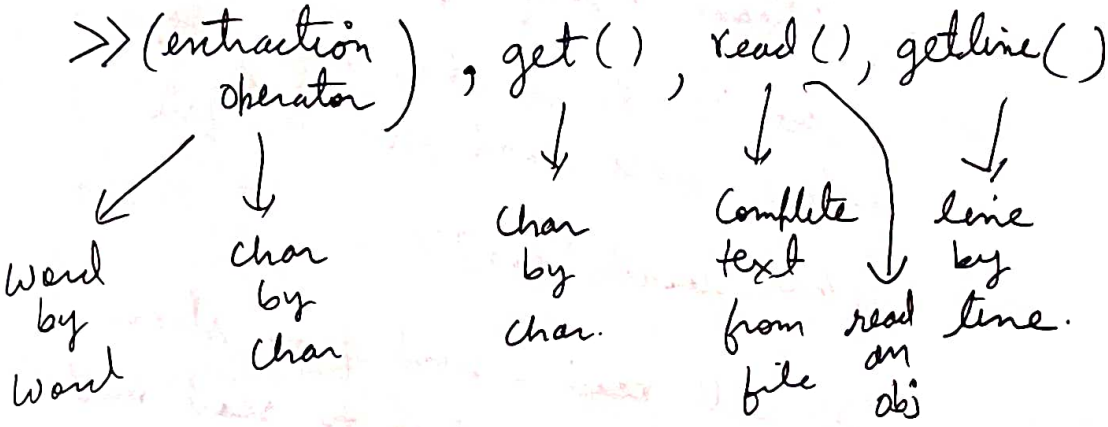
Ways to perform read operation on a file

(Unit-5)

- Read opr ~~to~~ from file can be done using



Reading from a file-1

- Read "word by word" from a file using "**>> operator**"

- >> operator can be used to read 1 word at a time from the stream.

- If a char array is provided to >>, then it will read 1 word from the file

```
char words[255];
my in >> words;
```

(my in: i/p stream; words: char array.)

Note *: >> will ignore whitespaces.

- WS are spaces, tabs, newline

- Prog to read from a file & display it

```
int main()
{
    ifstream myin("file.txt");
    ~~char~~ //create a char array to
    //store data read from the file
    char words[255];
```

- //>> will read 1 word from stream & then stop. So use loop to read mul words from the stream.

- //when all words r read, the state of stream will be false., in that case the loop will be broken.

//So loop will be as follows.

```
while(1)
{
    myin >> words;
    //read 1 word w/o whitespace.
    if(!myin) cout << words; //Print only when stream is true
    // if end of file is reached, then
    //stream will be false, so break loop
    else break;
}
```

```
myin.close();
}
```

Note. If char arr is p/v to >> then it will read only 1 word at a time.
. It will not read complete text.
. So loop needs to be used.
~~*. wsp r removed so the o/p will~~
. ** Whitespaces (wsp) r ignored by >>
So o/p will be
"This is line1 This is line2".

Reading from a file-2

. Read "char by char" from a file using ">> operator"

. >> can be used to read 1 char from the file.
. In this case char var should be p/v to the >> operator instead of char array

. eg
```
char ch;
myin >> ch;
```

. loop should be used until eof arrives.
in that case myin will be false.

Program:

```
int main()
{
    ifstream myin("file.txt");
    char ch2; // Check if file open succes
    if (!myin)
    {
        cout<<"Can't open";
        return 0;
    }

    //var to store char read from file
    char ch;

    while(1)
    {
        myin>>ch;
        //read 1 char.

        if(myin) cout<<ch;
        //myin true means eof not reached.

        else break;
        //myin false means eof ~~reached~~ reached
            so break loop.

    }

    myin.close()
}
```

Note · when a char var is p/v to >> it will read only 1 char

· If a char array is p/v then it will read 1 word

· >> ignores whitespaces · in both cases.

· Loop should be used in both cases

· o/p will be :-

"This is line 1 This is line 2".

