Computer Programming

Dr. Deepak B Phatak
Dr. Supratik Chakraborty
Department of Computer Science and Engineering
IIT Bombay

Session: Creating a Binary File

Quick Recap

- We wrote a program to handle data in external files
  - We created a text file markdata.txt, using C++ functions
- We recall that a file on disk is like an array of bytes
  - Just as an array element can be accessed through an index value, one or more bytes can be directly accessed by giving the **position** of the starting byte
- There are file functions in C++ which can read or write a number of bytes at a specified position in the file

Overview

- In this session, we will study a program to create a binary file, in which fixed length records are written
- Later, we will see how these records can be directly accessed, read, and updated

Files, records, and fields

- Recall the format of the text file created (marksdata.txt)

10101 Anil 112 12.50

10102 Amit 111 15.00

10103 Shefali 112 17.00

- Each line contains a **record** of one student's information
- A record has 4 **fields or attributes**
  - Roll number, name, batch number, marks
- Each field, and thus each record has a fixed length (in bytes)

Fixed length records

- We know the record size (number of bytes in a record), say **S**
- If data for 10,000 students is written to a disk file, then the file will contain 10,000 x **S** bytes
- If we know which is the relative position **r**, of the record of a student in the file, then we can directly read the data for that student
  - **r*** (**S**-1) will be the starting byte position of the record
  - Next **S** bytes will contain the record itself

A record structure

IIT Bombay

- There is no need to store all values in a record in text format
  - We can directly store values in the internal format
- One good way is to define a structure for our record

```
struct studentinfo {
    int roll;
    char name[30];
    int batch;
    float marks;
}
```

Size and elements of a structure

- We define a structure variable s

  struct studentinfo s;

- Individual elements of s can now be accessed by

  s.roll, s.name, s.batch , and s.marks

- The size (in bytes) of a structure can be found by

  int rec_size; rec_size = sizeof(struct studentinfo)

Size and elements of a structure

- Suppose we define a structure variable s

  struct studentinfo s;

- Individual elements of s can now be accessed by

  s.roll, s.name, s.batch , and s.marks

- The size (in bytes) of a structure can be found by

  int rec_size; rec_size = sizeof(struct studentinfo)

- Most compilers will count the size of our record as 44 bytes
  - Elements need to be allocated at **word** boundary (divisible by 4)

Program logic for creating a database file

Open input text file, output binary file

Read one line from input text file, into four variables

While (**not** eof input file){

    assign values to elements of structure variable

    write the structure variable to output file

    Read next line from input text file

}

Close files

Program logic for creating a database file

Open input text file, output binary file

Read one line from input text file, into four variables

While (**not** eof input file){

    assign values to elements of structure variable

    write the structure variable to output file

    Read next line from input text file

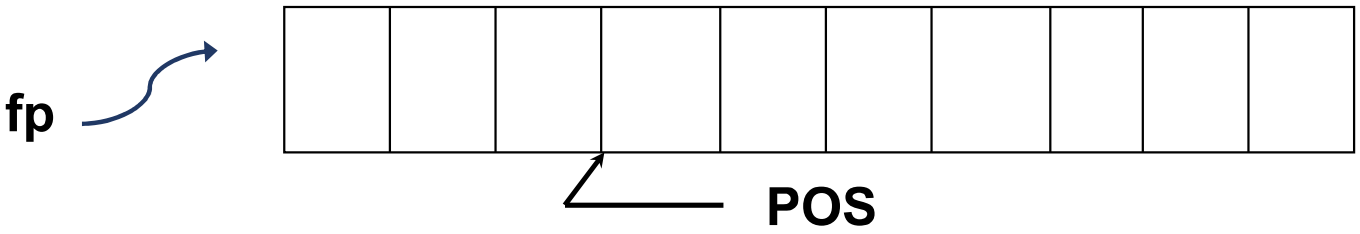
}

Close files

Organizing records in a direct access file

IIT Bombay

- We can directly access data in a binary file giving position



- Current position can be found using a function ftell()

  long POS; POS = ftell(fp);

- Reading/writing happens at this position, which automatically advances after each such operation

Program to create a binary file


```
/*Program create_student_db.cpp*/
/* For creating a binary file containing records of students*/
#include <iostream>
#include <cstring>
#include <cstdio>
using namespace std;
```

Program …


```
struct studentinfo{
  int roll;
  char name[30];
  int batch;
  float marks;
};
```

Program …

IIT Bombay

```
int main() {
 struct studentinfo s;
 int rec_size;
 rec_size = sizeof(struct studentinfo);
 cout << "Size of record is:  "<<rec_size<<endl;
 FILE *fp_input, *fp_output;
```

Program …

IIT Bombay

```
fp_input = fopen("markdata.txt", "r" );
if (fp_input == NULL){
   cout << "Could not open input file" << endl;
   return -1;
}
fp_output=fopen("studentdb","wb");
 if (fp_output == NULL){
   cout << "Could not open output file"<< endl;
   return -1;
 }
```

