Unit- 3.2

Pointer Arithmetic

Operations:

1. Increment/Decrement of a Pointer
2. Addition of integer to a pointer
3. Subtraction of integer to a pointer
4. Subtracting two pointers of the same type
5. Comparison of pointers

1. Increment/Decrement of a Pointer:

- increments by the number equal to the size of the data type for which it is a pointer.

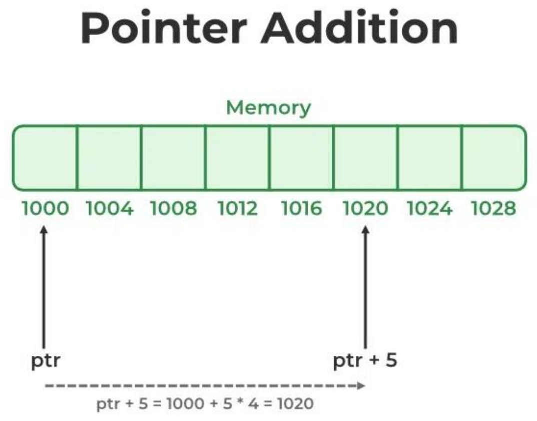
- If an integer pointer that stores **address 1000** is incremented by size of an integer new address will be 1004.

- for float it will also be incremented by 4 size of float ie 1004

- For decrement comes under subtraction, same as increment.
- Eg: int 1000; it will be decremented by size of int and new address is 996.

2. Addition of integer to a pointer:

Ptr = ptr+5 (ptr = 1000+size of int*5) = 1020

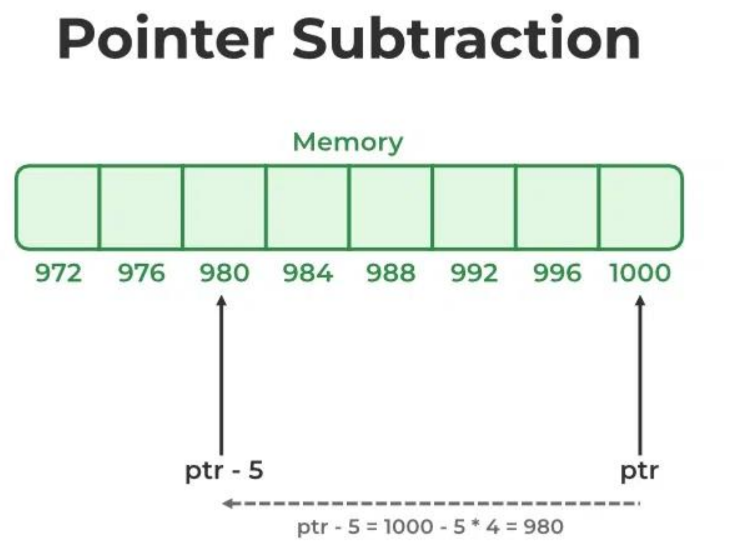


3. Subtraction of integer to a pointer:

- **Ptr = ptr-5 (ptr = 1000-size of int*5) = 980**

4. Subtraction of Two Pointers:

ptr1(address:1000) and **ptr2(addr ss:1004);** difference between addresses is 4 bytes; Since the size of int is 4 bytes, therefore the **increment between ptr1 and ptr2** is given by **(4/4) = 1**.



5. **Comparison of Pointers**

>, >=, <, <=, ==, !=

Pointer and Array:

```
int n[4] = {25, 50, 75, 100};

// Get the value of the first
element n[0] in n

printf("%d", *n);


Output:25
A[0]  a[1]       a[2]  a[3]
*a    *(a+1) *(a+2)    *(a+3)
```

```
int myNumbers[4] =
{25, 50, 75, 100};

// Get the value of the
second element in myNumbers
printf("%d\n", *(myNumbers
+ 1));

// Get the value of the third
element in myNumbers
printf("%d", *(myNumbers
+ 2));
Output:
50
75
```

Practice Problems

- If ptr is a pointer to int, having value ptr=100. After ptr++, what is the value of ptr?

a. 100 b. 101 c. 102 d. 103

- A Pointer is?

a. A keyword used to create variables.

b. A variable that stores address of an instruction.

c. A variable that stores address of other variable.

