Working with Files

Program Persistent Data

Lecture 7

Review

- In C there are buffers required to work with files.
- Streams are declared using `FILE *fp;`
- These streams are required for each file that you work on.
- To open and use the stream, error check that the file exists then close when finished:

```
fp = fopen("write.txt","w");
if (fp == NULL)
    {printf("Can’t open file.\n");}
fclose(fp);
```

Review – *text* file <stdlib.h>

Instruction	**Meaning**
`fgetc(fp)`	Read a char from file using stream.
`fputc(fp)`	Write a char to file using stream.
`fgets(string,size,fp)`	Read a string from file using stream. It reads a string of a specified size.
`fputs(string,fp)`	Write a string to file using stream.
`fprintf(fp,“Hi %s, you are %i”,s,a)`	Write the content to the file using stream.
`fscanf(fp,”%s %s %i”,a,b,&c)`	Read a formatted line from file using stream.

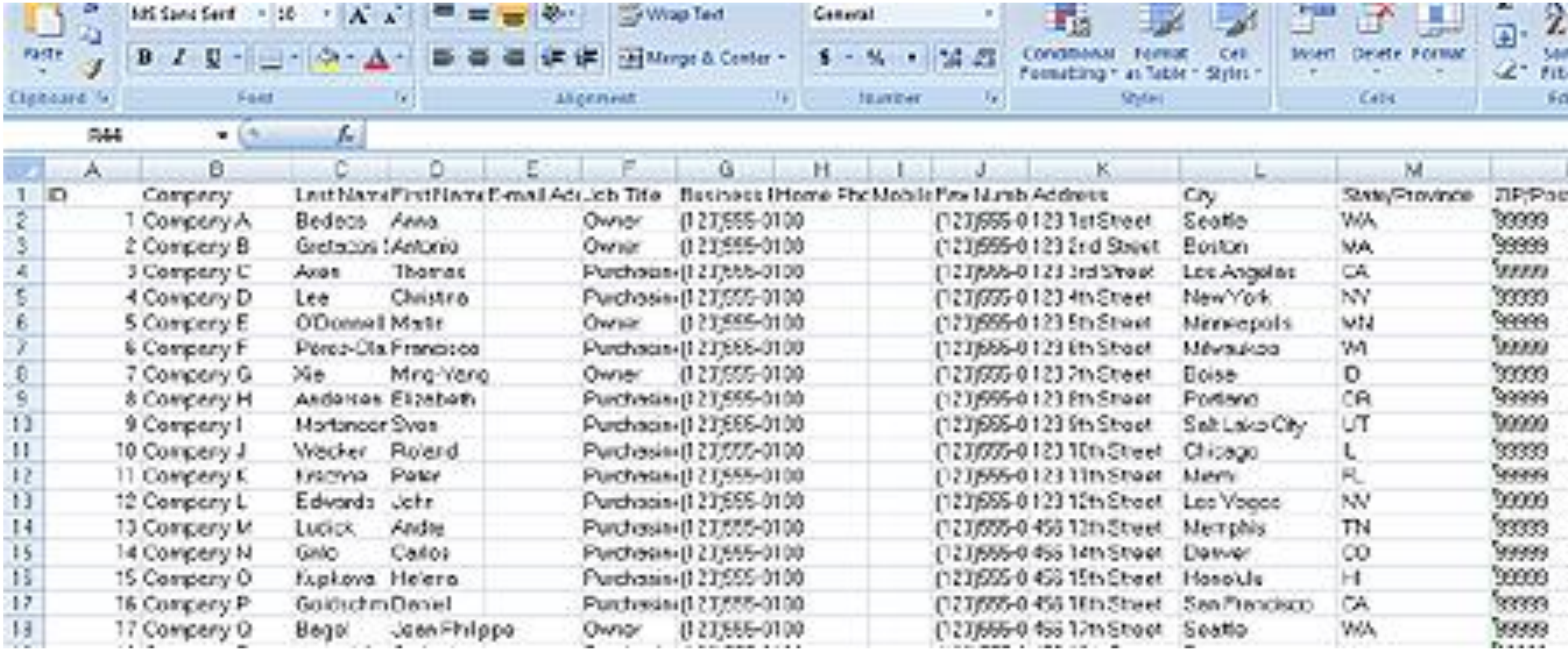
#include <string.h>

Name	Example	Meaning
1. Strlen()	`len=strlen(str);`	Get the length of a string
2. strcmp	`strcmp(str, "jane" )`	Compare 2 strings
3. strcpy	`strcpy(str, “jane”);`	Copy a strings to another
4. strcat	`strcat( str, “ Ferris” );`	Concatenate 2 strings
5. strstr	`Strstr(str, “jane”)`	Look for a substring in a string

