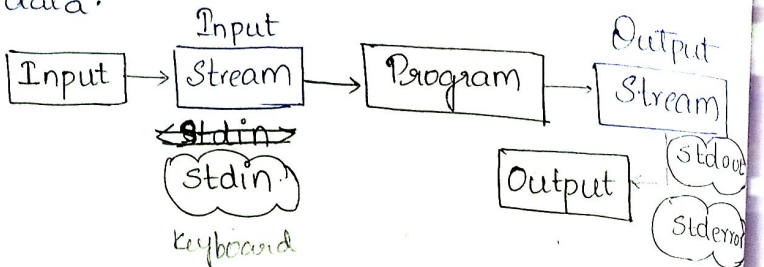
return 0;

}

02/03/2022

FILES:
- Text file: understandable/readable
- Binary file: Not a readable format

Storing & retrieving data.



* FILE → structure.

To use a file we create a pointer.

FILE *fp;

* fopen: establish a stream & open a file.

ex: fopen("input.txt", "r");

SYNTAX: fopen should contain 2 strings

fopen("file path/filename", "mode");
- mode → reading / giving data i.e writing

↓

- Establishes a link b/w input file & program.

After the opening of the file successfully: it returns a file pointer.

fp = fopen("input.txt", "r");

Modes: Types ⟹ for a text file:

1) read = "r"
2) write = "w"
3) append = "a"
4) read & write = "r+"
5) read & write = "w+"
6) read & append = "a+"

for a binary file

1) "rb" = read
2) "wb" = write
3) "ab" = append
4) "rb+" = read & write
5) "wb+" = read & write
6) "ab+" = read & append

* fopen:) read mode:

→ When file is successfully opened, it returns the file pointer.

→ But when file did not open (or) it is damaged / does not exist, it returns NULL to the program

```
fp = fopen("input.txt", "r");
if (fp == NULL)
{ printf("cannot open");
  exit(0);
}
```

- when the file is opened, pointer points to the first element of the file.
→ input stream is opened & linked to file and is closed.

2) write mode:

```
fp = fopen("input.txt", "w");
```

(not mandatory to exist)

* If the file is not present, it creates a new one and is opened to enter data.
* If the file is present, it is opened and existing content is replaced with the newly entered data and the old data is lost.
- The above cases, returns a FILE pointer.
→ But when the data is full / memory is filled, it returns NULL.
→ output stream is opened & is linked to file to enter data & closed.

3) Append Mode:

```
fp = fopen("input.txt", "a");
```

* If the given file is already existing, it opens and points to the last location and starts writing from the end. There is no overlap of the content.
* If the file doesnot exist, it is created & pointer points to 1st location.

4) Read & write Mode:

~~FILE * fp1, * fp2;~~

`fclose(filepointer);` → used to close the opened file

Read data from KB → O/P stream → Write to a file → Read from file → I/P stream → Display on monitor

read & write → first read & then write → r+
read & write → first write & then read → w+

*CHARACTER INPUT/OUTPUT FUNCTION:

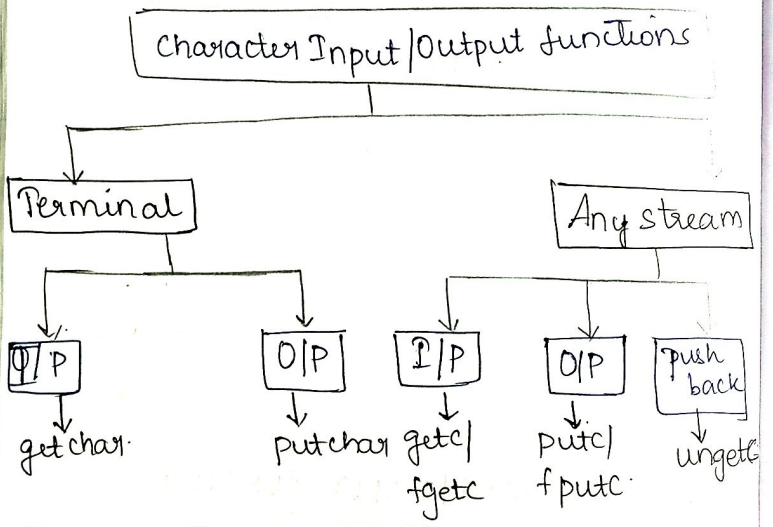
stdio.h: getchar() → read a character from keyboard.
putchar() → printing the character

conio.h: getch → waits to click a key on the keyboard

getchd → to know which key we clicked from keyboard.

* getc, fgetc, fputc, putc.

- fgetc → to read character from file
- fputc → to write character to a file
- getc → it can be linked to file stream/ pointer & stdin too.
- putc → it can be linked to file stream/ pointer & stdout too.



* FUNCTION DECLARATION:

→ int getchar(void);
→ int putchar(int o_char);
→ int getc(FILE *spin);
→ int fgetc(FILE *spin);

getc, ungetc, getc → Reads the old character

getc, getc → Reads a next character

→ int putc(int onechar, FILE *fp);
→ int fputc(int onechar, FILE *fp);
→ int ungetc(int onechar, FILE *fp);

04/03/2022

* Write a program to read the data from the keyboard and write it into a file.

