Unit -6 Hashing and Files

Unit -6 : Content

- **Hashing:**
- Hash tables and scattered tables:
  - Basic concepts,
  - hash function,
  - characteristics of good hash function,
  - Different key-to-address transformations techniques,
  - synonyms or collisions,
- collision resolution techniques-
  - linear probing,
  - quadratic probing,
  - rehashing,
  - chaining with and without replacement.
- **File:**
- Concept of File
- File types and file organization :
  - sequential,
  - index sequential
  - and Direct Access
- Comparison of different file organizations.

Learning Objectives

- File as a data type/Data structure
- C++ I/O streams.
- Reading and writing sequential files.
- Reading and writing random access files.
- Symbol table notation and its usage
- Symbol table implements ion using Hash table
- Issues with ahs table and handling it with various techniques

Course objective ::

C19203.6	Students will be able to describe Concept and terminologies Notion of Symbol Table, OBST, AVL Trees, Heap data structure, of hashing using Hash Table and scattered tables, describe and select hash function by applying open and close addressing techniques for collision resolution such as linear probing, quadratic probing, rehashing, chaining with and without replacement, perform heap sort.
C19203.7	Students will be able to explain concept of File, File types and file organization (sequential, index sequential and Direct Access),compare sequential, index sequential and Direct Access file organization .

File Organization and Storage Structures

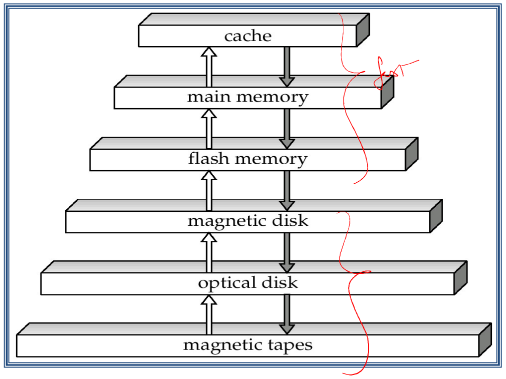
o Storage of data

– Primary Storage = Main Memory
  - Fast
  - Volatile
  - Expensive

– Secondary Storage = Files in disks or tapes
  - – Non-Volatile
  - – Non Expensive
  - – Slow

Secondary Storage is preferred for storing data

Storage Hierarchy



©Silberschatz, Korth and Sudarshan
Modifications & additions by S Bird, J Bryson

What is a File?

Single Unit
ABC.txt
file
INode

- A *file* is a collection of related data that a computers treats as a single unit.
- Computers store files to secondary storage so that the contents of files remain intact when a computer shuts down.

HDD
R/W – ABC.txt
RAM
Stream

- When a computer reads a file, it copies the file from the storage device to memory;
- when it writes to a file, it transfers data from memory to the storage device.

file
RAM
HDD
stream
load

Types of files :

- Types of file base on usage
- Data files : data type
  - A **data file** is any file that contains information, but not code;
  - it is only meant to be read or viewed and not executed.
  - For example:
    - web page
    - a letter you write in a word processor
    - and a text file are all considered data files.

text
Codes ASCII

- Code files : — Specialized program – C++ Compiler / Interpreters/
  - Contain the code , executed to see the result after execution . VisStudio.
  - Example :
    - .java, .cpp,.c etc files used to store your codes

Why file (data files)

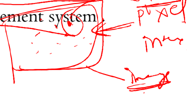
- Files are places where data can be stored permanently.
- Some programs expect the same set of data to be fed as input every time it is run.
  - Cumbersome.
  - Better if the data are kept in a file, and the program reads from the file.
- Programs generating large volumes of output.
  - Difficult to view on the screen.
  - Better to store them in a file for later viewing/ processing

File
load
loads records
BIOS
Operators
OS
O/P
200 pages

Applications of files

- Database applications :
  - ticket reservation system
  - hotel management system
  - online examinations
  - student admission process etc.
- Software System applications
  - -- operating system,
  - -- language processors
  - -- graphics systems etc.
- To Build a vital software system of the future,
  - We need to be able to structure and manipulate information effectively and efficiently
  - To fulfill this we need a information management system
  - File organization is one of its component.

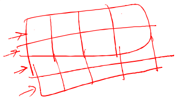
files
Image
File – Hot read
pixel Value
0 – 255
Image
RGB
pixel



Data files

- What is a file?
  - Collection of records
  - Record is a collection of fields.
- Basic operations
  - Insertion
  - Deletion
  - Search/retrieval
  - Update

text

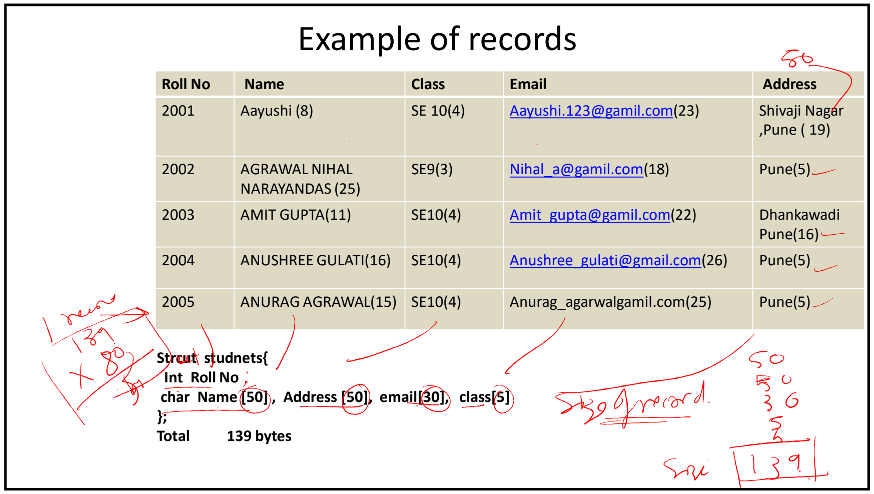


Basic Concepts

- Information are stored in data files
- Each file is a sequence of records
- Each record consists of one or more fields

Sno	Lname	Position	NIN	Bno
SL21	White	Manager	WK440211B	B5
SG37	Beech	Snr Asst	WL432514C	B3
SG14	Ford	Deputy	WL220658D	B3

- A file is a collection on information, usually stored on a computer's disk.
- Information can be saved to files and then later reused.