d. All of the above

- **What is the output?**

```
void main()
{
    int *pc, c;
    c = 5;
    pc = &c;
    printf("%d", *pc);
}
```

a. Address of c b. 5 c. address of pc d. error

More practice: https://gtu-mcq.com/BE/Civil-Engineering/Semester-1/3110003/3819/MCQs?q=9aZHDjblmRk=

-A pointer value refers to

a. A float value b. An integer constant

c. Any valid address in memory d. none

- Address stored in the pointer variable is of type __________

a. Integer b. Floating c. hexadecimal d. Charcter

- Consider the 32 bit compiler. We need to store address of integer variable to integer pointer. What will be the size of integer pointer?

a. 6 bytes b. 2 bytes c. 4 bytes d. 10 bytes

```
- Void main()
{
    int* pc, c;
    c = 5;
    pc = &c;
    c = 1;
    printf("%d, %d", c,*pc);
}
```

a. 1,1 b. 1,5 c. 5,1 d. error

Pointer to function

- `int *f(int a); /* function f returning an int* */`
- `int (*g)(int a); /* pointer g to a function returning an int */`
- `auto(*fp)()->int;`

- Structure Vs Union;
- A user can access individual members at a given time.
- In a union, A user can access only one member at a given time.

- Which of the following operator is used to select a member of a structure variable.

a. . (dot) b. , (comma) c. : (colon) d. ; (semicolon)

- What is the size of a C structure?

a. C structure is always 128 bytes.

b. Size of C structure is the total bytes of all elements of structure.

c. Size of C structure is the size of largest element

d. None of these

- find output

```
#include<stdio.h>

void main()

{

  int x = 10, y = 20;

int *p = &x, *q = &y; *p = *q;

*q = 30;

   printf("%d, %d", x,y);

}
```

a. x = 10, y = 20 b. x = 20, y = 30 c. x = 30, y = 20 d. x = 30, y = 30

- Which of the following cannot be a structure member?

a. Another structure b. Array

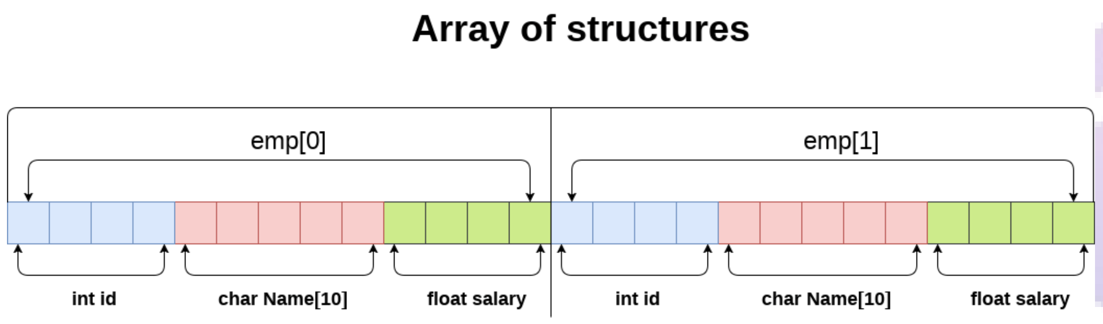
c. Function d. none

-Find the output

```
#include<stdio.h>

void main( ){

int *p, *q;

int x = 10, y = 20;

p = &x;

q = &y;

*p++;

++*q;

p = q;

*p = *q + 1;

printf("%d", *p);

}
```

a. 20 b. 21 c. 22 d. 23

Array of Structures



```
struct employee
{
    int id;
    char name[5];
    float salary;
};
struct employee emp[2];
```