Non text files

Name	Example	Meaning
fread	fread(var, size, number, FILEpointer);	Read from position in file
fwrite	fwrite(var, size, number, FILEpointer);	Write to position in file
fseek	fseek(FILE, offset in byte, whence); SEEK_SET/ _CUR or _END	Go to 1 of 3 places; start, current or end of file
ftell	ftell(FilePointer);	Where is the pointer now?
rewind	rewind(FilePointer);	Point to start if the file

Structs as records in a DB



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	ID	Company	Last Nam	First Nam	E-mail Ad	Job Title	Business I	Home Pho	Mobile	Fax Numb	Address	City	State/Province	ZIP/Pos
2	1	Company A	Bedecs	Anna		Owner	(123)555-0100			(123)555-0	123 1st Street	Seattle	WA	99999
3	2	Company B	Gratacos S	Antonio		Owner	(123)555-0100			(123)555-0	123 2nd Street	Boston	MA	99999
4	3	Company C	Axen	Thomas		Purchasin	(123)555-0100			(123)555-0	123 3rd Street	Los Angeles	CA	99999
5	4	Company D	Lee	Christina		Purchasin	(123)555-0100			(123)555-0	123 4th Street	New York	NY	99999
6	5	Company E	O'Donnell	Martin		Owner	(123)555-0100			(123)555-0	123 5th Street	Minneapolis	MN	99999
7	6	Company F	Pérez-Ola	Francisco		Purchasin	(123)555-0100			(123)555-0	123 6th Street	Milwaukee	WI	99999
8	7	Company G	Xie	Ming-Yang		Owner	(123)555-0100			(123)555-0	123 7th Street	Boise	ID	99999
9	8	Company H	Andersen	Elizabeth		Purchasin	(123)555-0100			(123)555-0	123 8th Street	Portland	OR	99999
10	9	Company I	Mortensen	Sven		Purchasin	(123)555-0100			(123)555-0	123 9th Street	Salt Lake City	UT	99999
11	10	Company J	Wacker	Roland		Purchasin	(123)555-0100			(123)555-0	123 10th Street	Chicago	IL	99999
12	11	Company K	Krschne	Peter		Purchasin	(123)555-0100			(123)555-0	123 11th Street	Miami	FL	99999
13	12	Company L	Edwards	John		Purchasin	(123)555-0100			(123)555-0	123 12th Street	Las Vegas	NV	99999
14	13	Company M	Ludick	Andre		Purchasin	(123)555-0100			(123)555-0	456 13th Street	Memphis	TN	99999
15	14	Company N	Grilo	Carlos		Purchasin	(123)555-0100			(123)555-0	456 14th Street	Denver	CO	99999
16	15	Company O	Kupkova	Helena		Purchasin	(123)555-0100			(123)555-0	456 15th Street	Honolulu	HI	99999
17	16	Company P	Goldschm	Daniel		Purchasin	(123)555-0100			(123)555-0	456 16th Street	San Francisco	CA	99999
18	17	Company Q	Bagel	Jean Philippe		Owner	(123)555-0100			(123)555-0	456 17th Street	Seattle	WA	99999

The key field

- Within structure arrays the structs are referred to as Records.
- Usually one field of the record is used as a key, the value used to sort and identify each record
- A good key is unique, but in many applications a unique key is not needed
- The field representing the key is called the key fields. All the other are the non-key fields. Maybe more than one field could be the key.