Example of records

Roll No	Name	Class	Email	Address
2001	Aayushi (8)	SE 10(4)	Aayushi.123@gamil.com(23)	Shivaji Nagar ,Pune (19)
2002	AGRAWAL NIHAL NARAYANDAS (25)	SE9(3)	Nihal_a@gamil.com(18)	Pune(5)
2003	AMIT GUPTA(11)	SE10(4)	Amit_gupta@gamil.com(22)	Dhankawadi Pune(16)
2004	ANUSHREE GULATI(16)	SE10(4)	Anushree_gulati@gmail.com(26)	Pune(5)
2005	ANURAG AGRAWAL(15)	SE10(4)	Anurag_agarwalgamil.com(25)	Pune(5)

```
Strcut studnets{
 Int Roll No
 char Name [50], Address [50], email[30], class[5]
};
Total      139 bytes
```

50

1 record

139

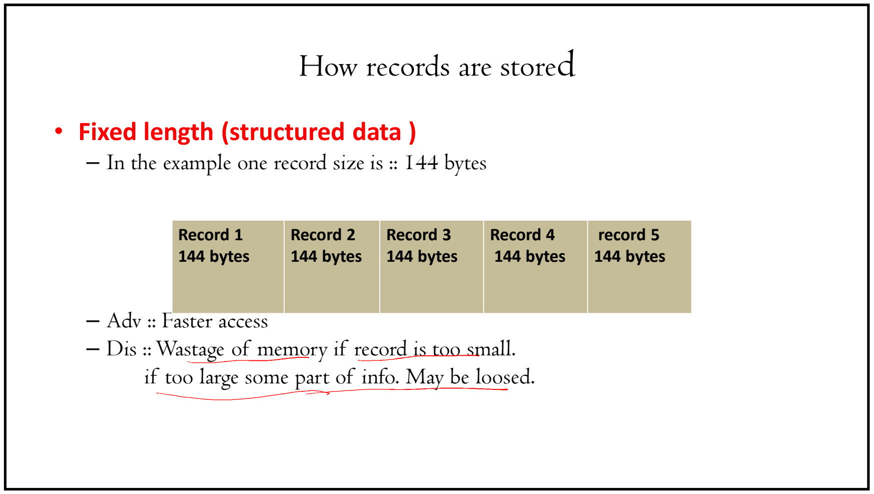
x 80

139 of record.

50
50
30
5

Size 139

13



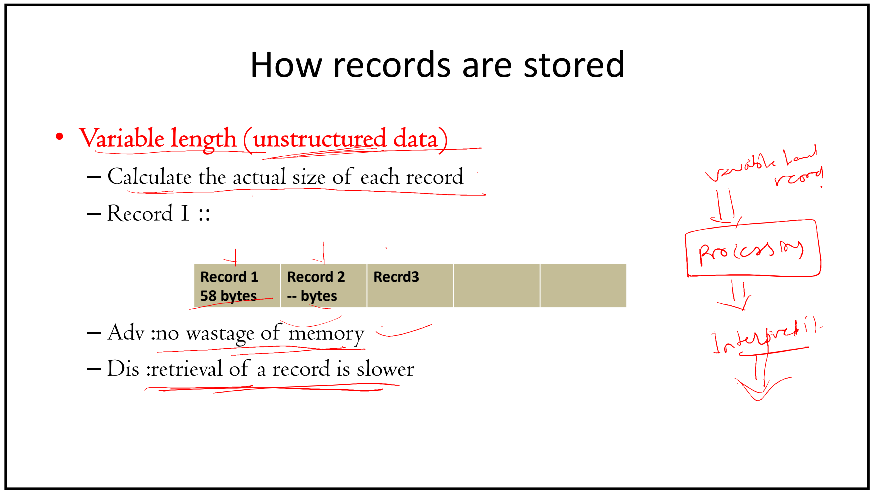
How records are stored

- **Fixed length (structured data)**
  - In the example one record size is :: 144 bytes

Record 1	Record 2	Record 3	Record 4	record 5
144 bytes	144 bytes	144 bytes	144 bytes	144 bytes

  - Adv :: Faster access
  - Dis :: Wastage of memory if record is too small.
    if too large some part of info. May be loosed.

14



How records are stored

- Variable length (unstructured data)
  - Calculate the actual size of each record
  - Record I ::

Record 1	Record 2	Recrd3		
58 bytes	-- bytes			

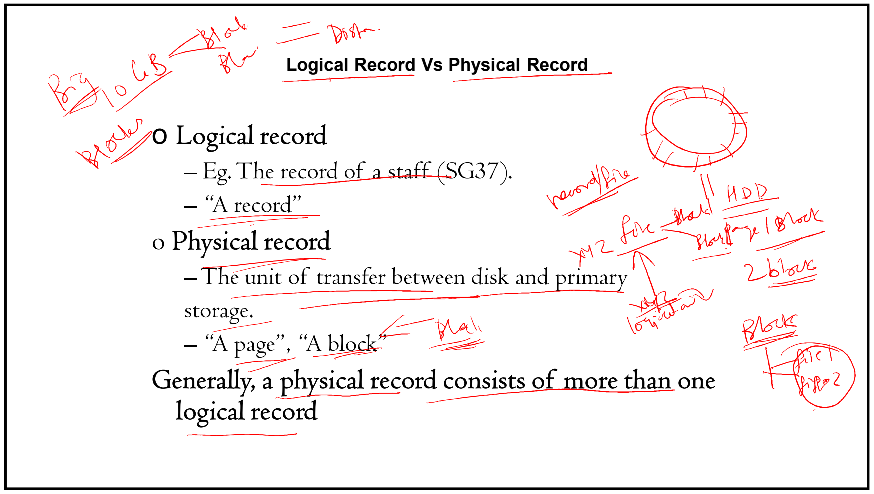
  - Adv :no wastage of memory
  - Dis :retrieval of a record is slower

variable len record

Processing

Interpretation

15



Logical Record Vs Physical Record

Big 10 GB

Block

Disk

Blocks

- Logical record
  - Eg. The record of a staff (SG37).
  - "A record"
- Physical record
  - The unit of transfer between disk and primary storage.
  - "A page", "A block"

