulating address and values with pointer

```
#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("Address of c:%d\n",pc);
    printf("Value of c:%d\n\n",*pc);

    c=11;
    printf("Address of c:%d\n",pc);
    printf("Value of c:%d\n\n",*pc);

    *pc=2;
    printf("Address of c:%d\n",&c);
    printf("Value of c:%d\n\n",c);

    return 0;
}
```

Example 5: Pointer to pointer

```
#include   <stdio.h>

int main () {

    int var;
    int *ptr;
    int **pptr;
    var = 3000;

    ptr = &var;

    pptr = &ptr;

    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;
}
```

Example 5: Arrays and pointers

```
#include <stdio.h>
int main()
{
    int a[5] = {1,2,3,4,5};
    int *p = a;
    printf("a[0] at address%d\n",&a[0]);
    printf("p = %d\n",p);
    printf("p[2] = %d\n", p[2]);
    *(p+2) = 9;
    printf("p[2] = %d\n", p[2]);
    p[2] = 7;
    printf("p[2] = %d\n", p[2]);
    return 0;
}
```

Example 6: Passing arrays and pointers

```
#include<stdio.h>

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

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

Perform the following tasks.

Task 1: Implement the following function which takes the radius of a circle as one of its parameters and stores the circumference and area using the other two parameters.

void calcCircleInfo(int radius, int *circumference, int *area)

Task 2: Implement the following function which accepts a string as parameter and reverses it, without using any function from the string library.

void strReverse(char *str)