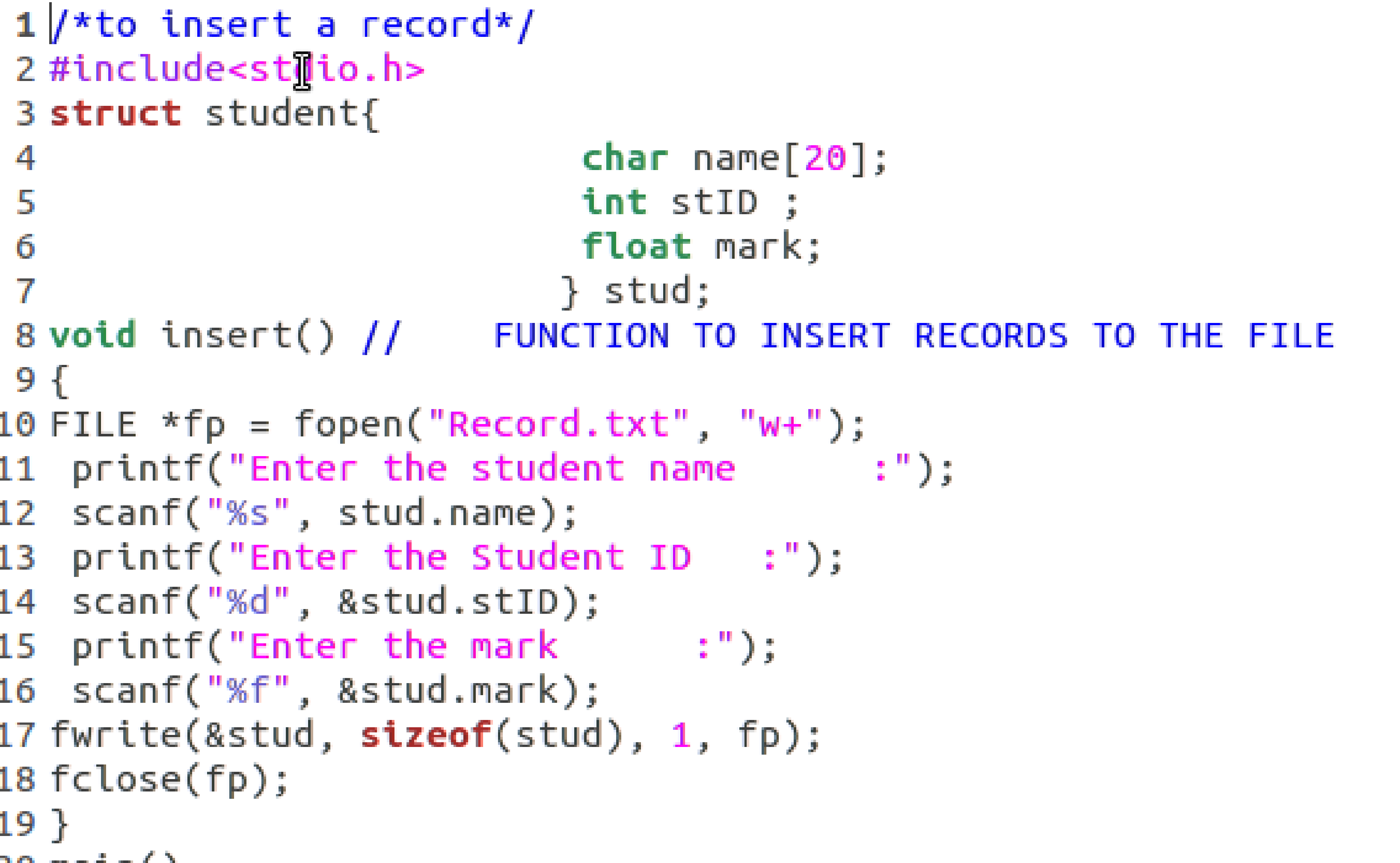
Struct of lecture

- In our examples we refer to a file containing record of the following struct:

```
Struct student{
   char name[20];
   int stID ;
   float mark;
};
typedef  student typeS;
main(){
      typeS student;
```