Generally, a physical record consists of more than one logical record

record/file

HDD

xyz file

Block/Page 1 Block

2 block

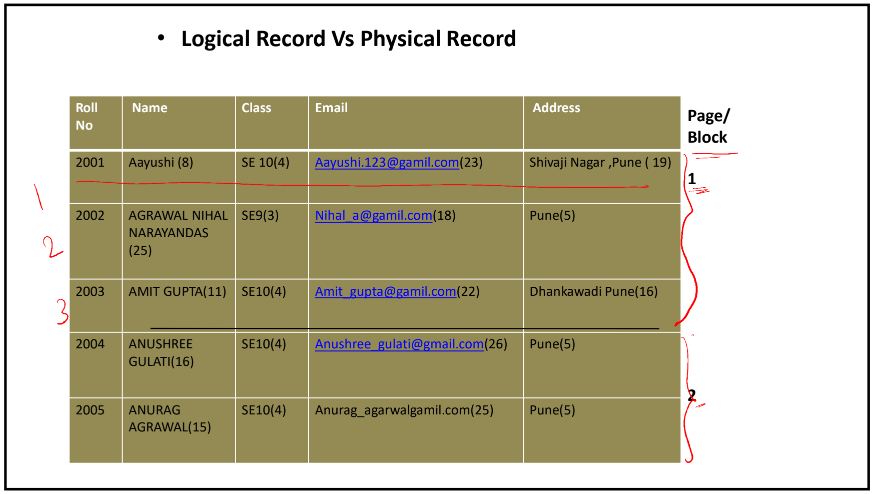
logical

Block

file1

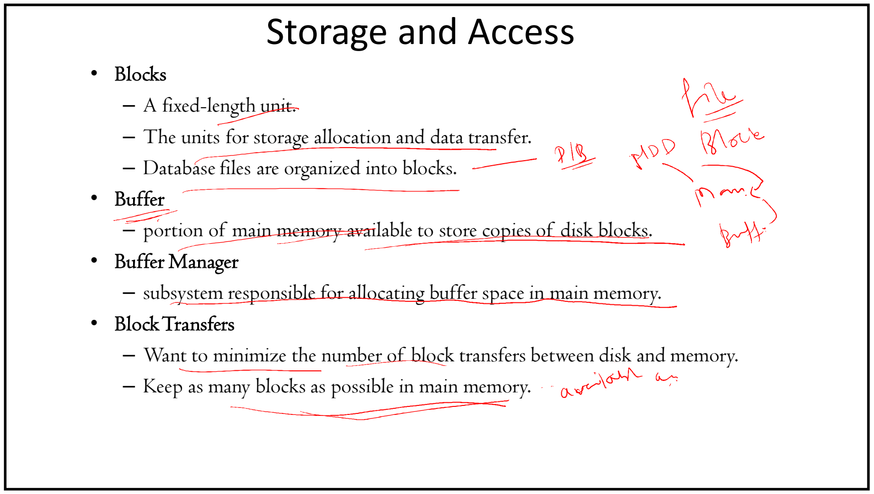
file2

16



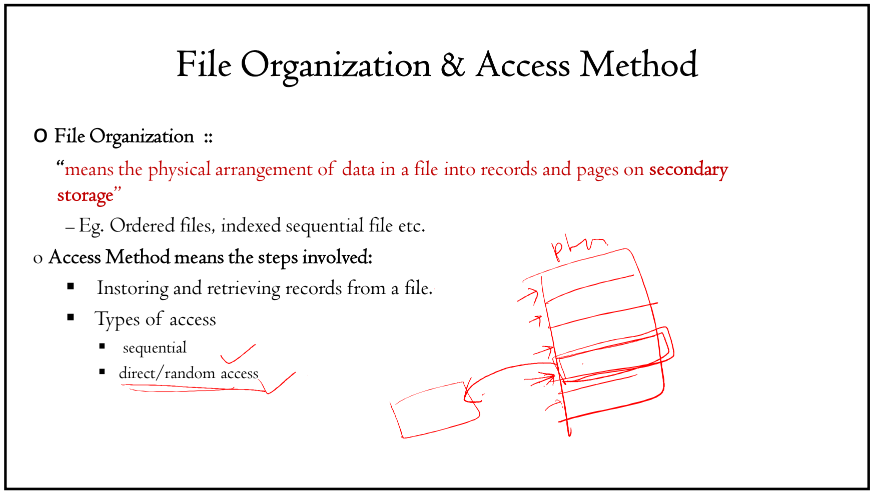
- **Logical Record Vs Physical Record**

Roll No	Name	Class	Email	Address	Page/ Block
2001	Aayushi (8)	SE 10(4)	Aayushi.123@gamil.com(23)	Shivaji Nagar ,Pune (19)	1
2002	AGRAWAL NIHAL NARAYANDAS (25)	SE9(3)	Nihal_a@gamil.com(18)	Pune(5)	
2003	AMIT GUPTA(11)	SE10(4)	Amit_gupta@gamil.com(22)	Dhankawadi Pune(16)	
2004	ANUSHREE GULATI(16)	SE10(4)	Anushree_gulati@gmail.com(26)	Pune(5)	2
2005	ANURAG AGRAWAL(15)	SE10(4)	Anurag_agarwalgamil.com(25)	Pune(5)	



