File I/O Functions Part-I

fscanf, fprintf

Four Ways to Read and Write Files

- Formatted file I/O
- Get and put a character
- Get and put a line
- Block read and write

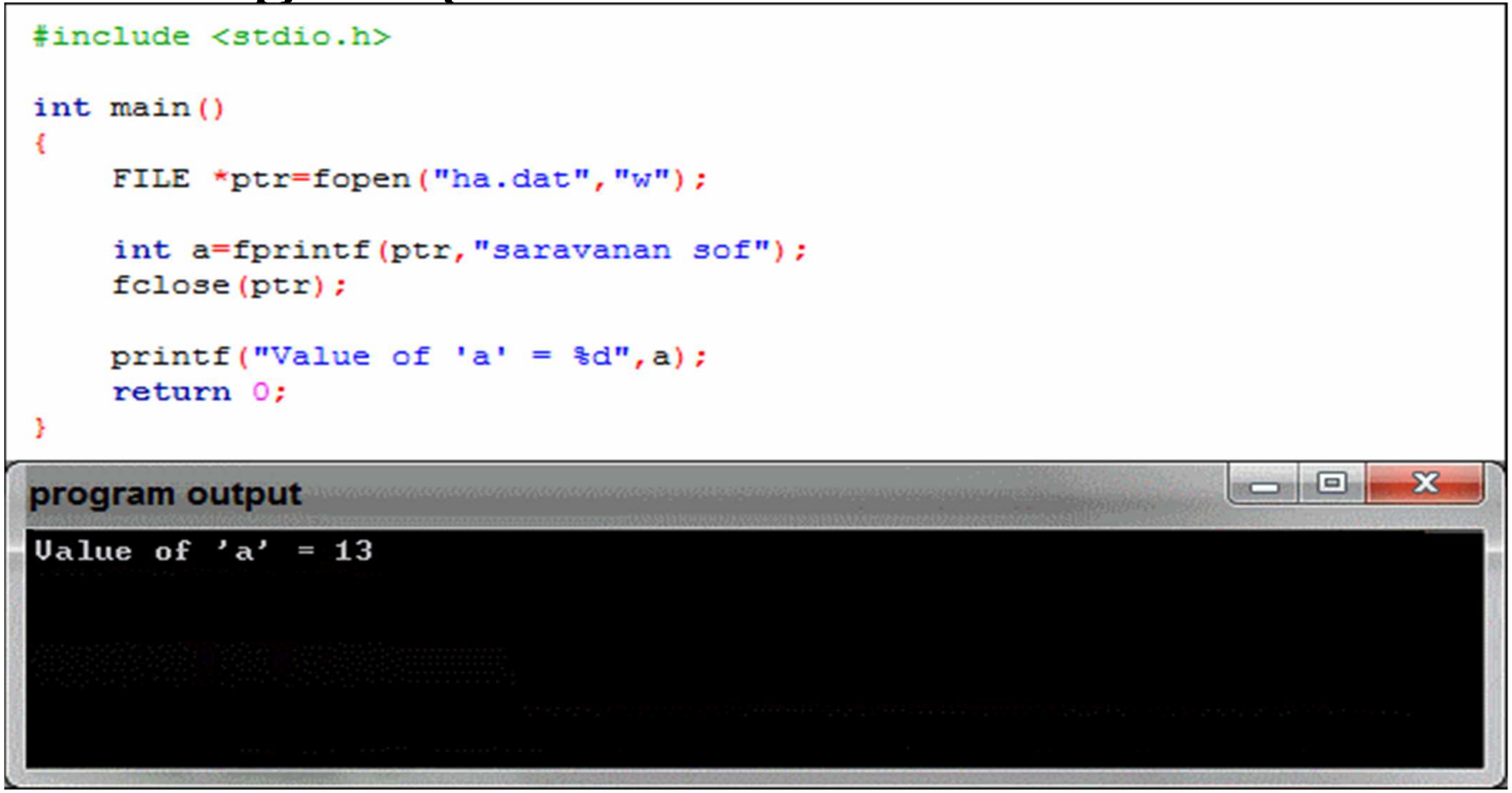
Formatted File I/O

- Formatted File input is done through fscanf:
  - int fscanf (FILE *stream, const char *format-string, argument-list);
- Formatted File output is done through fprintf:
  - int fprintf(FILE *stream, const char *format-string, argument-list);

Writing to a file: fprintf( )

- fprintf() works exactly like printf(), except that its first argument is a file pointer. The remaining two arguments are the same as printf
- The behaviour is exactly the same, except that the writing is done on the file instead of the display

Working of fprintf



```
#include <stdio.h>

int main()
{
    FILE *ptr=fopen("ha.dat","w");

    int a=fprintf(ptr,"saravanan sof");
    fclose(ptr);

    printf("Value of 'a' = %d",a);
    return 0;
}
```

program output

```
Value of 'a' = 13
```

Example Program fprintf

```
#include<stdio.h>
int main()
{
 int i, n=2;
 char str[50];
 //open file sample.txt in write mode
 FILE *fptr=fopen("sample.txt", "w");
 if (fptr == NULL)
 {
  printf("Could not open file");
  return 0;
 }
```