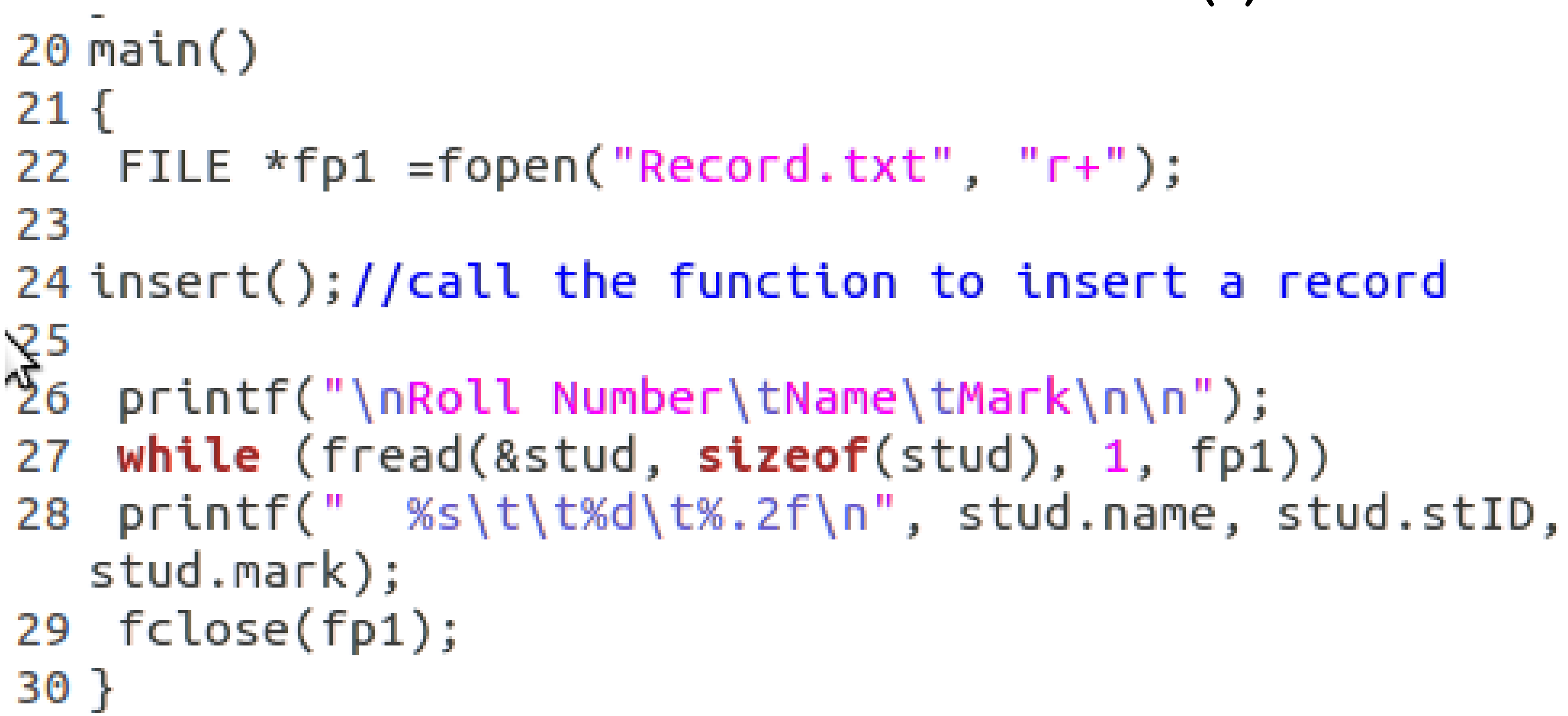
StID is a unique key

How to insert a new record?