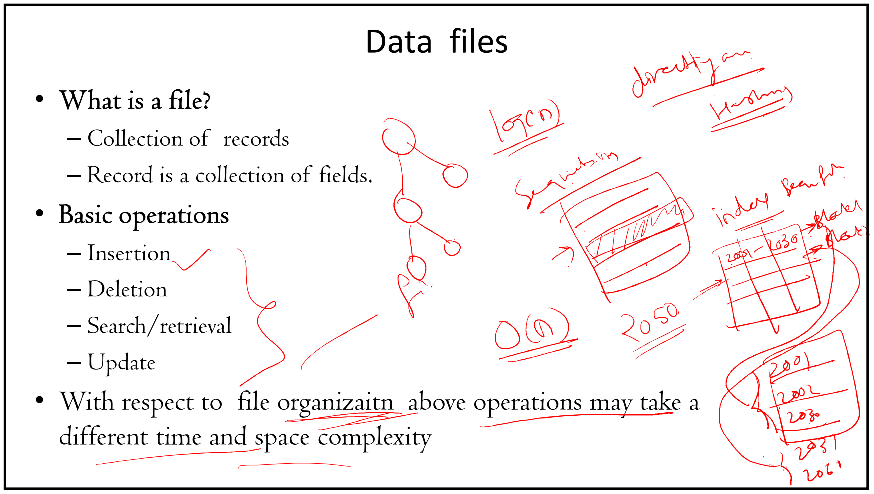
Storage and Access

- Blocks
  - A fixed-length unit.
  - The units for storage allocation and data transfer.
  - Database files are organized into blocks.
- Buffer
  - portion of main memory available to store copies of disk blocks.
- Buffer Manager
  - subsystem responsible for allocating buffer space in main memory.
- Block Transfers
  - Want to minimize the number of block transfers between disk and memory.
  - Keep as many blocks as possible in main memory.



File Organization & Access Method

- File Organization ::
  "means the physical arrangement of data in a file into records and pages on **secondary storage**"
  - Eg. Ordered files, indexed sequential file etc.
- Access Method means the steps involved:
  - Instoring and retrieving records from a file.
  - Types of access
    - sequential
    - direct/random access



Data files

- What is a file?
  - Collection of records
  - Record is a collection of fields.
- Basic operations
  - Insertion
  - Deletion
  - Search/retrieval
  - Update
- With respect to file organizaitn above operations may take a different time and space complexity

log(n)
directly an
Hashing
Sequential
index Search
2001-2030
Block
Block
O(n)
2050
2001
2002
2030
2031
2061

Logical vs. Physical deletion

- How it is done in the array?
- Logical deletion : Delete a record with roll no 2002

140 bytes
-20 bytes
120 byte

Data file before deletion File1.txt

Roll No	Name	Class
2001	Aayushi (8)	SE 10(4)
2002	AGRAWAL NIHAL NARAYANDAS (25)	SE9(3)
2003	AMIT GUPTA(11)	SE10(4)
2004	ANUSHREE GULATI(16)	SE10(4)
2005	ANURAG AGRAWAL(15)	SE10(4)

Data file after deletion : File1.txt

Roll No	Name	Class
2001	Aayushi (8)	SE 10(4)
-1	%%%%%	%%%%
2003	AMIT GUPTA(11)	SE10(4)
2004	ANUSHREE GULATI(16)	SE10(4)
2005	ANURAG AGRAWAL(15)	SE10(4)

move the records 1 posi up

Search record and rewrite with updated values.

Logical vs. Physical deletion

- How it is done in the array?
- Physical : Delete a record with roll no 2002

Data file before deletion File1.txt

Roll No	Name	Class
2001	Aayushi	SE 10
2002	AGRAWAL NIHAL	SE9
2003	AMIT GUPTA	SE10
2004	ANUSHREE GULAT	SE10
2005	ANURAG AGRAWAL	SE10

Data file during deletion : temp.txt

Roll No	Name	Class
2001	Aayushi	SE 10
2003	AMIT GUPTA	SE10
2004	ANUSHREE GULAT	SE10
2005	ANURAG AGRAWAL	SE10

Data file after deletion : File1.txt

Roll No	Name	Class
2001	Aayushi	SE 10
2003	AMIT GUPTA	SE10
2004	ANUSHREE GULAT	SE10
2005	ANURAG AGRAWAL	SE10

file1.txt
2001
2003
2004
2005

Sequential

Search record
Create a temporary file
Copy all the record excluding the record which is deleted from fileI .c to temporary file
Delete original file(fileI.txt)
Rename the new file as original I)

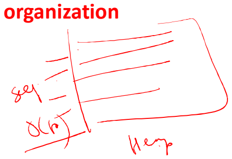
Physical deletion

Example of records

Strcut studnets{
Int Roll No
char Name [50] , Address [50], email[30], class[5]
};
Total 139 bytes

Types of file organization

- Pile/heap /unordered file organization::
  - Simplest one
  - Non keyed sequential files
  - records are not ordered.



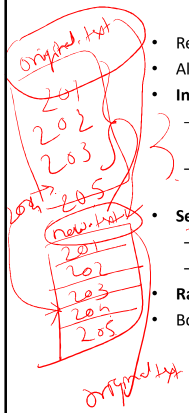
Operations on heap:

- Quick insertion (no particular ordering)
  - When a new record is created, it is put in the last on the page of the file if there is sufficient space. Otherwise
  - a new page is added to the file
- Slow retrieval (only allow linear search)
  - reading pages from the file until a required record is found.
- Logical or Physical Deletion (logical is preferred)
  - To delete a record, the record is marked as deleted.
  - Space is reclaimed during periodical reorganization.

rec

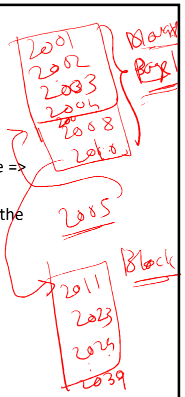
25

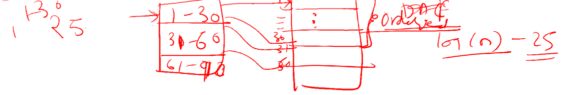
Ordered /ordered sequential files



- Records are sorted on field(s) => Key
- Also called as Ordered sequential files.
- **Inserting a record**
  - If the appropriate page is full, may have to reorganize the whole file => Time consuming
  - Solution: use a temporary unsorted file (transaction file). Merge to the sorted file periodically.
- **Search/Retrieval :**
  - Sequential as well as binary search is possible.
  - Faster access to a record.
- **Rarely used unless come with an index => Indexed Sequential File**
- Both **Heap Files** and **Ordered Files** are also called **Sequential Files.**

log n





26

Direct Access files

address
ph — new

- Direct Files are also called Hash Files or Random Files
  - ➢ No need to write records sequentially
  - ➢ Use a hash function to calculate the number of the page (bucket) which a record should be located
  - ➢ Eg., use the division-remainder calculation method that,

(1) Unit — Base Address + 5(sized record)

2001+5×10

2010

(1)