Reading from a file - 3

· Reading "char by char" from a file using "get()" fun

· fstream provides a get fun · It reads 1 char from file.

· eg char ch;

myin.get(ch);

↓

Char read from file will be Stored in ch

//Wrong way to use get()

// ch = myin.get() // This is wrong.

· it doesn't ignores ws.

· loop shud be used with get()

Prog :-

```
int main()
{
    ifstream myin("file.txt");
    char ch;
    if(!myin)
    {
        cout<<"Can't open";
        return 0;
    }
    while(1)
    {
        myin.get(ch);
        if(myin) cout<<ch;
        else break;
    }
    myin.close();
}
```

Note :- • get() will read 1 char
• use while loop to read mul char
• it doesn't ignore ws
• O/P will be

" This is line1
This is line2".

Reading from a file - 4

- Read "complete text" from a file using "read fun".

- Read fun is a v. convenient way to read complete text from a file
- No loop is needed.
- No ws r ignored
- syntax :-

```
char *words[255];
myin.read( words, sizeof(words) );
```

words → Char array to wh. data will be stored.
↓
This param must be char *

sizeof → size of char array

Prog :-

```
int main()
{
    ifstream myin("file.txt");
    if(!myin)
    {
        cout<<"Can't open";
        return 0;
    }
    char words[255];
    //Char array to store data read from the file.
```

```
myin.read(words, sizeof(words));
cout<<words; //Print the text read
                from file;

myin.close();
}
```

// Read fun is able to read complete text from the file at a time, so no loop is required.

Reading from a file - 5

- Reading "an obj" from a file using "read fun"
- Read() is capable of reading an obj from a file.
- The file must store data in the same format as that of an obj.
- No loop is needed.
- Syntax

```
student tempobj;
myin.read((char*)&obj, sizeof(obj));
```

Prog :-

```
//Assume a student class is already
created here
  :  :  :

int main()
{
   student tempobj;
   //create a temp obj to store the data
   //read from a file

   ifstream myin("file.txt");
   if(!myin)
   {
      cout<<"Can't open";
      return 0;
   }
   myin.read((char*)&tempobj, sizeof(tempobj));
```

get a ptr to the obj & convert to (char *)

sizeof obj

- This param must be of type (char *)

```
   tempobj.get();
    // Print obj details.

   myin.close();
}
```

Reading from a file - 6

- Read "line by line" from a file using "getline()" fun
- getline() allows you to ~~write~~ read a line from the stream.
- newline char ('\n') will be ignored
- syntax:

```
char words[255];
myin.getline(words, sizeof(words));
```

- This will only read 1 line from the stream, so loop must be used to read multi lines from the stream.
- Spaces r not ignored but '\n' will be ignored.

Prog

```
int main()
{
    ifstream myin("file.txt");
    char words[255];
    if(!myin)
    {
        cout<<"Can't open";
        return 0;
    }
```

```
while (1)
{   myin.getline (words, sizeof (words));
    if (!myin) cout << words;
    else break;

}
//. This loop will read 1 line from the file
//Then it will check if the stream state
  is true or false
//Stream will be true when eof is not
  reached so it will print
//Stream will be false when eof is reached
//. So it will break loop.

  myin.clear ();

} //main ends
```

Note: The "\n" is ignored. So o/p will be.

"This is line1 This is line 2".

//All lines will be printed in 1 line

Summary of diff ways to read from a file

Method	description	use	loop required	Remark.
>> (extraction operator)	word by word	char words[255]; myin >> words	Yes	whitespaces are ignored
	char by char	char ch; myin >> ch;	"	"
get()	char by char	char ch; myin.get(ch);	"	—
read()	complete text	char words[255] myin.read (words, sizeof(words));	X	—
	object	student tempobj; myin.read (.....)	X	—
getline()	line by line	char words[255]; myin.getline (.....)	✓ Yes	'\n' is ignored

Note: :- which method to use depends on your requirement. Usually read() is a convenient way to read complete text from file

2) If text needs to be processed char by char then use get() fun to read char

3) obj can be read using read() fun.