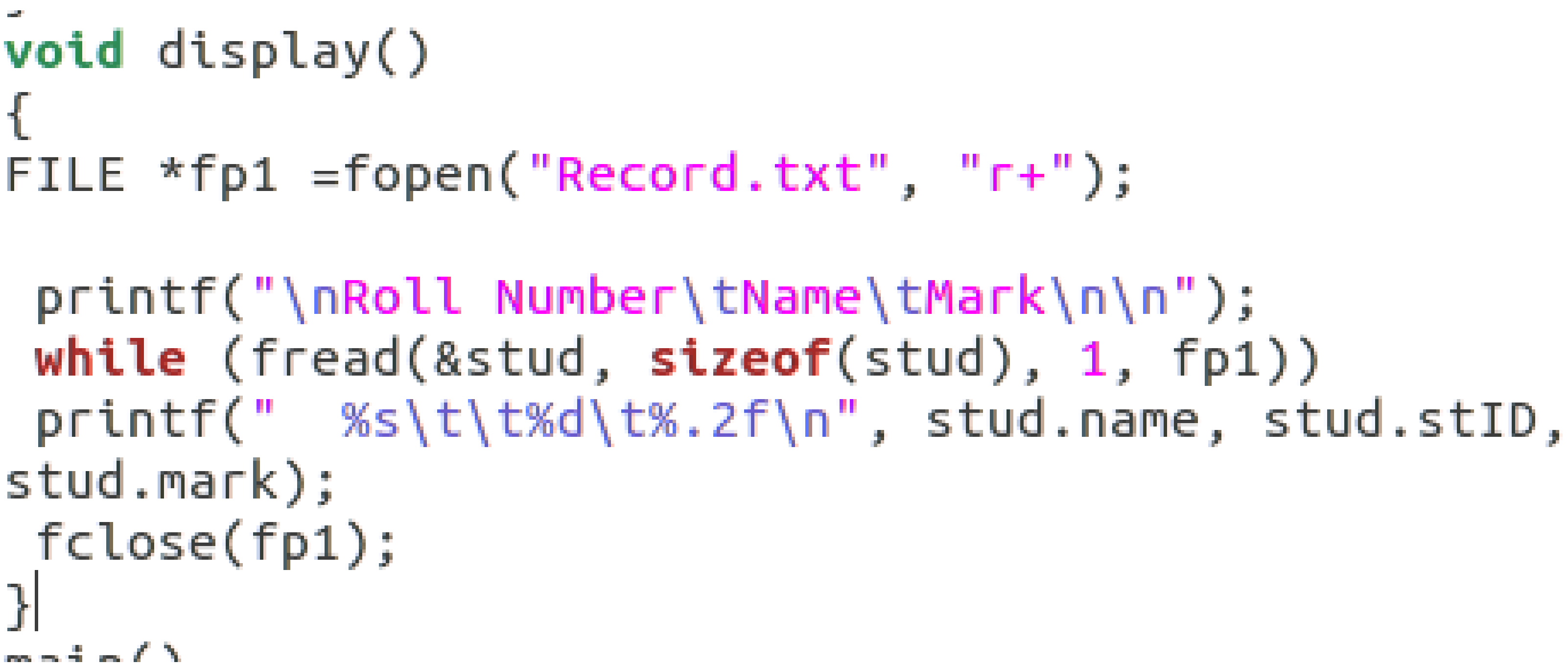
```c
/*to insert a record*/
#include<stdio.h>
struct student{
                        char name[20];
                        int stID ;
                        float mark;
                       } stud;
void insert() //    FUNCTION TO INSERT RECORDS TO THE FILE
{
FILE *fp = fopen("Record.txt", "w+");
 printf("Enter the student name      :");
 scanf("%s", stud.name);
 printf("Enter the Student ID   :");
 scanf("%d", &stud.stID);
 printf("Enter the mark      :");
 scanf("%f", &stud.mark);
fwrite(&stud, sizeof(stud), 1, fp);
fclose(fp);
}
main()
```

