1918
TALLINNA
TEHNIKAÜLIKOOL

Programmeerimine I ja II

Failid & nende kasutamine

Vladimir Viies Vladimir.viies@gmail.com

Tallinna Tehnikaülikool

Euroopa Liit
Euroopa Sotsiaalfond

Eesti tuleviku heaks

Files

FAILID

Faili mõiste

- **Fail on vähim nimega ühte tüüpi andmete kogum, mida saab salvestada arvuti mällu ja teisaldada ühest mälu tasemest teise.**
- Andmete säilitamiseks on vajalik need kirjutada faili.
- Programmeerimise kursustes vaatleme peamiselt tekstifaile *(failinimi.failitüüp, tekstifailide puhul on tüübi tähis txt, näiteks* arvud.txt*).*
- Tekstifailid koosnevad sümbolitest (char) ja andmevahetus tekstifailiga toimub stringide kaudu.

Failisüsteem

- File Concept
- Access Methods
- Directory Structure
- File-System Mounting
- File Sharing
- Protection

Failide liigid

- Contiguous logical address space
- Types:
  - Data
    - numeric
    - character
    - binary
  - Program

Faili struktuur

None - sequence of words, bytes

Simple record structure

- Lines
- Fixed length
- Variable length

Complex Structures

- Formatted document
- Relocatable load file

Can simulate last two with first method by inserting appropriate control characters

Who decides:

- Operating system
- Program

Faili atribuudid

- **Name** – only information kept in human-readable form
- **Identifier** – unique tag (number) identifies file within file system
- **Type** – needed for systems that support different types
- **Location** – pointer to file location on device
- **Size** – current file size
- **Protection** – controls who can do reading, writing, executing
- **Time, date, and user identification** – data for protection, security, and usage monitoring
- Information about files are kept in the directory structure, which is maintained on the disk

Operatsioonid failidega

- File is an **abstract data type**
- **Create**
- **Write**
- **Read**
- **Reposition within file**
- **Delete**
- **Truncate**
- *Open(F_i)* – search the directory structure on disk for entry F_i, and move the content of entry to memory
- *Close (F_i)* – move the content of entry F_i in memory to directory structure on disk

Faili töötlemiseks vajalikud andmed

- Several pieces of data are needed to manage open files:
  - File pointer: pointer to last read/write location, per process that has the file open
  - File-open count: counter of number of times a file is open – to allow removal of data from open-file table when last processes closes it
  - Disk location of the file: cache of data access information
  - Access rights: per-process access mode information
  - File type

Faili tüübid

file type	usual extension	function
executable	exe, com, bin or none	ready-to-run machine-language program
object	obj, o	compiled, machine language, not linked
source code	c, cc, java, pas, asm, a	source code in various languages
batch	bat, sh	commands to the command interpreter
text	txt, doc	textual data, documents
word processor	wp, tex, rtf, doc	various word-processor formats
library	lib, a, so, dll	libraries of routines for programmers
print or view	ps, pdf, jpg	ASCII or binary file in a format for printing or viewing
archive	arc, zip, tar	related files grouped into one file, sometimes compressed, for archiving or storage
multimedia	mpeg, mov, rm, mp3, avi	binary file containing audio or A/V information

Failidele juurdepääsu meetodid

Sequential Access

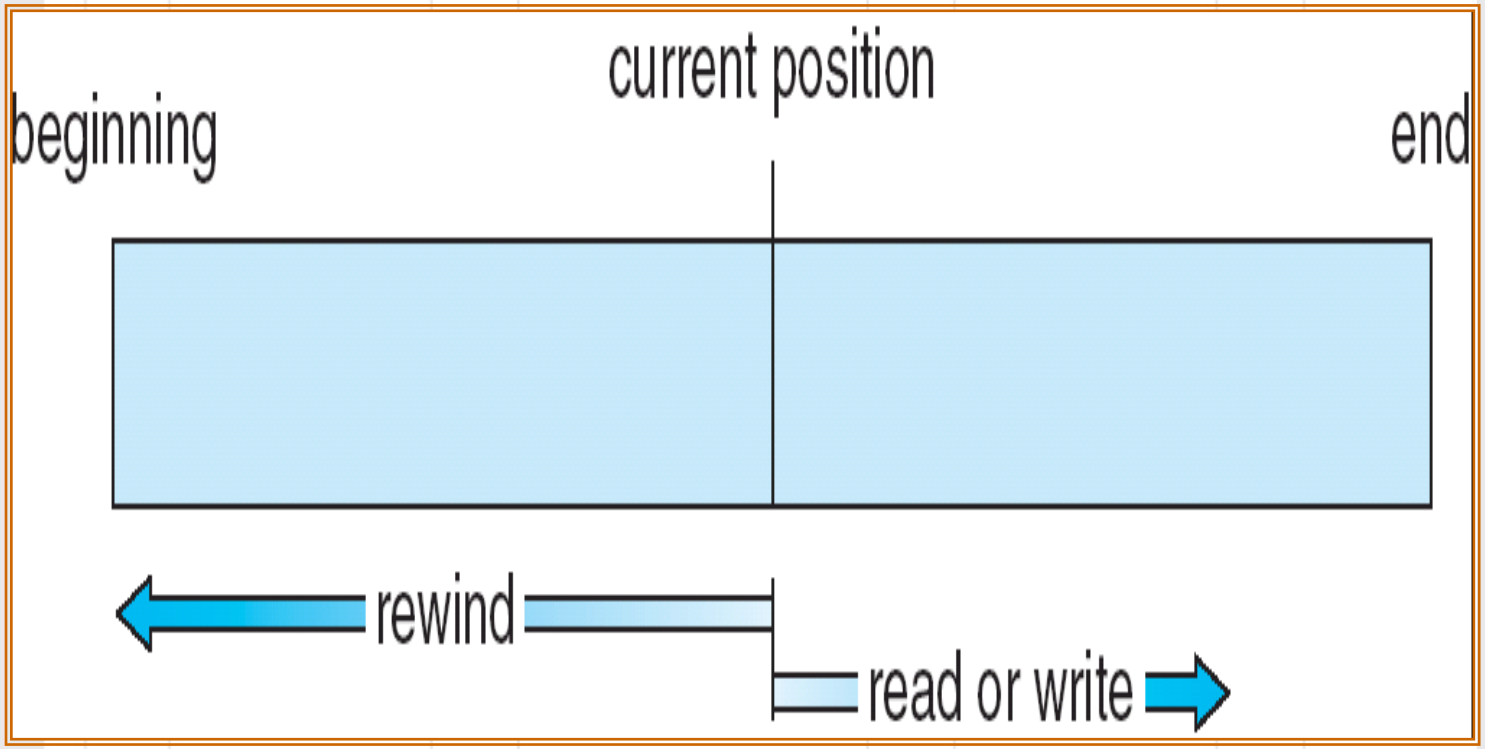
read next
write next
reset
no read after last write
(rewrite)