O(n) — n record

Hash function

Bucket

bucket_no = Record_key mod 10 Unit

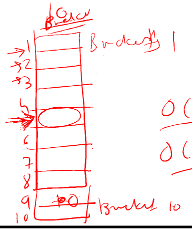
① Repeat ≤ Strong

15 mod 10 = 5

Bucket

5 mod 10 = 5

O(n) O(log n)



2 ≈ 1

O(1)

O(1) < O(n)

O(1) < O(log)

27

File format

- Two types
- Text file
  - It is visible file we can see the content on monitor
  - Eg : any text document, c program file, note pad fiel etc.
- Binary file :

  A byte file. (different storage size for Integer, float , other data type)

  Non visible content ,cant see on monitor

  eg .obj, .exe files

28

Binary Vs. Text file

- How we will represent : 234 in text and binary file ?
- Text file :

2	3	4

|--1byte--|---1 byte--|--1byte----| = 3 byte

Binary file: consider it as an integer no:

|--- 2 bytes-------|

(Int/char float. Number for Binary file

32767

char seqn for text)

- 32767 represented as a text file:

3	2	7	6	7
00110011	00110010	00110111	00110110	00110111

- Why?

00110011	00110010	00110111	00110110	00110111
51 ASCII	50	55	54	55
3	2	7	6	7

Binary Decimal Character

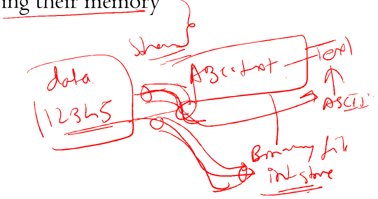
29

Text Streams & Binary Streams

- *Text streams* consist of sequential characters divided into lines. Each line terminates with the newline character (\n).

ABCDE \n
FGHIJK \n
LMM

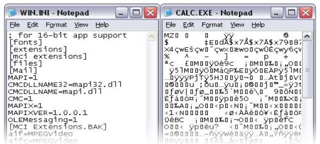
- *Binary streams* consist of data values such as integers, floats or complex data types, "using their memory representation."



30

Binary Vs .Text file

- Advantages & Disadvantages
- Text files contains text usually in ASCII everything else is Binary it is 1234 vs "1234"
- In binary files, the binary representation of values is written to the file
  - The integer '4929067' which takes 4 bytes in memory will also take 4 bytes in the file
- In case of text file, each value is written as a series of characters (ASCII or Unicode)
  - The integer '4929067' will be written as text and will take 7 bytes in ASCII encoding and 14 (7 x 2) bytes in Unicode encoding
- Binary files are more efficient for reading and writing of data for machines while text files are more human readable.
- Binary representations are much more space-efficient.



```
; for 16-bit app support
[fonts]
[extensions]
[mci extensions]
[files]
[Mail]
MAPI=1
CMCDLLNAME32=mapi32.dll
CMCDLLNAME=mapi.dll
CMC=1
MAPIX=1
MAPIXVER=1.0.0.1
OLEMessaging=1
[MCI Extensions.BAK]
aif=MPEGVideo
aifc=MPEGVideo
```

```
MZ    ÿÿ         @
$        ‡E dÅ$x7Å$x7Å$x79 87
X4çwEšçw ˜çwx æwo çwÕÉçwy6çw
%    ^    -    ]    =    p    +
ªc  £ M  ÿ0è9c   ¡ M  ‰ ¡„O
 ÿ5]M  ÿÖ MàQP‰E ÿÓ EÀPÿ5]M
… ÿÿÿPjTÿ5HJ  ÿ ¬    …Àt j V
 @   u ;ðu …ÿu ¡ @  j ™_÷ÿJt
 føV| fø_  ‰5`M  é\    9 ÔN
Èƒá ó¤¡˜M  ÿp è5O   ¡˜M  ‰× ¡
  ‰A ¡„O  ‹p ‹N ¡˜M  ‹x
‹1‹N           ‹ø‹ÀÀé Ó¥‹Èƒá Ó¤
0èBC   ¡ M  ‰ ¡¬O  ‹  ÿp èfC
O  ‹  ÿp èu?   ‹ ˜M  ‰ ¡„O  ‹
```

- Reason for using binary formats is that they can map directly onto the internal representation of the data that the program is using.
- If program uses a complex data structure that's full of integer, float, and byte values, when I read it in from a text file I have to parse every character of text and interpret it, spending processing time (and writing logic) to figure how to convert the 88 bits in "2983102931," into the 32-bit integer that the text represents.
- If you read it from a binary file, the program just copies those 32 bits from the file into memory.

Int → int var a

float → float.

31

Types of file organization

- **Pile/heap /unordered file organization::**
  - Simplest one
  - Non keyed sequential files
  - records are not ordered.

Operations on heap:

- **Quick insertion (no particular ordering)**
  - When a new record is created, it is put in the last

    on the page of the file if there is sufficient space. Otherwise
  - a new page is added to the file
- **Slow retrieval (only allow linear search)**
  - reading pages from the file until a required record is found.
- **Logical or Physical Deletion (logical is preferred)**
  - To delete a record, the record is marked as deleted.
  - Space is reclaimed during periodical reorganization.

32

Ordered /ordered sequential files

- Records are sorted on field(s) => Key
- Also called as Ordered sequential files.
- **Inserting a record**
  - If the appropriate page is full, may have to reorganize the whole file => Time consuming
  - Solution: use a temporary unsorted file (transaction file). Merge to the sorted file periodically.
- **Search/Retrieval :**
  - Sequential as well as binary search is possible.
  - Faster access to a record.
- **Rarely used unless come with an index => Indexed Sequential File**
- Both **Heap Files** and **Ordered Files** are also called **Sequential Files.**

Direct Access files

- **Direct Files are also called Hash Files or Random Files**
  - **No need to write records sequentially**
  - **Use a hash function to calculate the number of the page (bucket) which a record should be located**
  - **Eg., use the division-remainder calculation method** that,

bucket_no = Record_key mod 10

Direct Access File

	Bucket
2010	0
2001	1
2002	2
	3
	4
	5
	....
....	...

Problem: If a new 2011 is created, which bucket to go if existing bucket is already full?

o **Collision Management :** Open addressing, Unchained overflow, Chained overflow, Multiple hashing

Direct Files

Open Addressing

o **Upon a collision, the system performs a linear search to find the first available slot.**

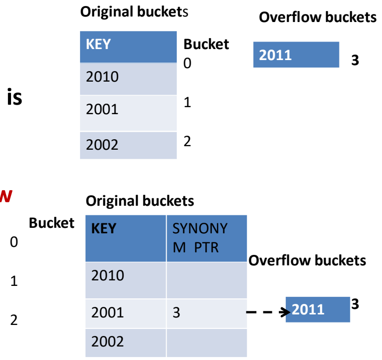
o **When last bucket has been searched, starts from the first bucket.**

O 11 **will be inserted to: Bucket 4**

	Bucket
2010	0
2001	1
2002	2
2011	3
	4
	5
	....
....	...

Direct Files

- **Overflow Buckets**
- o **An overflow area is maintained for collisions.**
- **Unchained overflow buckets**
- Chained overflow buckets



Original buckets

KEY	Bucket
2010	0
2001	1
2002	2

Overflow buckets

2011	3

Original buckets

Bucket	KEY	SYNONYM PTR
0	2010	
1	2001	3
2	2002	

Overflow buckets

2011	3

Direct Access

Multiple Hashing

o **Upon collision, apply a second hashing function to produce a new hash address in an overflow area.**

Direct Files

- **Insertion / deletion/Searching**
- Very fast
- Ideally O(1)

- **Limitation (of Hashing)**

Inappropriate for some retrievals:

– based on pattern matching

eg. Find all students with ID like 98xxxxxx.

– Involving ranges of values

eg. Find all students from 50100000 to 50199999.

– Based on a field other than the hash field

Indexes

- **Index: A data structure that allows particular records in a file to be located more quickly**

~ Index in a book

- **An index can be sparse or dense:**

Sparse: record for only some of the search key values

- (eg. Staff Ids: CS001, EE001, MA001).
- **Applicable toordered data files only.**

Dense: record for every search key value.

•**(eg. Staff Ids:**CS001, CS002, .. CS089, EE001, EE002, ..

Indexes

- TERMINOLOGY
  - Data file: a file containing the logical records
  - Index file: a file containing the index records
  - Indexing field: the field used to order the index records in the index file
  - Key: One or more fields which can uniquely identify a record (eg. No 2 students have the same student ID).

TYPES OF INDEXES

- **Primary Index:** An index ordered in the same way as the data file, which is sequentially ordered according to a key. (The indexing field is equal to this key.)
- Most of the time it is sparse index.
- **Secondary Index** An index that is defined on a non ordering field of the data file.
  - Extra index other than primary index.
  - It uses a index field as an **candidate key** (distinct values) or non key field(containing duplicate values)
  - The indexing field need not contain unique values.
  - Its dense index.
- **Clustered index :** If index file and data files both are organized on non key field called as clustered indexing.
  - Clustered index is a sparse index
- **Multilevel index :** hierarchy of index is maintained.

- A data file can associate with at most one primary index plus several secondary indexes.

Indexed Sequential Files

- **What are Indexed Sequential Files?**

A sorted data file with a primary index(Ordered Sequential file with index)

- **Advantage of an Indexed Sequential File**

Allows both sequential processing

and individual record retrieval through the index. (i.e both sequential as well as random access to data).

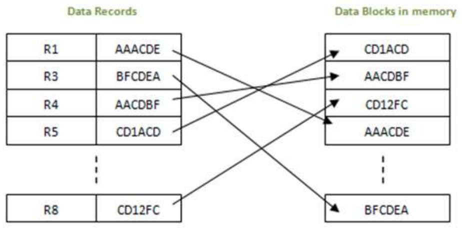
- Method is known as **Index Sequential Access Method (ISAM)**
- **Structure of an Indexed Sequential File**

o A primary storage area(actual storage area)

o A separate index or indexes(dictionary pertaining physical location of records)

o An overflow area(if primary storage area is exhausted)

Index sequential file



Random or direct file organization

- Records are stored randomly but accessed directly.
- To access a file stored randomly, a record key is used to determine where a record is stored on the storage media.
- Magnetic and optical disks allow data to be stored and accessed randomly.

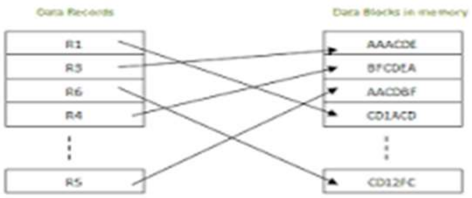
Direct Access fiels

- Insertion, deletion and searching of records can be done at random.

Fixed Record size

- Location of the records are computed by using their unique key (identifier)
- Logical ordering of records may or may not correspond to their physical sequence.
- Random access files can be updated in place.
- Often a HASH function is used to locate a record

Direct Access fiels



Functions required for DAF

- open - open a file- specify how it's opened (read/write) and type (binary/text)
- close - close an opened file
- read - read from a file
- write - write to a file
- seekp/seekg- move a file pointer to somewhere in a file
- tellp/tellp - tell you where the file pointer is located

read( )- read a block of binary data or reads a fixed number of bytes from the specified stream and store in a buffer.

Syntax : Stream_object.read((char *)& Object, sizeof(Object));

e.g **file.read((char *)&s, sizeof(s));**

write( ) – write a block of binary data or writes fixed number of bytes from a specific memory location to the specified stream.

Syntax : Stream_object.write((char *)& Object, sizeof(Object));

e.g **file.write((char *)&s, sizeof(s));**

49

The file pointer indicates the position in the file at which the next input/output is to occur.

Moving the file pointer in a file for various operations viz modification, deletion , searching etc. Following functions are used

seekg(): It places the file pointer to the specified position in input mode of file.

e.g file.seekg(p,ios::beg); or

file.seekg(-p,ios::end), or

file.seekg(p,ios::cur)

i.e to move to p byte position from beginning, end or current position.