Call the `insert()`

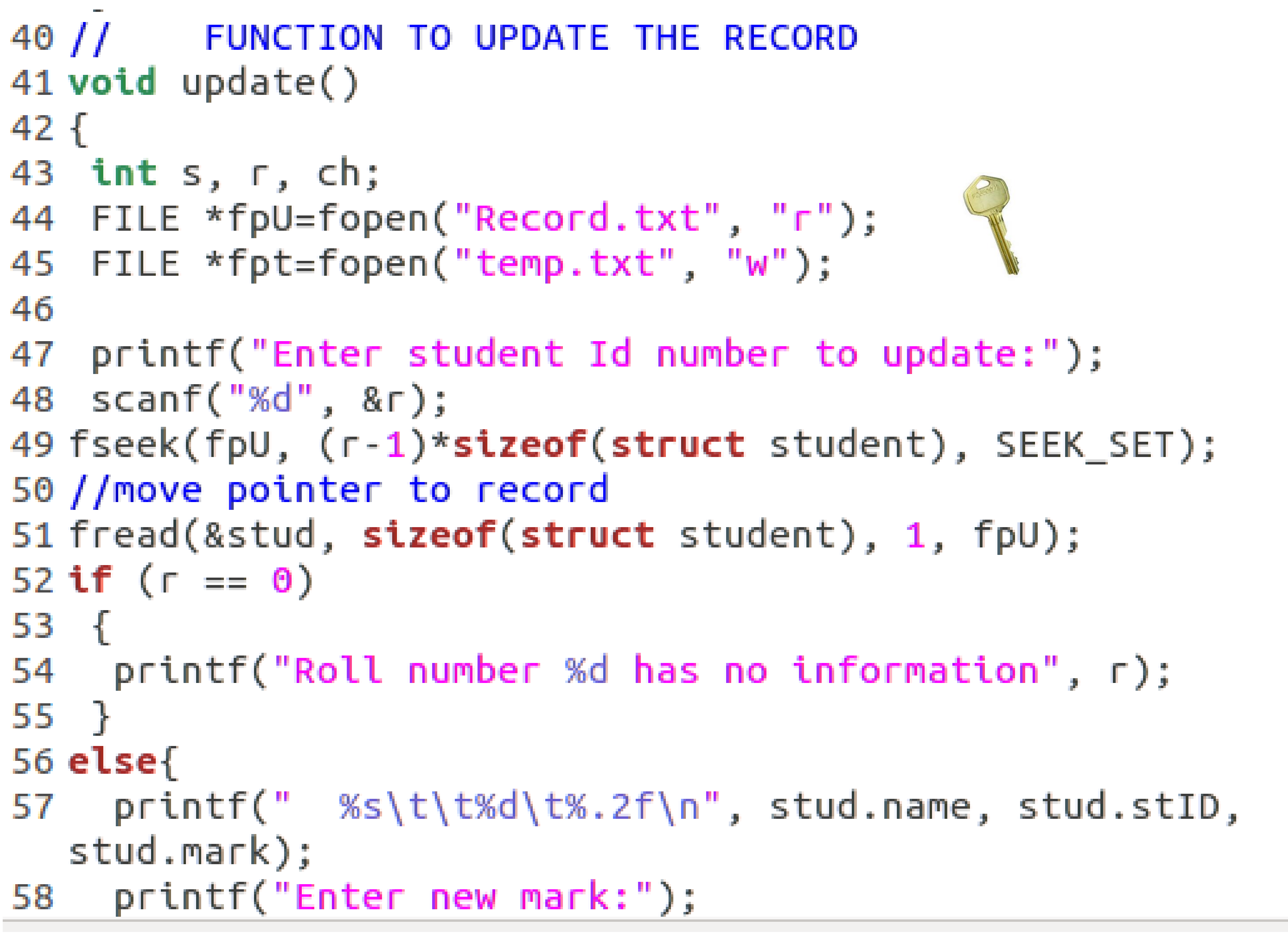


```
main()
{
 FILE *fp1 =fopen("Record.txt", "r+");

insert();//call the function to insert a record

 printf("\nRoll Number\tName\tMark\n\n");
 while (fread(&stud, sizeof(stud), 1, fp1))
 printf("  %s\t\t%d\t%.2f\n", stud.name, stud.stID, stud.mark);
 fclose(fp1);
}
```

Print out becomes `display()`

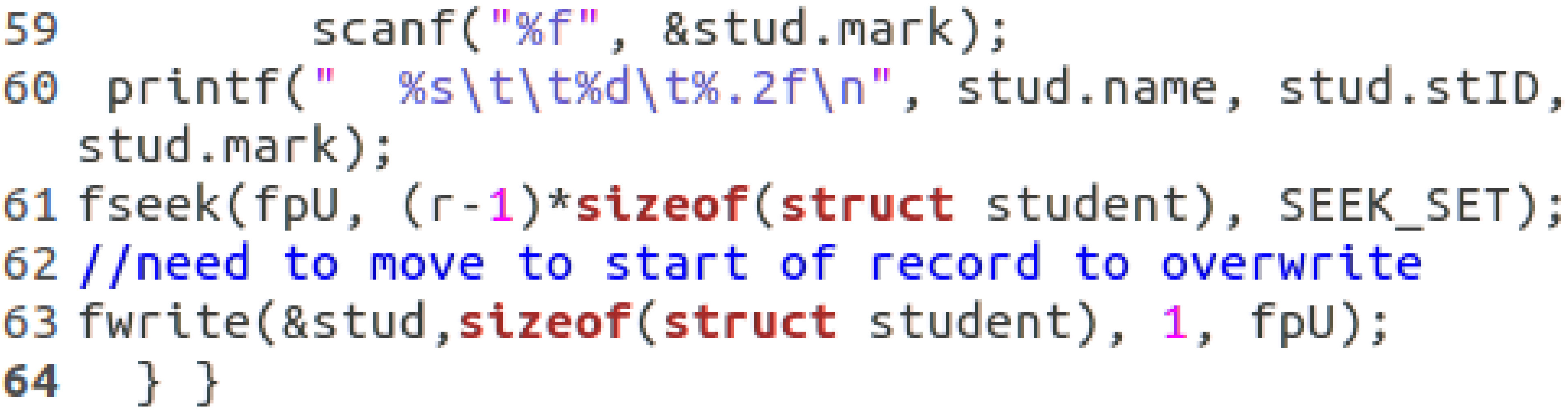


```
void display()
{
FILE *fp1 =fopen("Record.txt", "r+");

 printf("\nRoll Number\tName\tMark\n\n");
 while (fread(&stud, sizeof(stud), 1, fp1))
 printf("  %s\t\t%d\t%.2f\n", stud.name, stud.stID,
stud.mark);
 fclose(fp1);
}
main()
```



