CLASS 7 - TUTORIAL

Prof. Pallab Dasgupta

Dept. of CSE

POINTER ARITHMETIC

Following operations can be done on a pointer variable.

- **Pointer Assignment:**
  - ptr2 = ptr1;
- **Incrementing a Pointer or Pointer Addition:**
  - ptr1 = ptr1 + 2;
  - ptr1++;

```
Ex:
    int * ptr1;            //1024
    int * ptr2 = ptr1+n;    //ptr2 = 1024 + (n * size_of_datatype_of the_pointer)
    So if,
    int * ptr3 = ptr1 + 1;  //ptr3 = 1024 + 4 = 1028
```

- **Decrementing a Pointer or Pointer Subtraction:**
  - ptr1 = ptr1 – 2;
  - ptr1--;
- **Pointer Comparison:**
  - if(ptr1 == ptr2)
  - if(ptr1 < ptr2)
  - if(ptr1 <= ptr2)
  - if(ptr1 > ptr2)
  - if(ptr1 >= ptr2)

POINTER ARITHMETIC COMMON MISTAKES

Since, pointer stores address hence we must ignore the operations which may lead to an illegal address. Such as the following operations can not be performed on pointers.

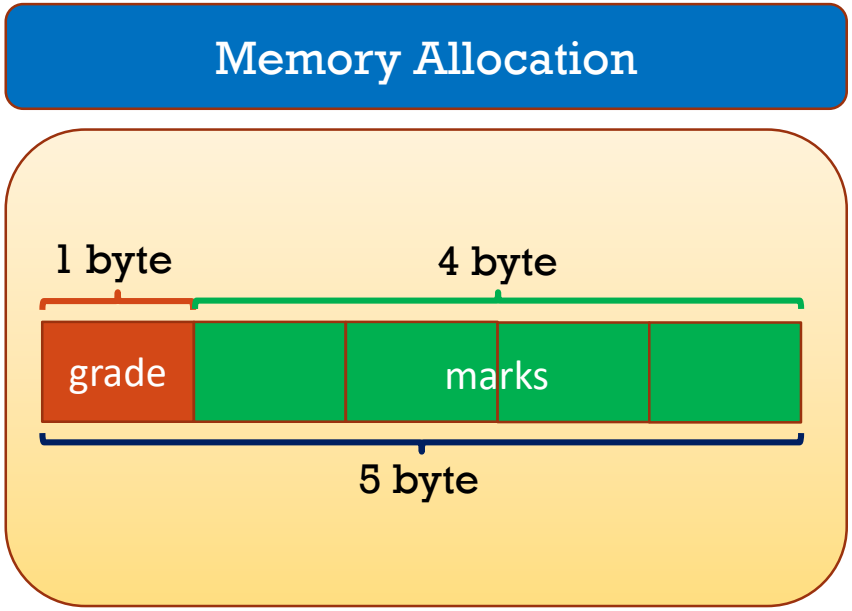
- **Addition:** Address + Address
- **Multiplication:** Address * Address
- **Modulus:** Address % Address
- **Division:** Address / Address
- **Bitwise AND:** Address & Address
- **Bitwise XOR:** Address ^ Address
- **Bitwise OR:** Address | Address
- **Bitwise NOT:** ~Address

STRUCTURE

A structure is a user defined data type in C. A structure creates a data type that can be used to group items of possibly different types into a single type.

Ex:

```
struct score
{
    char grade;    // 1 byte
    int marks;     // 4 byte
};
```

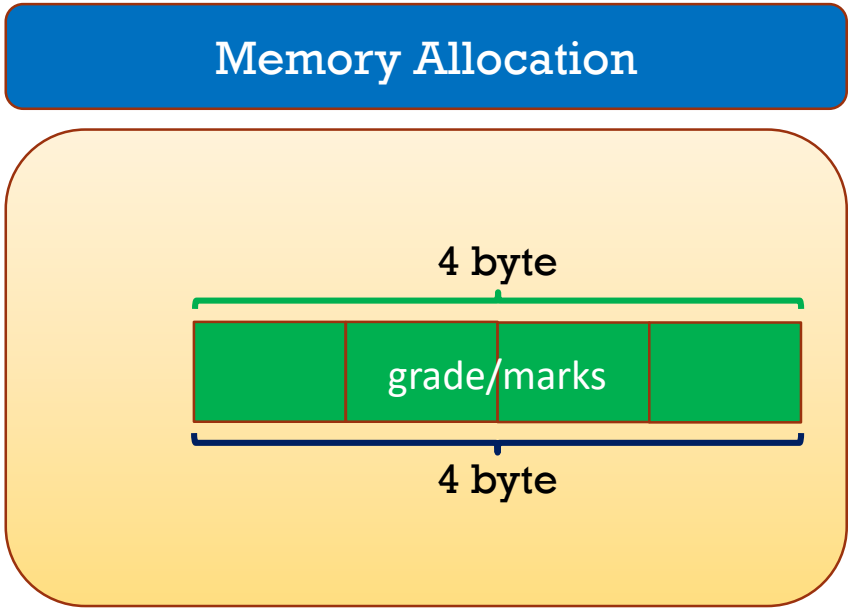


UNION

Like Structures, union is a user defined data type. In union, all members share the same memory location.