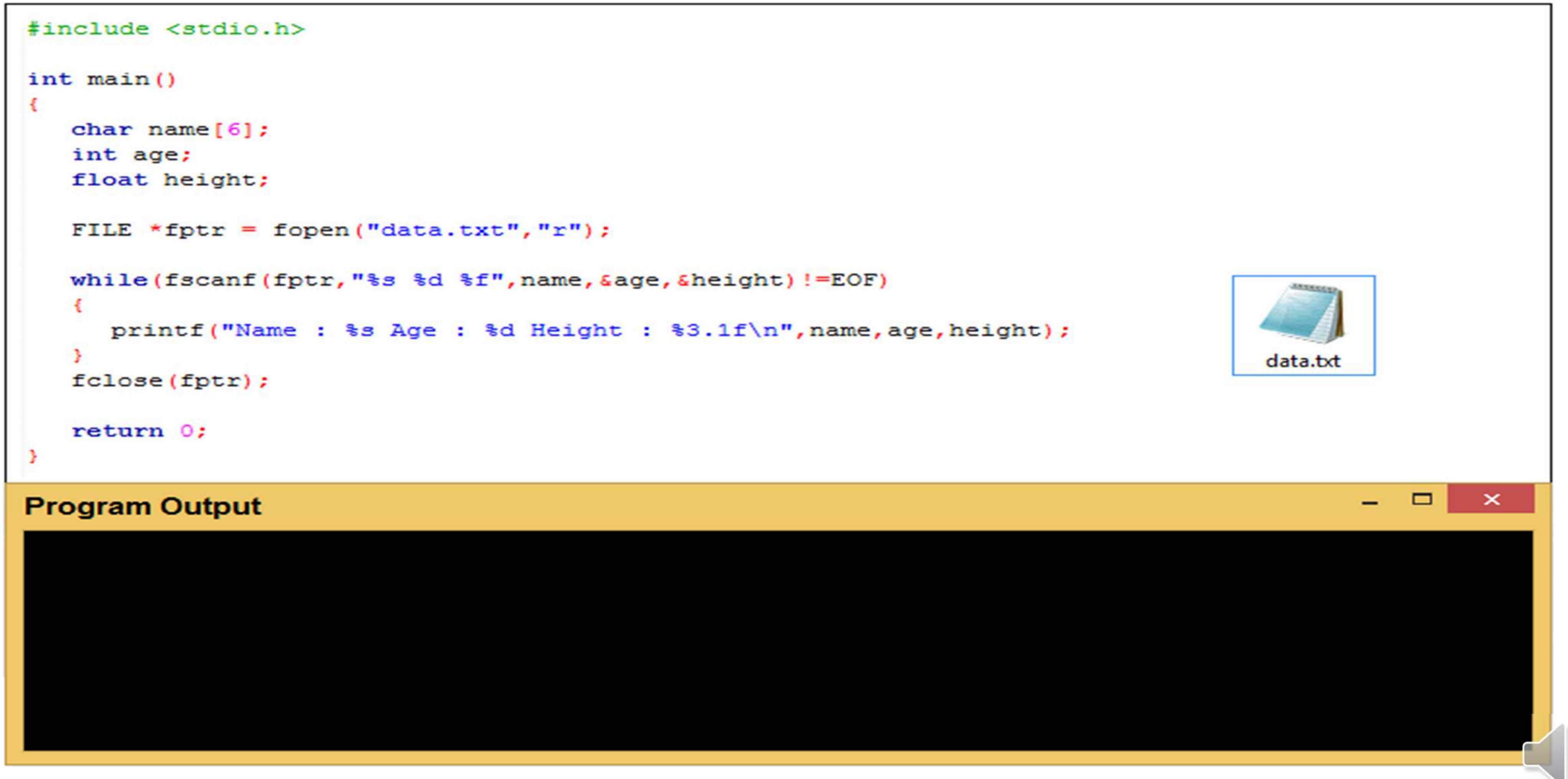
```
 for (i=0; i<n; i++)
 {
  puts("Enter a name");
  gets(str);
  fprintf(fptr,"%d.%s\n", i, str);
 }
 fclose(fptr);
 return 0;
}
```

```
Input:                    sample.txt
ABC                         0. ABC
DEF                         1. DEF
```

Reading from a file: fscanf( )

- fscanf() works like scanf(), except that its first argument is a file pointer. The remaining two arguments are the same as scanf
- The behaviour is exactly the same, except
  - The reading is done from the file instead of from the keyboard (think as if you typed the same thing in the file as you would in the keyboard for a scanf with the same arguments)
  - The end-of-file for a text file is checked differently (check against special character EOF)
  - Return Value
    - This function returns the number of input items successfully matched and assigned, which can be fewer than provided for, or even zero in the event of an early matching failure.

Working of fscanf



```c
#include <stdio.h>

int main()
{
    char name[6];
    int age;
    float height;

    FILE *fptr = fopen("data.txt","r");

    while(fscanf(fptr,"%s %d %f",name,&age,&height)!=EOF)
    {
       printf("Name : %s Age : %d Height : %3.1f\n",name,age,height);
    }
    fclose(fptr);

    return 0;
}
```

```c
/*c program demonstrating fscanf and its
usage*/
#include<stdio.h>
int main()
{
        FILE* ptr = fopen("abc.txt","r");
        if (ptr==NULL)
        {
                printf("no such file.");
                return 0;
        }


        char buf[100];
        while (fscanf(ptr,"%*s %*s %s ",buf)==1)
                printf("%s\n", buf);

        return 0;
}
```

Assuming that abc.txt has content in below format

```
        NAME      AGE       CITY
        abc        12       Hyderabad
        bef        25       Delhi
        cce        65       Bangalore
```

Output:

```
Hyderabad
Delhi
Bangalore
```