50

seekp(): It places the file pointer to the specified position in output mode of file.

e.g file.seekp(p,ios::beg); or file.seekp(-p,ios::end), or file.seekp(p,ios::cur)

i.e to move to p byte position from beginning, end or current position.

tellg(): This function returns the current working position of the file pointer in the input mode.

e.g int p=file.tellg();

tellp(): This function returns the current working position of the file pointer in the output mode.

e.f int p=file.tellp();

51

File.seekp(File.tellg()-sizeof(stock);

OR

File.seekp(-sizeof(stock),ios::cur);

52

Goals

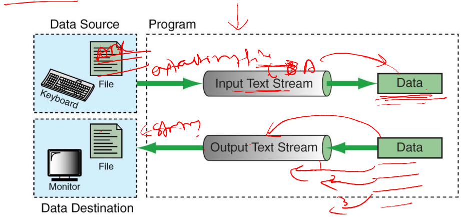
By the end of this unit you should understand …

- … how to open a file to write to it.
- … how to open a file to read from it.
- … how to open a file to append data to it.
- … how to read strings from a file.
- … how to write strings to a file.

53

Streams

- In C++, we input/output data using streams. We can associate a stream with a device (i.e. the terminal) or with a file.
- C supports two types of files
  - Text Stream Files
  - Binary Stream Files



from Figure 7-1 in **Forouzan & Gilberg***, p. 395*

54

C++ File I/O – Outline

- Introduction:
- Streams
- Creating a sequential file
- Iostream Library Header Files
- Stream Input/Output Classes and Objects
- File operations in C++

55

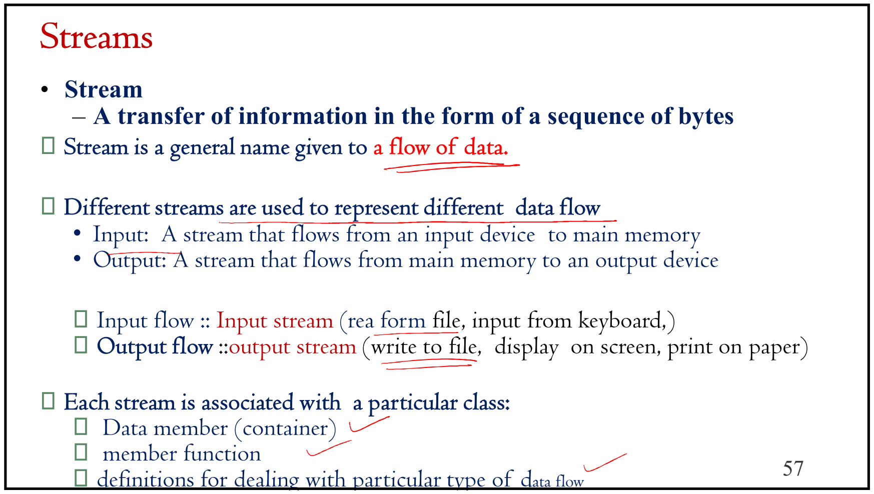
Introduction

- Many C++ I/O features are object-oriented
  - Use references, function overloading and operator overloading
- C++ uses type safe I/O
  - Each I/O operation is automatically performed in a manner sensitive to the data type
- Extensibility
  - Users may specify I/O of user-defined types as well as standard types

C++ → text →

Binary →

56



Streams

- **Stream**
  - **A transfer of information in the form of a sequence of bytes**

□ Stream is a general name given to a flow of data.

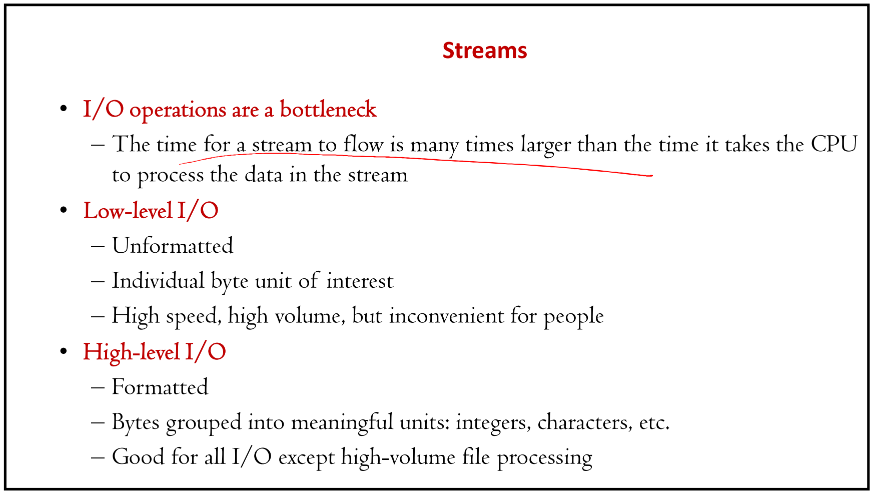
□ Different streams are used to represent different data flow
- Input: A stream that flows from an input device to main memory
- Output: A stream that flows from main memory to an output device

□ Input flow :: Input stream (rea form file, input from keyboard,)
□ Output flow ::output stream (write to file, display on screen, print on paper)

□ Each stream is associated with a particular class:
□ Data member (container)
□ member function
□ definitions for dealing with particular type of data flow

