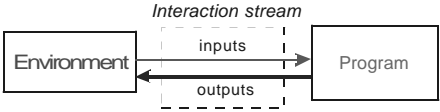


Topic: File input/output

- I. Streams
- II. Access methods
- III. C++ style
 - Input, output, random access
 - Stream classes: *ifstream*, *ofstream*
- IV. C style
 - The *FILE* data type
 - Opening files
 - Writing to, reading text files
 - Character input/output
 - Random access

David Keil 2/03 1

A stream is infinite



- Stored data occupies finite space
- A user-controlled input loop may never end
- I/O streams express *interactive* (non-algorithmic) computations

David Keil 2/03 2

The extractor and inserter form expressions

- The value of `(cin >> input1)` is `cin`, so it may be followed by `>> input2`
- The value of `(cout << input1)` is `cout`, so it may be followed by `<< " + " etc.`
- The `<<` (inserter) and `>>` (extractor) operators are *left_associative*

David Keil 2/03 3

File access methods

1. Sequential
 - Appropriate for batch processing
 - Kinds: (a) read; (b) write; (c) append
 - I/O file stream objects (C++)
 - The *FILE* data type (C)
2. Random
 - Appropriate for transaction processing
 - Index often used
 - Requires fixed-length records
 - In database terminology, a *record* is like a structure stored in a file

David Keil 2/03 4

File I/O in C++

- A file stream object is declared and used
- Standard classes defined in *fstream.h*: *ifstream*, *ofstream*, descended from *ios*
- Member functions: *open*, *close*, *eof*
- Extractor and inserter work as with console I/O
- Random-access member functions are *seekp* and *seekg*

David Keil 2/03 5

Reading 2 integers from a file (C++)

```

#include <iostream.h>
#include <fstream.h>
void main()
{
    int input1,input2;
    ifstream infile("ADDINFIL.TXT",ios::nocreate);
    if (!infile.good())
        cout << "File not found\n";
    else
    {
        cout << "Reading 2 integers from file"
             ADDINFIL.TXT." << endl;
        infile >> input1 >> input2;
        infile.close();
        int sum = input1 + input2;
        cout << input1 << " + " << input2
             << " = " << sum << endl;
    }
}
    
```

[addinfil.cpp] David Keil 2/03 6

Writing to a file

```
#include <fstream.h>

void main()
{
    cout << "Enter 2 integers you wish to add:
";
    int input1,input2,sum;
    cin >> input1 >> input2;
    sum = input1 + input2;
    ofstream outfile("FILEADD.OUT");
    outfile << input1 << " + " << input2
        << " = " << sum << endl;
    outfile.close();
}
```

[fileadd.cpp]

Sample I/O:

Enter 2 integers you wish to add: **4 9**
 [Contents of FILEADD.OUT after program runs:] **4 + 9 = 13**

2 ways to open a file

- Pass a file name to file stream constructor:
ofstream fout("testscor.txt");
- Declare a file stream object and then use it to open a text file:
ofstream fout;
fout.open("testscor.txt");
- Both the above examples declare a file stream object, *fout*, and use it to open a text file, *testscor.txt*
- Both techniques work with either *ofstream* or *ifstream*, with appropriate modifications of type

David Keil 2/03 8

Reading a file to the end

```
void main()
{
    char fil_nm[80];
    cout << "File name: ";
    cin >> fil_nm;
    ifstream infile(fil_nm,ios::nocreate);
    if (!infile)
        cerr << "File " << fil_nm << " not found";
    else {
        char line[80];
        while (!infile.eof()) {
            infile.getline(line,79);
            cout << line << endl;
        }
        infile.close();
    }
}
```

Read disk until file pointer reaches end of file

[echofile.cpp]

David Keil 2/03 9

Testing the file-input operation

```
void main()
{
    int count;
    float score;
    ifstream infile("myscore.txt",ios::nocreate);
    if (!infile.bad())
    {
        count=0;
        while(!infile.eof())
            if (infile >> score)
            {
                cout << score << " ";
                ++count;
            }
        infile.close();
        cout << "count = " << count << endl;
    }
}
```

Output using file with contents
 89
 93
 80[newline]
89 93 80 count = 3

[readscor.cpp]

David Keil 2/03 10

Pass file stream objects by reference

```
void save(ofstream& fout,char* message);

void main()
{
    ofstream fout("savemesg.txt");
    save(fout,"Happy New Year");
    fout.close();
}


void save(ofstream& fout,char* message)
// Writes <message> to file <fout>.
{
    fout << message << endl;
}
```

[savemesg.cpp]

Written to SAVEMESG.TXT: **Happy New Year**

David Keil 2/03 11

Problem: transaction processing

- *Example:* a million-customer credit database 
- System must update your file record when you make a purchase or payment
- System must tell vendor your credit status *in real time*
- Reading or writing a million-record sequential file would take too long