sizeof (emp) = 4 + 5 + 4 = 13 bytes

sizeof (emp[2]) = 26 bytes

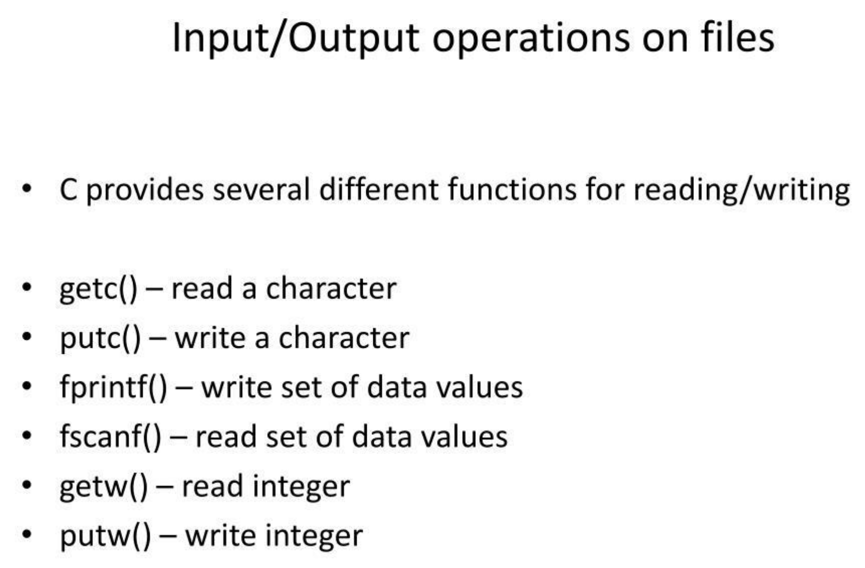
- **passing structure to function**
  - Pass by value (passing actual value as argument)
  - Pass by reference (passing address of an argument)

Structure and pointer

- Declare a Structure Pointer
  - **struct** structure_name *ptr;
- Initialization of the Structure Pointer
  - ptr = &structure_variable;
- Access Structure member using pointer:
  1. Using (*) asterisk or indirection operator and dot (.) operator.
  2. Using arrow (->) operator or membership operator.

Input/output operations on files

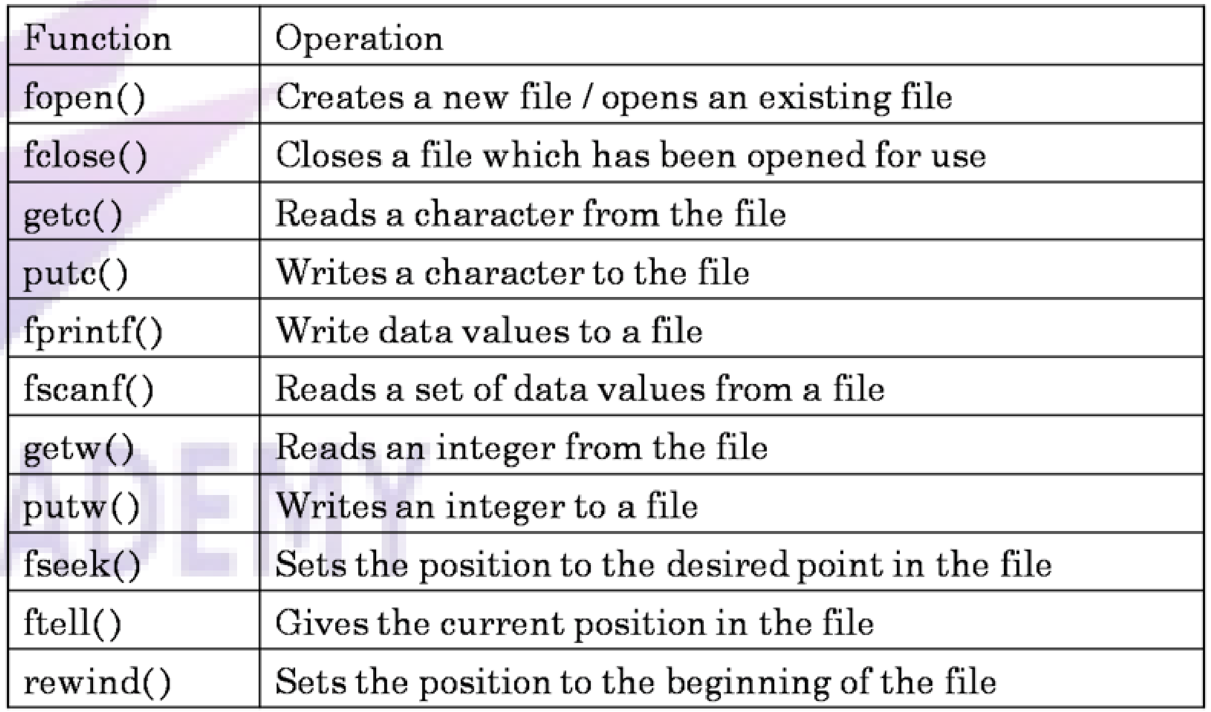
- Opening file:
  - FILE *fopen(const char * filename, const char * mode);