57

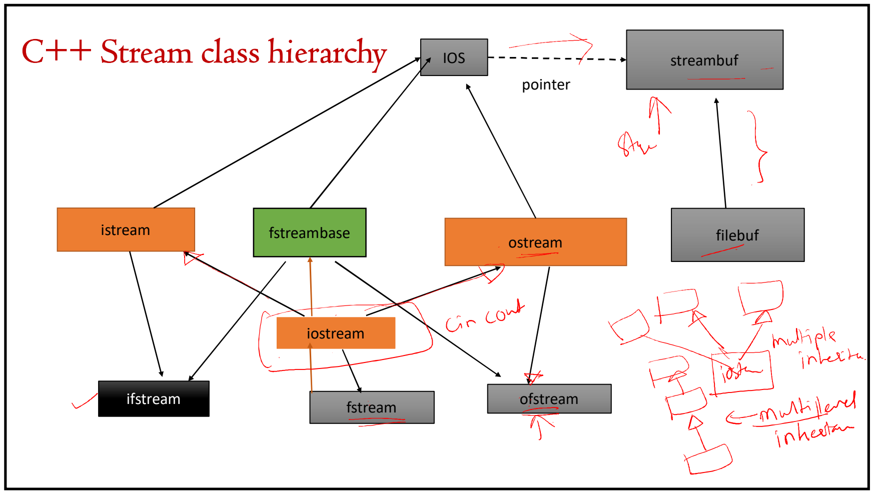
57



Streams

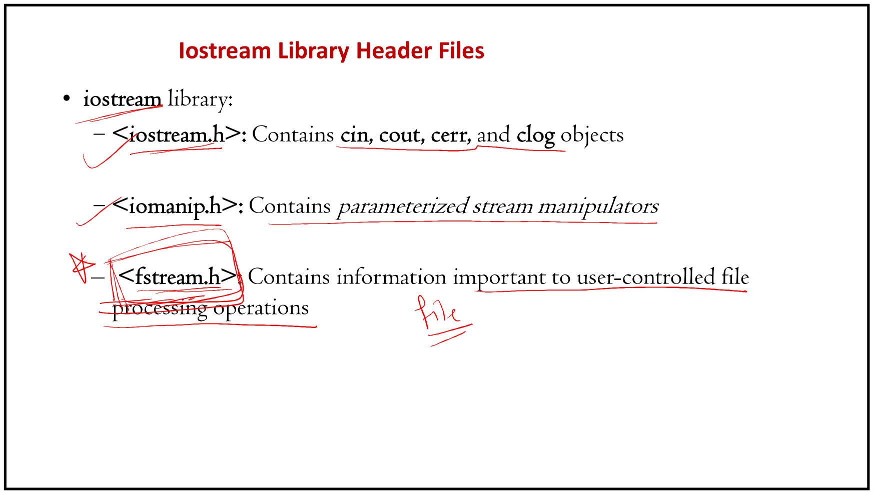
- I/O operations are a bottleneck
  - The time for a stream to flow is many times larger than the time it takes the CPU to process the data in the stream
- Low-level I/O
  - Unformatted
  - Individual byte unit of interest
  - High speed, high volume, but inconvenient for people
- High-level I/O
  - Formatted
  - Bytes grouped into meaningful units: integers, characters, etc.
  - Good for all I/O except high-volume file processing

58



C++ Stream class hierarchy

59



Iostream Library Header Files

- iostream library:
  - <iostream.h>: Contains cin, cout, cerr, and clog objects
  - <iomanip.h>: Contains *parameterized stream manipulators*
  - <fstream.h>: Contains information important to user-controlled file processing operations

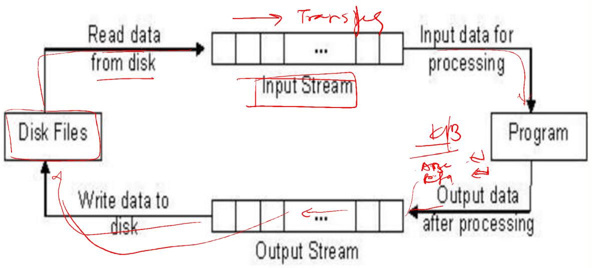
file

60

Stream Input/Output Classes and Objects

- istream: input streams
  - cin >> someVariable;
    - cin knows what type of data is to be assigned to someVariable (based on the type of someVariable).
- ostream: output streams
  - cout << someVariable;
    - cout knows the type of data to output
  - cerr << someString;
    - Unbuffered - prints someString immediately.
  - clog << someString;
    - Buffered - prints someString as soon as output buffer is full or flushed

File input and output streams



File creations and operations on it

- Declare file pointer
- Open file
- Write/output into the file
- Read /retrieve from the file
- Perform other operations on the file

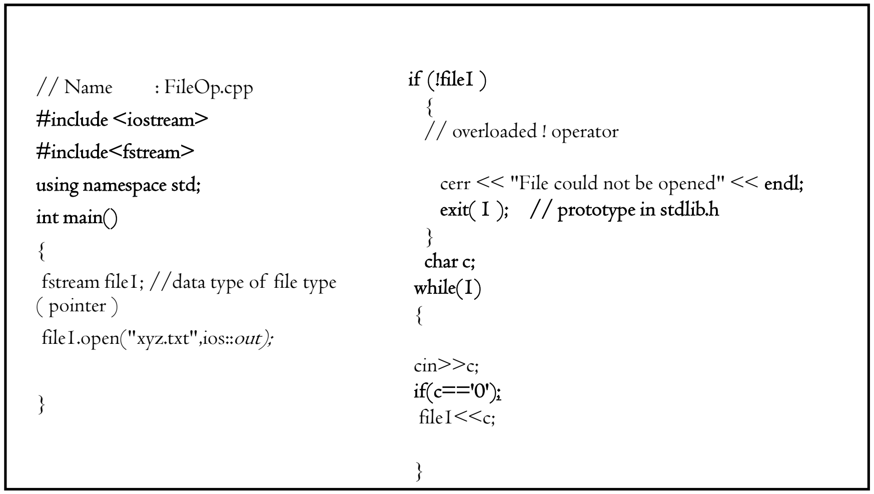
Creating a sequential file

```
// Create a sequential file
#include <iostream.h>
#include <fstream.h>
#include <stdlib.h>
int main()
{
  // ofstream constructor opens file
  ofstream  outFile( “Account.dat", ios::out );

  if ( !outFile ) {  // overloaded ! operator
    cerr << "File could not be opened" << endl;
    exit( 1 );   // prototype in stdlib.h
  }
cout << "Enter the account, name, and balance.\n"
     << "Enter end-of-file to end input.\n? ";
  int account;
  char name[ 30 ];
  float balance;

  while ( cin >> account >> name >> balance ) {
    outFile << account << ' ' << name
            << ' ' << balance << '\n';
    cout << "? ";
  }

  return 0;  // ofstream destructor closes file
}
```



```
// Name        : FileOp.cpp
#include <iostream>
#include<fstream>
using namespace std;
int main()
{
 fstream file1; //data type of file type
( pointer )
 file1.open("xyz.txt",ios::out);

}
```

```
if (!file1 )
   {
  // overloaded ! operator

      cerr << "File could not be opened" << endl;
      exit( 1 );    // prototype in stdlib.h
   }
  char c;
 while(1)
 {

 cin>>c;
 if(c=='0');
  file1<<c;

 }
```