David Keil 2/03 12

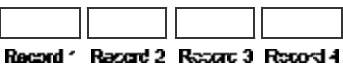
Solution: random access

- Avoid reading or writing entire file
- Instead, read or write only one data record at a time
- Use random access to peek at or to overwrite existing data in the middle of a file
- If data is in chronological or other order, maintain an index that tells where to find a customer's record

David Keil 2/03 13

Random access uses fixed-length items

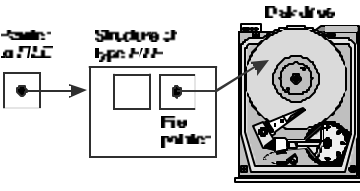
- Each record (employee, course-registration, etc.) is of uniform size
- Program calculates disk location of a record, as $(index - 1) \times record\ size$, offset from beginning of file



David Keil 2/03 14

How random access works

- A pointer to a *FILE* structure is programmer's handle to use for file access
- A file pointer in the *FILE* structure stores the disk location of the next access



David Keil 2/03 15

Opening a random file in C++

```
class rand_files
{
public:
    rand_files(char nm[]) {
        strcpy(name,nm);
        file.open(name,ios::in|ios::out); }
    ~rand_files() { file.close(); }
    void create();
    void sort();
    void display();
private:
    char name[80];
    fstream file;
    int get_num(int index);
    void put_num(int num,int index);
    void swap(int index_1,int index_2);
};
```

- The C++ library *fstream.h* defines class *fstream*
- Function *open* takes parameter *ios::in* / *ios::out*

David Keil 2/03 16

Using some random-access I/O functions

```
void main()
{
    rand_rand_files rf("randrand.out");
    int a = rf.get_num(0),
        b = rf.get_num(1);
    rf.put_num(a,1);
    rf.put_num(b,0);
}
```

Read values a, b from locations 0,1

Write those values back at locations 1,0

Equivalent:

```
int a = rf.get_num(0);
rf.put_num(rf.get_num(1),0);
rf.put_num(a,1);
```

David Keil 2/03 17

Random read/write in C++

```
const int MAX_DIGITS = 2;
int rand_files::get_num(int index)
// Reads, returns int value at location <index>.
{
    file.seekg((MAX_DIGITS+1) * index);
    int input;
    file >> input;
    return input;
}

void rand_files::put_num(int num,int index)
// Writes <num> to location <index>.
{
    file.seekp((MAX_DIGITS+1) * index);
    file << num;
}
```

- The C++ library *fstream.h* provides *fstream* member functions *seekg* for random read, *seekp* for random write

David Keil 2/03 18

Finding size of a file

```

void main()
{
    cout << "File name: ";
    char file_name[40];
    cin >> file_name;
    ifstream infile(file_name, ios::nocreate);
    if (infile)
    {
        infile.seekg(0, ios::end);
        cout << "Size of " << file_name << ": "
            << infile.tellg() << endl;
        infile.close();
    }
    else
        cout << "File " << file_name
            << " not found" << endl;
}
    
```

Prompts for name of file, displays size.

David Keil 2/03 19

Summary of C++ file I/O

- C++ library *fstream.h* defines *stream classes*, whereas C library *stdio.h* defines *FILE* type and global functions
- For input file, use *ifstream*; for output use *ofstream*; for random access use *fstream*
- A file stream object may replace *cin* or *cout* without other changes
- Member functions: *open*, *close*, *eof*, *seekg*, *seekp*

David Keil 2/03 20

Example of file output in C

```

#include <stdio.h>
void main(void)
{
    int i;
    FILE *outfile = fopen("fcount.txt", "w");
    for (i=1; i <= 5; i++)
        fprintf(outfile, "%d", i);
    fclose(outfile);
}
    
```

Programmer chooses file name
Open to write
Pointer to FILE

Contents of fcount.txt after execution: 1 2 3 4 5

David Keil 2/03 21

Facts about file output in C

- Formatted output accepts field widths, left/right spec, etc.
- Opening a file to write erases all old data
- It is wise to get user permission to overwrite existing data
- To append to a file, replace "w" with "a":
FILE* fv = fopen("x.txt", "a");
- In DOS, printer port's file name is "LPT1":
FILE* printer = fopen("LPT1", "w");

David Keil 2/03 22

File input with fscanf

```

void main(void)
{
    const char FILE_NAME[] = "read_int.txt";
    int input;
    FILE *infile = fopen(FILE_NAME, "r");
    if (infile == NULL)
        printf("%s not found.\n", FILE_NAME);
    else
    {
        fscanf(infile, "%i", &input);
        printf("Found in file: %i\n", input);
        fclose(infile);
    }
}
    
```

Open file
Standard error output
Read integer from disk
Display it

- Like the extractor <<, *fscanf* reads up to a space

David Keil 2/03 23

Text file input with loop

```

#include <stdio.h>
#define LINE_SZ 80
void main(void)
{
    const char FILE_NAME[LINE_SZ] = "fgreet.txt";
    char line[LINE_SZ];
    FILE *infile = fopen(FILE_NAME, "r");
    if (!infile)
        fprintf(stderr, "File %s not found.\n", FILE_NAME);
    else
    {
        while (fgets(line, LINE_SZ-1, infile) != NULL)
            printf("%s", line);
        fclose(infile);
    }
}
    
```