→ Opening a file in write mode.

```
#include <stdio.h>
int main( ).
{ char ch;
  FILE *fp;
  fp= fopen("sample.dat", "w");
if(fp==NULL)
{ printf("File cannot be created");
    exit(0);
  }
else
{ while((ch=getchar())!=EOF)
    { fputc(ch,fp); }
```

ctrl+D → stop accepting input

```
} fclose(fp);
  return 0;
  }.
```

* Write a program to read the contents of file and display it on monitor.

```
#include <stdio.h>
#include <stdlib.h>
int main()
{ char ch;
   FILE *fp;
  fp = fopen("sample.dat", "r");
  if (fp == NULL)
  { printf("File does not exist");
    exit(0);}
  else
  { while (! feof(fp)).
    { ch = fgetc(fp);
      putchar(ch); }. }
   fclose(fp);
  return 0; }
```

feof(fp) → return 1 pointer at end of file

* Write a program to copy the contents of one file to another file.

```
#include <stdio.h>
int main()
{ char ch;
  FILE *fp1, *fp2;
  fp1 = fopen("sample.dat", "r");
  fp2 = fopen("output.dat", "w");
  if (fp1 == NULL).
   { printf("File does not exist");
     exit(0);
   }
  if (fp2 == NULL)
  { printf("File does not exist");
    exit(0);
  }
else{
while (! feof(fp1)).
 { ch = fgetc(fp1);
     fputc(ch, fp2); }
   }
   fclose(fp);
  return 0; }
```

*Write a program to store student records into a file.

```
#include <stdlib.h>
#include <stdio.h>
struct student
{ int rno;
  char name[30];
  int tmarks;
};
int main()
{ int n,i;
  struct student s,q;
  FILE *fp;
  fp=fopen("sample.dat","wb");
  printf("Enter no.of students");
  scanf("%d",&n);
  for(i=1;i<=n;i++)
  { printf("Enter student record");
    printf("Enter rno");
    scanf("%d",&s.rno);
    printf("Enter name");
    scanf("%s",s.name);
    printf("Enter marks");
    scanf("%d",&s.tmarks);
    fwrite(&s,sizeof(s),1,fp);
  } fclose(fp);
  return 0;
}
```

For binary file
fwrite
fread

*Write a program to read the student details from file and display it on the monitor.

continuing above program

```
fp=fopen("student.dat","rb");
for(i=1;i<=n;i++)
{ fread(&q,sizeof(q),1,fp);
  printf("%d\t%s\t%d\n",q.rno,
         q.name,
         q.tmarks);
} fclose(fp);
return 0;
}
```

ftell: current location
fseek: get the pointer to a particular location
rewind: get back to beginning.

→ Reading from file:

* fscanf(fp,"%d",&b); ⇒ Any datatype
fprintf(fp,"%d",c);

fscanf ⇒ reads complete integer from a file
fprintf ⇒ prints complete integer to a file

05/03/2022

* Binary files : storing in the form of bytes
* Text files : characters (ascii).

→ TEXT FILE: → fgetc(fp) / fscanf(fp, "%d", &c);
fputc(fp) Characters
fprintf(fp, "%d", c);
write the data of diff. datatypes.

→ BINARY FILE: → fwrite(address of structure variable, size, no. of records)

→ fwrite(&structure variable, sizeof(SV), no. of records, fp);

→ fread(&structure variable, sizeof(SV), no. of records, fp);

⇒ For random access of file:

1) ftell()
2) fseek()
3) rewind()

• ftell:

⇒ long int ftell(file pointer);

* to give the current file pointer location.

Eg: FILE *fp;
fp = fopen("file.c", "r");
printf("%ld", ftell(fp));

fseek()

• fseek:

⇒ fseek(file pointer, offset, position).

* point the file pointer to a particular location in file).

⇒ offset: tells how many bytes you want to move the pointer → offset = +ve: forward
offset = -ve: backward

* position: can have the values.
0 → from beginning.
1 → from current location.
2 → from the end of the file.

SEEK_SET : 0
SEEK_CUR : 1 } alternate
SEEK_END : 2

fseek(fp, 3, SEEK_END); } Errors
fseek(fp, -3, SEEK_SET);

fseek(fp, 0, SEEK_SET); → valid and locates to beginning location

• rewind:

rewind(fp);

* to make the file pointer bring back to first location.

* Write a program to read the details of a specified employee.

```
#include <stdio.h>
typedef struct
    { char name[20];
      int id;
      char department[10];
      int age;
    } EMPLOYEE;
int main()
{   int n;
    FILE *fp;
  fp=fopen("employee.dat","rb");
    EMPLOYEE e1;
  if (fp==NULL)
  {
    printf("File is not opened");
  }
 else
 {
   printf("Enter the record number to be
                        read");
   scanf("%d",&n);
  fseek(fp,(n-1)*sizeof(E),0);
  fread(&e1,sizeof(E1),1,fp);
   printf("%s\n", e1.name);
   printf("%d\n", e1.id);
   printf("%d\n", e1.age);
 }fclose(fp);
 return 0; }.
```

*Write a program to read 5 student records to a file: