Data types, Type Qualifiers, storage class specifiers and memory layout

Anup Kumar

Modifiers

- It is a keyword in c.
- It specifies the amount of memory space to be allocated for a variable.
- Modifiers are prefixed with basic data types to modify the allocated for a varible.

There are five data type modifiers in c programming language.

- long
- short
- signed
- unsigned
- long long

Basic data type.

Data type determines the type of data a variable will hold.

- char : 1 bytes
- Int : 4 Bytes
- Float : 4 Bytes
- Double : 8 Bytes

Range of n-bit number

- Signed : -2^(n-1) to (n-1) – 1

  eg: for 3 bits -2^(3-1) to 2^(3-1) -1

- Unsigned :0 to 2^(n) – 1

  eg : for 3 bits 0 to 2^(3) -1

Type qualifier

- It is used to indicate special properties of the object being declared.

Type-qualifier :

- const.
- volatile.

Type qualifier may appear with any type specifier.

Type qualifiers

const:

- can be applied to char, int, float, double
- initialized value cannot be modified
- .rodata block
- global const variables
- const applied to local/auto is stored in stack.

Volatile :

- not to optimize code

Type casting

A way to convert a variable from one data type to another data type.

-convert lower data type to higher data type to avoid data loss.

-Data will be truncated when higher data type is converted to lower.

```
#include <stdio.h>

int main ()

{
  int x;

  x = (float)13/7;

  printf("%f", x);

}
```

Storage Class specifiers

1)auto 2)register

3)extern 4)static

Syntax :

storage_class_specifier data_type variable_name;

Storage Class specifiers contd...

1. Automatic Storage Class

```
#include <stdio.h>

int main( )

{

  auto int num_1 = 10;

  {

    auto int num_2= 20;

    {

      auto int num_3 = 30;
```

```
printf ( "%d\t ", num_1);

  }

  printf ( "%d\t ", num_2);

 }

 printf( "%d\t", num_3);

}
```

OUTPUT:

```
10 20 30
```

Storage Class specifiers contd...

2. Register Storage Class

```
#include <stdio.h>

int main()

{
 register int num = 20;

   int *ptr;

   ptr = &num;

   printf("address of num : %u", ptr);

   return 0;

}
```

OUTPUT : ?

Storage class specifiers contd….

file1.c

```
#include<stdio.h>
int var = 10;//global variable
void fun()
{
printf(“%d”, var);
}
```

File2.c

```
#include”file1.c”
void main(void)
{
extern int var;
fun();
}
```

Storage Class specifiers contd…

```
#include <stdio.h>

 extern int x;

  int main()

{

  printf("x:
  %d\n", x);

}

 int x = 10;
```

```
OUTPUT : 10
```

4.static storage class specifiers

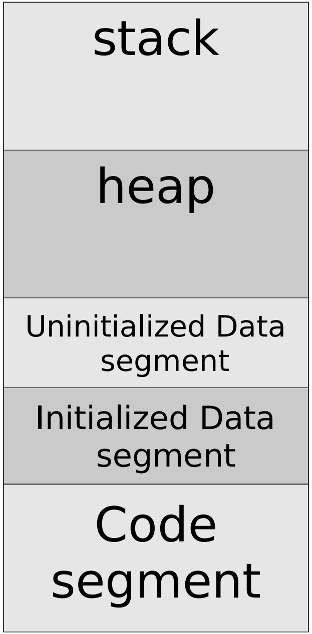
```
#include<stdio.h>

void main(void)

{

 staticfun();

staticfun();

}
```

```
Void staticfun()

{

        static int num;

        {

        static int num = 1;

                printf(“%d”, num);

                Num ++;}

                Printf(“%d\n”, num);

                Num++;}

                Output :

                1 0 2 1
```

Storage Class specifiers contd...

Storage Class Specifier	**Defined/Declared**	**Storage Space**	**Scope**	**Lifetime**
auto	Within a Block/Function	Stack	Block/Function	Within a Block/function
register	Within a Block/Function	Processor Registers	Block/Function	Within a Block/function
static	Within a Block/Function (Function/Block static)	Data Section	Block/Function	Throughout the Program
	Outside Function (File static)	Data Section	All Functions with in a same C file	Throughout the Program
extern	Outside Function	No Storage	All Functions with in a same C file	Throughout the Program

Memory Layout


stack

heap

Uninitialized Data segment

Initialized Data segment

Code segment

```
#include <stdio.h>

char str[ ] = “Global Edge"; /* global variable stored in Initialized Data
                        Segment in read-write area*/

const char str[ ] = “Global"; /* global variable stored in Initialized Data
                        Segment in read-only area*/


char ch; /* Uninitialized variable stored in bss*/  int main()
{
            static  int  num=  2 0 ;  /* static variable stored in Initialized Data
                    Segment*/
    static int num;                          /* Uninitialized static variable stored
            in bss */

    char *ptr= (char*)malloc(sizeof(char)); /* memory allocating in heap
                                                  segment */
    return 0;
}
```

Large enough to Deliver, Small enough to Care

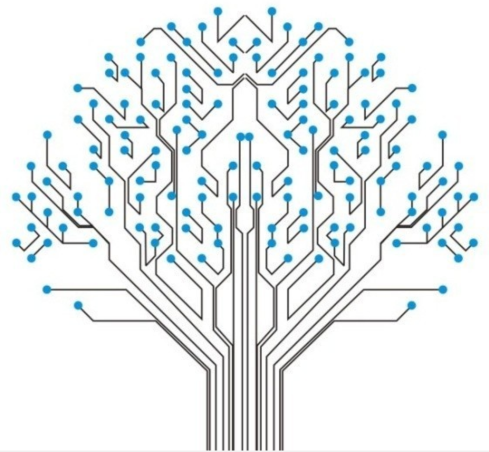
Global Village
IT SEZ
Bangalore

South Main Street
Milpitas
California

Raheja Mindspace
IT Park
Hyderabad

www.globaledgesoft.com

Thank you



Fairness

Learning

Responsibility

Innovation

Respect