Ex:

```
union score
{
    char grade;     // 1 byte
    int marks;      // 4 byte
};
```



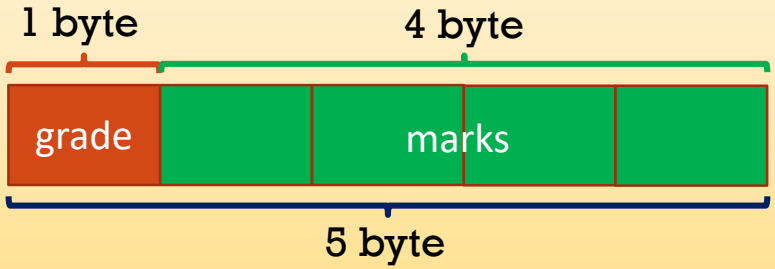
STRUCTURE VS UNION

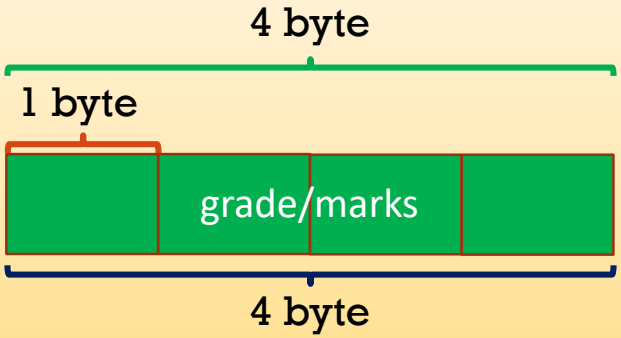
Structure

```
struct score
{
    char grade;
    int marks;
};
```

Union

```
union score
{
    char grade;
    int marks;
};
```





GETC()

It reads a single character from a given input stream and returns the corresponding integer value (typically ASCII value of read character) on success. It returns EOF on failure.

Ex:

```
void main()
{
        printf("%c", getc(stdin));
}
```

Input: g (press enter key)

Output: g

GETCHAR(), GETCH() AND GETCHE()

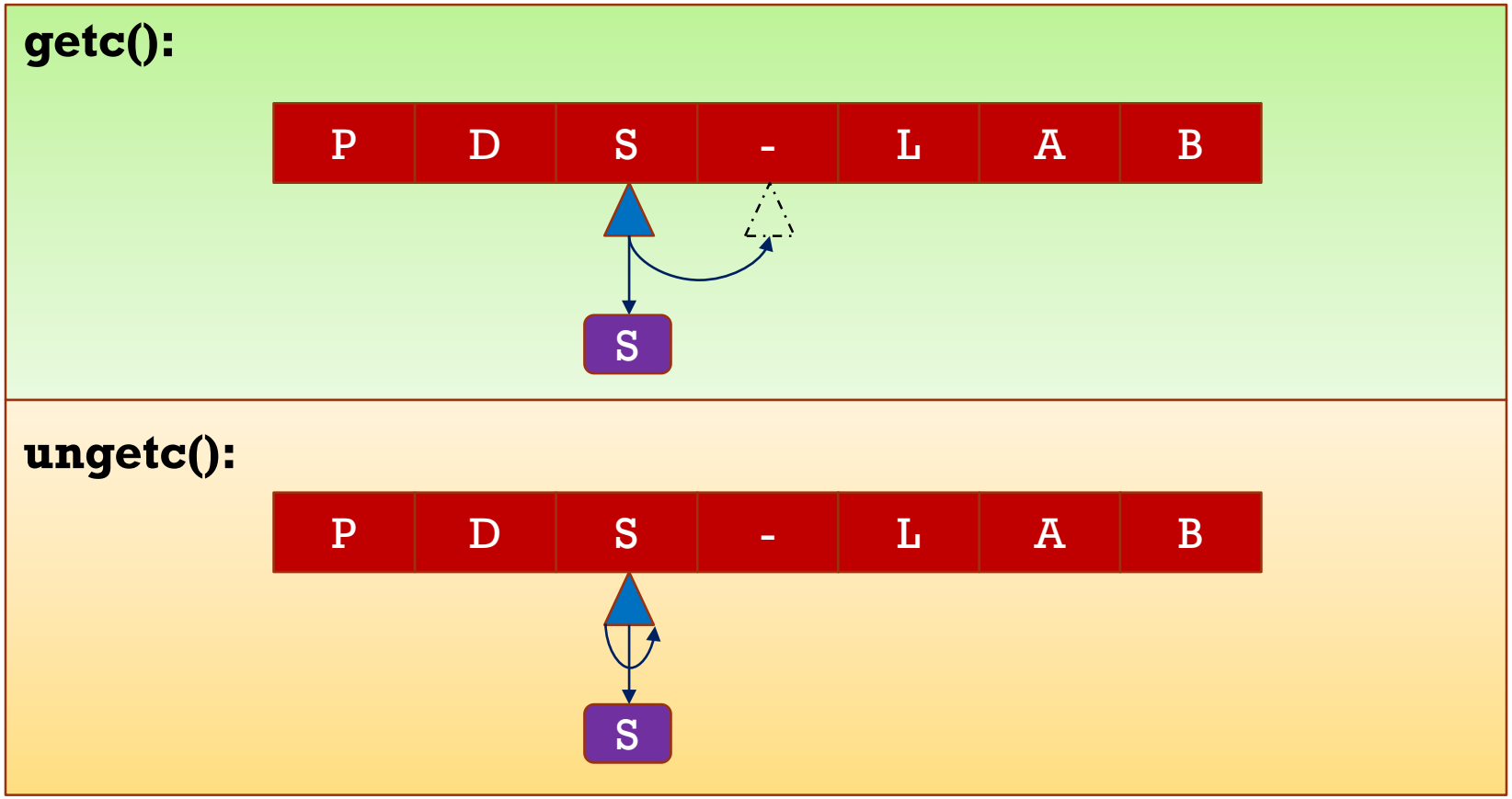
getchar(): The difference between **getc()** and **getchar()** is **getc()** can read from any input stream, but **getchar()** reads from standard input. So **getchar()** is equivalent to **getc(stdin)**.