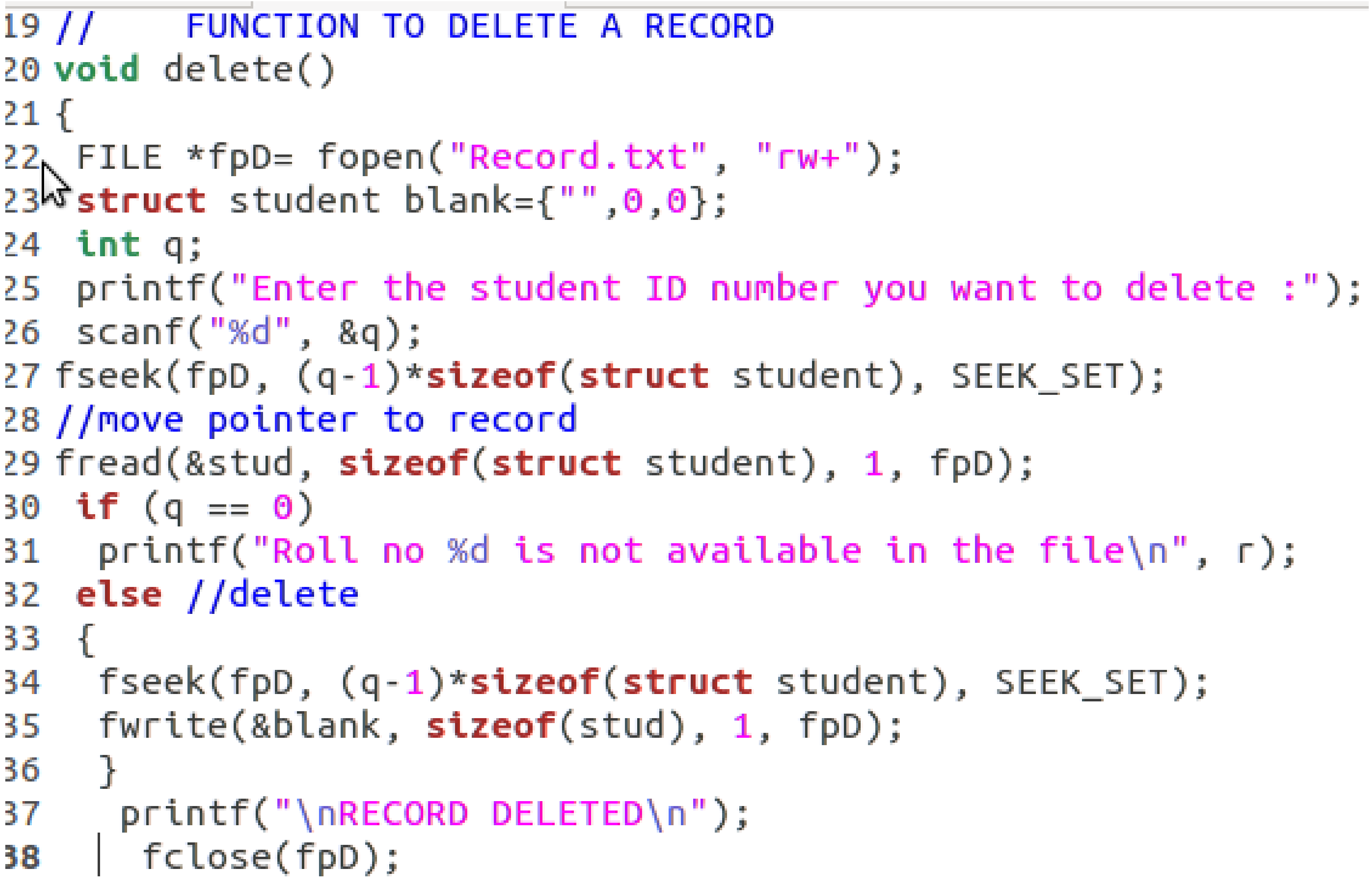
```c
//     FUNCTION TO UPDATE THE RECORD
void update()
{
 int s, r, ch;
 FILE *fpU=fopen("Record.txt", "r");
 FILE *fpt=fopen("temp.txt", "w");

 printf("Enter student Id number to update:");
 scanf("%d", &r);
fseek(fpU, (r-1)*sizeof(struct student), SEEK_SET);
//move pointer to record
fread(&stud, sizeof(struct student), 1, fpU);
if (r == 0)
 {
  printf("Roll number %d has no information", r);
 }
else{
  printf("  %s\t\t%d\t%.2f\n", stud.name, stud.stID, stud.mark);
  printf("Enter new mark:");
```