Open file
Read from file

*Output:
Hello there!*

David Keil 2/03 24

Using *feof*, *fgetc*, *putchar*

```
void main(void)
{
    [fgreet2.c]
    const char FILE_NAME[40] = "fgreet.txt";
    char ch;
    FILE *infile = fopen(FILE_NAME, "r");

    [...]
    while (feof(infile) == false)
    {
        ch = fgetc(infile); // Tells whether at end of file
        // Reads one char from disk
        putchar(ch);
    }
    fclose(infile); // Displays char parameter

    fputc(fp, ch) writes char ch to file opened as fp.
}
```

David Keil 2/03 25

Text file input

- Standard type: *FILE*
- Open file with *fopen* as for output, but use "r" instead of "w"
- Check for file not found
- **fscanf** (*pointer to FILE*, *format string*, *input variable addresses*);
- *gets()*, *fgets()*, *feof()*

David Keil 2/03 26

Random file I/O in C

- A file is opened in *read/write* mode:
FILE* pf;
pf = fopen(FILE_NAME, "rb+");
- Use ordinary file read/write functions after positioning file pointer
- Any text file may be read from or written to with random access
- Random means arbitrary, not chancy

David Keil 2/03 27

Read or write 1 *char* by random access

```
char read_char(FILE* pf, int index)
{
    char input;
    fseek(pf, index, SEEK_SET);
    input = fgetc(pf); // Reads, returns indexth
    // character in file
    return input;
}

void write_char(FILE* pf, int index, char ch)
{
    fseek(pf, index, SEEK_SET);
    fputc(ch, pf); // Writes ch to indexth
    // location of file.
}
```

David Keil 2/03 28

Using random access to swap data in file

```
void swap(FILE* pf, int index_1, int index_2)
{
    /* Swaps values at locations index_1, index_2 */
    char old_1 = read_char(pf, index_1); // Read values in file into variables
    char old_2 = read_char(pf, index_2);
    write_char(pf, old_2, index_1);
    write_char(pf, old_1, index_2);
}
// Store old value from index_1 at index_2:
// Copy from index_2 to index_1:
[from filesort.c]
```

David Keil 2/03 29

How to position the file pointer

- Function *fseek* positions file pointer prior to file read or write
- Note: a file pointer is not a pointer to *FILE* !
- Second parameter to *fseek* specifies offset in bytes
- As 3rd parameter, constant *SEEK_SET* offsets seek from the beginning of file:
fseek(pf, index, SEEK_SET);
- To offset from end of file use *SEEK_END*

David Keil 2/03 30

How large is our file?

```
FILE* pf = fopen(FILE_NAME,"rb+");
fseek(pf, 0, SEEK_END);
printf("%s is %i bytes long\n",
FILE_NAME,ftell(pf));
```

[see *filesort.c*]

- To find file length, position file pointer at end using *fseek*, *SEEK_END*
- Then call *ftell*, which returns current offset from beginning of file

David Keil 2/03 31

Summary of text file I/O in C

- Uses standard tools declared in *stdio.h*
- **FILE *pf;**
declares file-handle pointer variable; programmer chooses its name (e.g., *pf*)
- **pf = fopen("X.TXT", "w");**
opens file to write, "r" = read, "a" = append
- Standard functions: *fprintf*, *fscanf*, *fgets*, *feof*
- Good practice: check for file-not-found before reading or writing
- Random access permits efficient transaction processing

David Keil 2/03 32

Problems:

1. Write a program that reads a file of *int* into an array
2. Write a program that prompts for an ID number, searches an unsorted text file for that ID, and tells where the ID was found.

Example:

[IDS.TXT:]

22 41 82 76 55 147

David Keil 2/03 33

Functions, defined types, and files

```
Function call ----- int ave = average(5,2);
Function prototype ----- int average(int, int);
Function definition ----- int average(int a,int b)
                           { return (a+b) / 2; }
Enumerated-type ----- enum divisions {Day,Evening};
definition
Enumerated-type ----- divisions div = Day;
variable declaration
Structure type definition ----- struct courses
                               { int ID; divisions div; };
Structure variable ----- courses CSII = {252,Day};
initialization
Function prototype with ----- int course_ID(courses c);
structure parameter
Statements to open a file ----- FILE* pf = fopen("x.txt","r"); (C)
                               ifstream fin("emp.txt"); (C++)
```

David Keil 2/03 34

Problem

- Your customer brings a text file that contains the names of people who've entered a contest to guess the number of beans in a jar, and their guesses, followed by the correct number of beans in the jar.
- Sample file:
Bob 600
Sue 500
400
- Write a program that reads the data in this text file and displays the name of the person with the best guess.
- Sample output for the above file: **sue**

David Keil 2/03 35