Program …

IIT Bombay

```
int r, b; char n[30]; float m;
// attribute values for a record
int count=0;
```

Program …

IIT Bombay

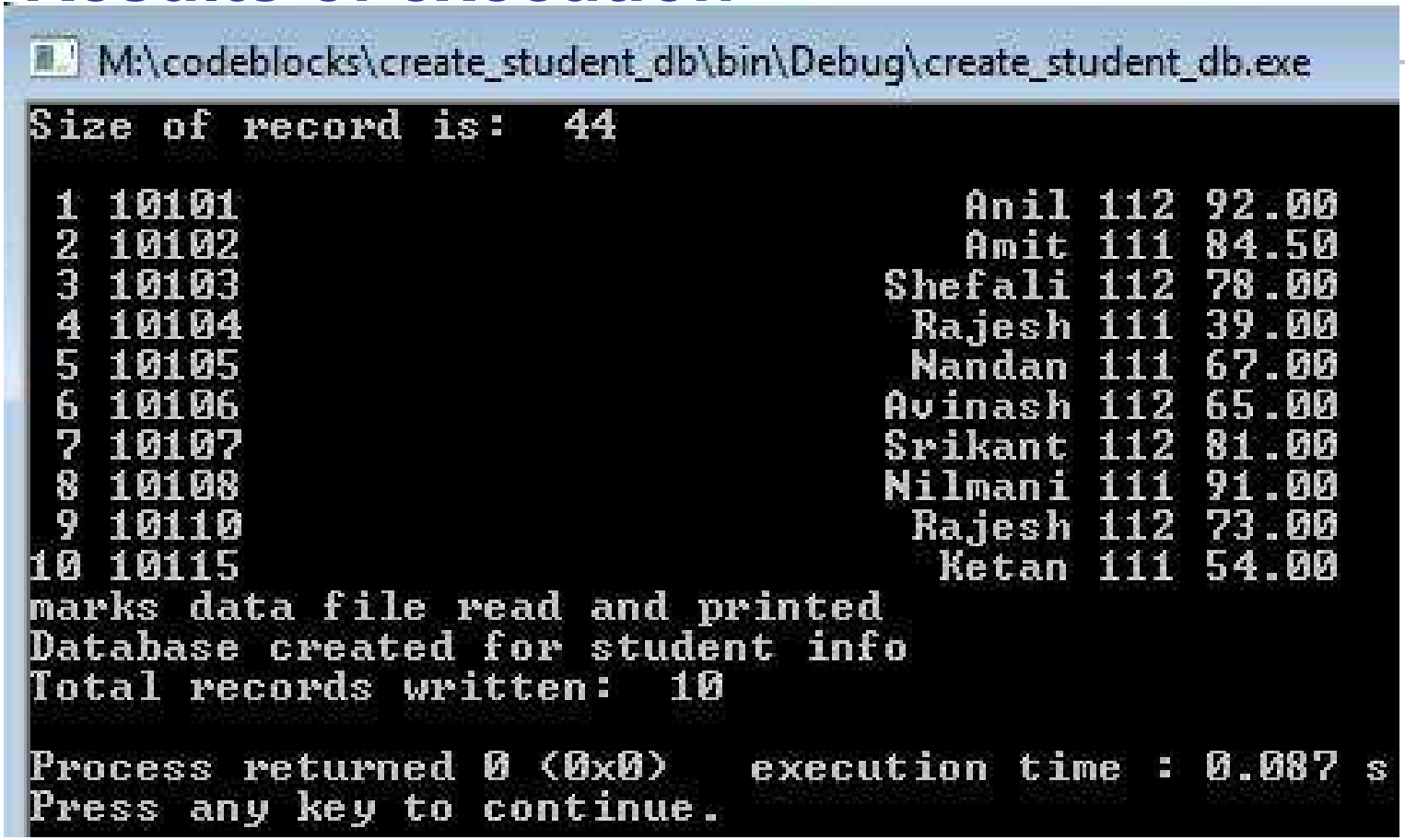
```
fscanf (fp_input, "%d %s %d %f",&r,n,&b,&m);
cout << endl;
while (!feof (fp_input)) {
   count++; s.roll=r, s.batch=b; s.marks=m; strcpy(s.name, n);
   fwrite(&s, rec_size, 1, fp_output);
   printf("%2d %5d %30s %3d %5.2f\n", count, s.roll, s.name,
                  s.batch, s.marks);
   fscanf (fp_input, "%d %s %d %f",&r,n,&b,&m);
}
```

Program …

IIT Bombay

```
  cout << "marks data file read and printed\n";
  cout << "Database created for student info\n";
  cout << "Total records written:  "<<count<<endl;
  fclose(fp_input);
  fclose(fp_output);
  return 0;
}
```

Results of execution



Summary

- We studied how to create a binary file, to write fixed length records containing students' data
- In the next session, we will see how to access and update the records of this file