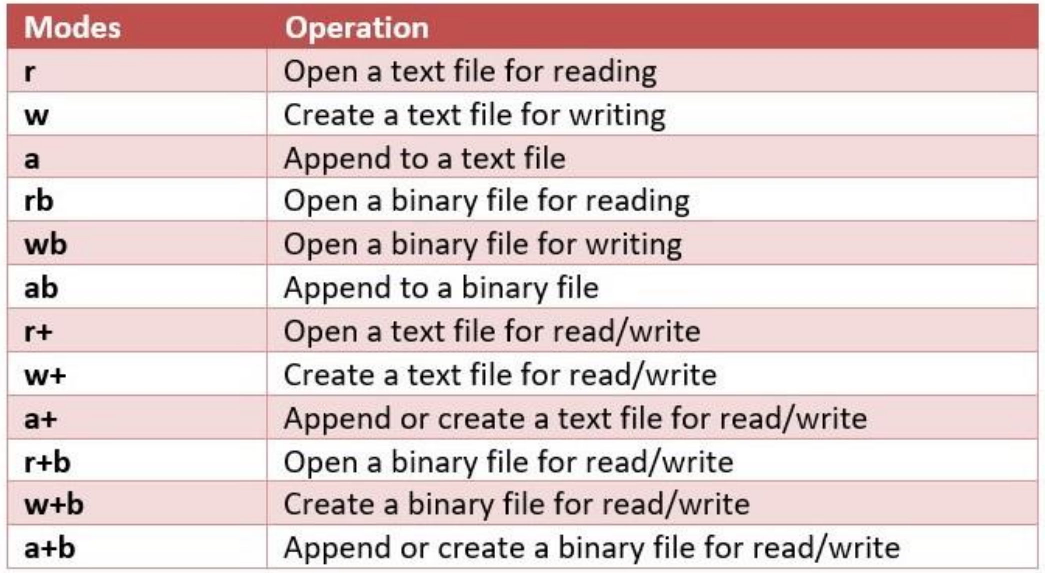
Input/Output operations on files

- C provides several different functions for reading/writing

- getc() – read a character
- putc() – write a character
- fprintf() – write set of data values
- fscanf() – read set of data values
- getw() – read integer
- putw() – write integer



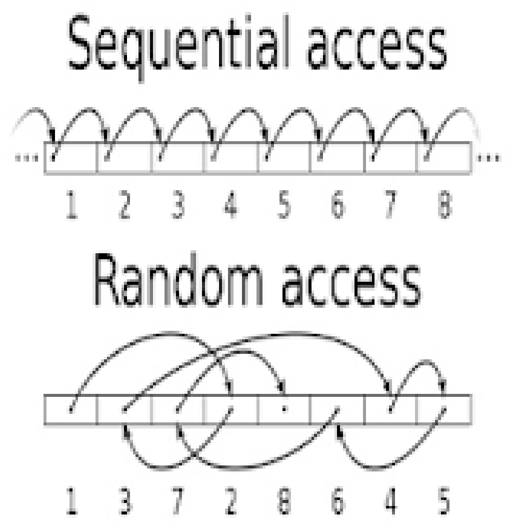
Function	Operation
fopen()	Creates a new file / opens an existing file
fclose()	Closes a file which has been opened for use
getc()	Reads a character from the file
putc()	Writes a character to the file
fprintf()	Write data values to a file
fscanf()	Reads a set of data values from a file
getw()	Reads an integer from the file
putw()	Writes an integer to a file
fseek()	Sets the position to the desired point in the file
ftell()	Gives the current position in the file
rewind()	Sets the position to the beginning of the file



Modes	Operation
r	Open a text file for reading
w	Create a text file for writing
a	Append to a text file
rb	Open a binary file for reading
wb	Open a binary file for writing
ab	Append to a binary file
r+	Open a text file for read/write
w+	Create a text file for read/write
a+	Append or create a text file for read/write
r+b	Open a binary file for read/write
w+b	Create a binary file for read/write
a+b	Append or create a binary file for read/write

- For practice problem:
- https://letsfindcourse.com/technical-questions/c/file-handling

Sequential and Random Access to File.



- **sequential access** means that a group of elements is accessed predetermined, ordered sequence

- **Random Access** files will be spited in to pieces and will be stored wherever spaces available.

- Sequential file may load faster and random access files may take time