update()



```
        scanf("%f", &stud.mark);
 printf("  %s\t\t%d\t%.2f\n", stud.name, stud.stID,
stud.mark);
fseek(fpU, (r-1)*sizeof(struct student), SEEK_SET);
//need to move to start of record to overwrite
fwrite(&stud,sizeof(struct student), 1, fpU);
  } }
```

Deleting records



```
//    FUNCTION TO DELETE A RECORD
void delete()
{
 FILE *fpD= fopen("Record.txt", "rw+");
 struct student blank={"",0,0};
 int q;
 printf("Enter the student ID number you want to delete :");
 scanf("%d", &q);
fseek(fpD, (q-1)*sizeof(struct student), SEEK_SET);
//move pointer to record
fread(&stud, sizeof(struct student), 1, fpD);
 if (q == 0)
  printf("Roll no %d is not available in the file\n", r);
 else //delete
 {
  fseek(fpD, (q-1)*sizeof(struct student), SEEK_SET);
  fwrite(&blank, sizeof(stud), 1, fpD);
  }
   printf("\nRECORD DELETED\n");
    fclose(fpD);
```

Command line arguments

```
$./filename.c file.txt


main(int argc, char
*argv[])
{
for (int i = 1; i <
argc; i++)
{ printf("%s\n",
argv[i]); }
  }
//this will output the name
of each of the command line
arguments (not the filename
as starts on 2nd elemnt)
```

- Passing command line arguments to main uses argc and argv .
- Argc is the number of arguments being passed
- The first of argv is always the name of the program (*filename*) .
- argv is the pointer to the file to be passed to the main specified in argc (file.txt).
- If multiple files are passed - pass with a space between

```
$./filename.c file.txt
file1.txt
```

Command line arguments

```
#include <stdio.h>
main( int argc, char *argv[] )
 {
FILE *fp;
if( argc == 2 )
{ printf("The argument supplied is %s\n",
argv[1]); }
else if( argc > 2 )
{ printf("Too many arguments supplied.\n"); }
else { printf("One argument expected.\n"); }
}
else
fp=fopen(argv[1],”r”);
...
```

time()

- `time()` gives us the number of seconds since 1970 which is a long integer.
- Used to seed random number but also useful to get the current time to time stamp transactions etc.

```
#include <stdio.h>
#include <time.h>
main()
{
printf("The current time is
%ld\n",time(NULL));
}
//the null pointer constant is used for the
time function similar to setting as void
As time () requires an argument, wont
compile without the NULL pointer
```

time()

- Seconds since 1970 is not very helpful for humans but good for time stamps & seeding the random #s.
- For timestamping use a time type `time_t` also defined within the <time.h> file.
- Use the time function to assign the time to the variable now:

```
{
time_t now;
time(&now);
printf("The recorded time was
%ld\n",time);
}
```

ctime()

- A very useful time function for humans is ctime().
- It converts the time() to a string.

```
{
time_t now;
time(&now);
printf("The recorded time was 
%s\n",ctime(&now));
}
//The ctime() is a string so don’t 
forget %s
```

localtime()

```
#include <stdio.h>
#include <time.h>
main()
{    time_t now;
     struct time *rightNow;

     time(&now);
     rightNow = localtime(&now);
     printf("Today is %d/%d at %d:%d\n",
               rightNow->tm_mon,
               rightNow->tm_mday,
               rightNow->tm_hour,
               rightNow->tm_min );     }
```

Macros

Very useful small functions

```
#define MAX 100
```

Useful macros

```
#define MIN(A,B) ( (A)<(B)?(A):(B) )
//Which is the maximum of the 2 numbers
#define ABS(N) ( (N) <0 ?-(N)(N) )
//absolute value of number N
```

Lab

- You now have all the required skills to prepare a Transaction Processing System or DataBase System.
- Create clients, write to file.
- Offer menus to users:
- That offers the management of the data: add/update/delete/display.
- Further code to track a shopping cart will be required and the addition of a time stamp as a unique identifier.