STORAGE CLASSES

BY : Dr. Shefali Arora

Assistant Professor, CSE

NIT Jalandhar

Storage classes

- There are four storage classes in C programming language, which are as follows −
- auto
- extern
- static
- register

Global/Local/Extern variables

```c
#include <stdio.h>
#include "external.h"
enum e{rock,paper,scissor};
int count=1;
int main()
{
    int choice_1,choice_2;
    printf("Player 1:Enter choice 0 for rock, 1 for player, 2 for scissor");
    scanf("%d",&choice_1);
    printf("Player 2:Enter choice 0 for rock, 1 for player, 2 for scissor");
    scanf("%d",&choice_2);
    printf("Player 1 entered %d",choice_1);
    printf("Player 2 entered %d",choice_2);
    printf("Game Count= %d\n",count);
    printf("%d value of a",a);

}
```

Add extern int a=14
In external.h file

Or int a=5 in file new.c
And extern int a in present file

Auto vs Static

- The all local variables which are defined within the function are known as `auto` (automatic) variables unless not specified i.e. by default a local variable is an auto variable. There is no need to put the keyword `auto` (it's an optional) while declaring a local variable.
- Static keyword should be used to declare a variable static. It initializes a variable to 0 and it remains in the memory as long as the program is running.

Precedence

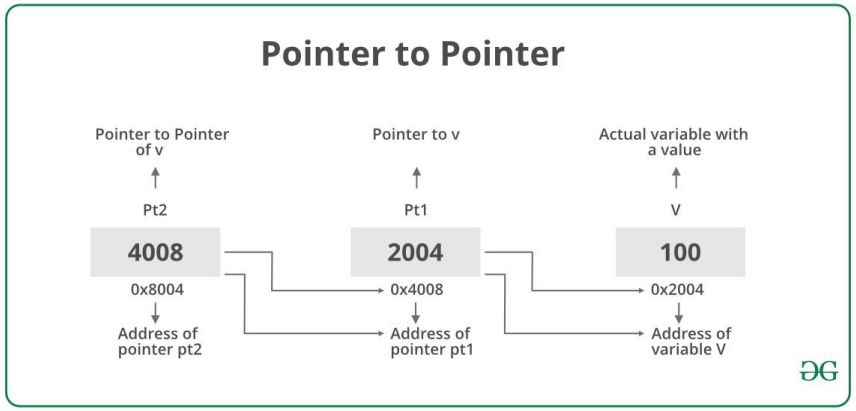
Category	Operator	Associativity
Postfix	() [] -> . ++ - -	Left to right
Unary	+ - ! ~ ++ - - (type)* & sizeof	Right to left
Multiplicative	* / %	Left to right
Additive	+ -	Left to right
Shift	<< >>	Left to right
Relational	< <= > >=	Left to right
Equality	== !=	Left to right
Bitwise AND	&	Left to right
Bitwise XOR	^	Left to right
Bitwise OR	|	Left to right
Logical AND	&&	Left to right

Pointer to Pointer

```c
#include <stdio.h>

int main()
{
    int a=10;
    int *p=&a;
    int **q=&p;
    printf("%d %d %d %d %d",p,*p,*q,**q, q);
}
```

Pointer to a pointer is **a form of multiple indirection or a chain of pointers**. Normally, a pointer contains the address of a variable. When we define a pointer to a pointer, the first pointer contains the address of the second pointer, which points to the location that contains the actual value as shown below.



ARRAYS

ARRAYS

- Collection of elements of same data type
- int a[5]={1,2,3,4,5}
- Marks of 50 students in Mathematics:

Declare array as int marks[50];

Instead of using separate variables

- Indexing begins from 0 onwards
- a[1] -> 1, a[2]->2 and so on
- Considering 4 bytes for integer, marks[50] reserves

50*4 bytes for the array elements.

INPUT MARKS & PRINT SUM

```c
#include <stdio.h>
int main()
{
    int marks[5],sum=0;
    for(int i=0;i<5;i++)
    {
    printf("Enter for student %d",i+1);
       scanf("%d",&marks[i]);
    }
     for(int i=0;i<5;i++)
    {
    printf(" for student %d marks are %d \n",i+1,marks[i]);
    }

     for(int i=0;i<5;i++)
    {
    sum+=marks[i];
    }
    printf("Sum is %d",sum);

    return 0;
}
```

Arrays & Pointers

```c
#include <stdio.h>

int main()
{
    int a[5]={1,2,3,4,5};
    printf("%d %d\n",a,*a);
    int *p=&a;
    printf("%d %d\n",p,*p);
    p++;
    printf("%d %d",p,*p);
}
```

```c
#include <stdio.h>

int main()
{
    int a[5]={10,20,30,40,50};
    int *p=&a[0];
    printf("%d \n",a[0]);
    printf("%d \n",*p);
    printf("%d \n",*p++);
    printf("%d \n",++*p);
    printf("%d \n",*++p);
    printf("%d \n",++(*p));
}
```

10 10 10 21 30 31

Arrays & pointers

```c
#include <stdio.h>

int main()
{
   int a[5]={10,20,30,40,50};
   int *p=a;
   for(int i=0;i<5;i++)
   {
       printf("\n%d",p[i]);
   }
   for(int i=0;i<5;i++)
   {
       printf("\n%d",*(p+i));
   }

    return 0;
}
```

```c
#include <stdio.h>

int main()
{
   int a[5]={10,20,30,40,50};
   int *p=a;
   for(int i=0;i<5;i++)
   {
       printf("\n%d",*p);
       p++;
   }

    return 0;
}
```

Pointer to Array vs Array of Pointers

```
**********************************************************************/
#include <stdio.h>
#include<stdio.h>

int main()
{
    // Pointer to an integer
    int *p;
    int (*ptr)[5];
    int arr[5]={8,9,7,6,5};
    p = arr;

    // Points to the whole array arr.
    ptr = &arr;
    for(int i=0;i<5;i++)
    {
        printf("%d\n",*(p+i));
    }

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

    p++;
    ptr++;

    printf("p = %d, ptr = %d\n", p, ptr);

    return 0;
}
```

```
#include <stdio.h>

const int MAX = 3;

int main () {

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

   for ( i = 0; i < MAX; i++) {
      ptr[i] = &var[i]; /* assign the address of integer. */
   }

   for ( i = 0; i < MAX; i++) {
      printf("Value of var[%d] = %d\n", i, *ptr[i] );
   }

   return 0;
}
```

2D Arrays

```
 8  ************************************************************
 9  #include <stdio.h>
10  #include<stdio.h>
11
12  int main()
13  {
14      // Pointer to an integer
15     int a[2][3]={{1,2,3},{4,5,6}};
16     for(int i=0;i<2;i++)
17     {
18         for(int j=0;j<3;j++)
19         {
20             printf("%d",a[i][j]);
21         }
22         printf("\n");
23     }
24
25     return 0;
26  }
27
```

input

```
123
456
```

- Consists of rows and columns
- Storage representation in matrix/tabular form
- Representation in memory: Collection of 1D arrays

Functions