getch(): It is a nonstandard function and is present in **conio.h** header file which is mostly used by MS-DOS compilers like Turbo C. It is not part of the C standard library or ISO C, nor is it defined by POSIX.

getche(): Like **getch()**, this is also a non-standard function present in **conio.h**. It reads a single character from the keyboard and displays immediately on output screen without waiting for enter key.

UNGETC()

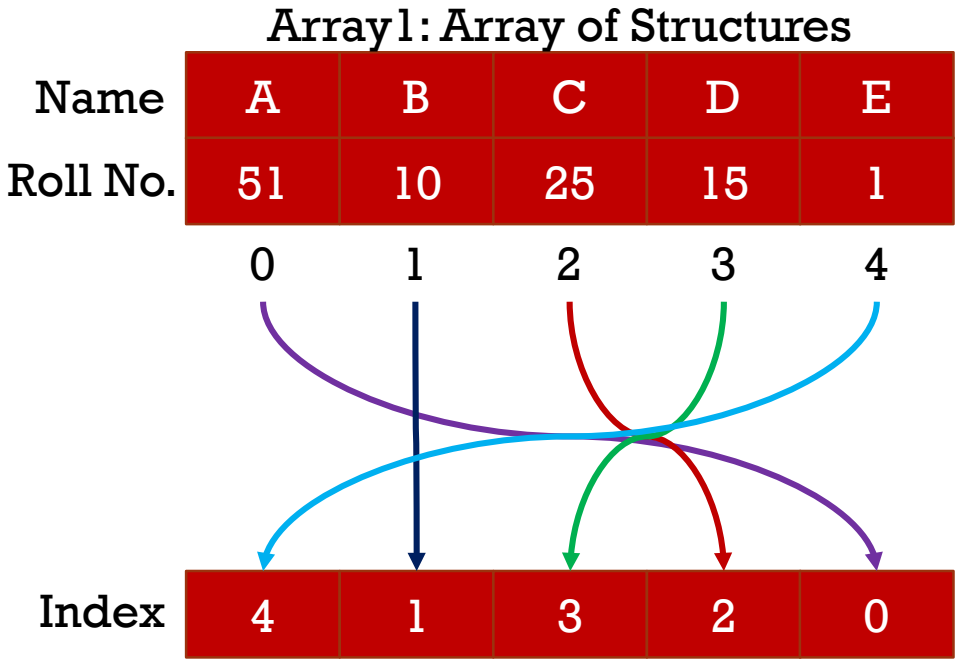
The **ungetc()** function takes a single character and shoves it back onto an input stream. It is the opposite of the **getc()** function, which reads a single character from an input stream. Also, **ungetc()** is an input function, not an output function.



READING FROM A FILE UNTIL EOF

```
void main()
{
   FILE *fp = fopen("test.txt", "r");
   int ch = getc(fp);
   while (ch != EOF)
   {
      /* display contents of file on screen */
      putchar(ch);
      /*read the next character from file*/
      ch = getc(fp);
   }
   fclose(fp);
}
```

SORT BY INDEX



Array2: Containing Index of Array1 Sorted by Roll No.

```
void main()
{
  struct student array1[5];
  int array2[5] = {0, 1, 2, 3, 4};
  //populate array1.
  for(int i=0; i<5; i++)
  {
    for (int j=i+1; j<5; j++)
    {
      if(array1[array2[i]].rollno <
                array1[array2[j]].rollno)
      {
        //Swap values
      }
    }
  }
}
```

THANK YOU!

Happy Coding