Direct Access

read *n*
write *n*
position to *n*
read next
write next
rewrite *n*

n = relative block number

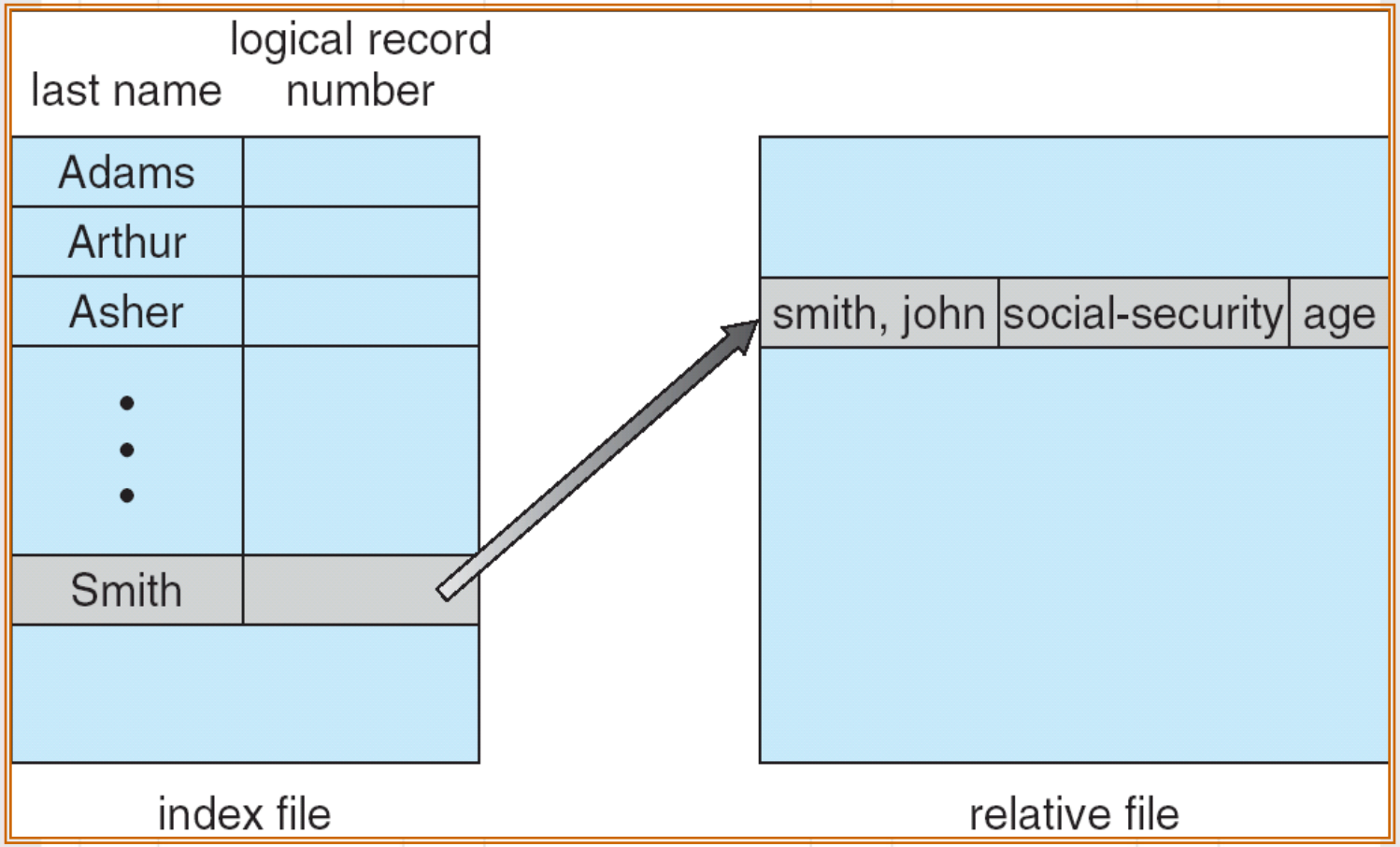
Faili järjestikuline töötlus



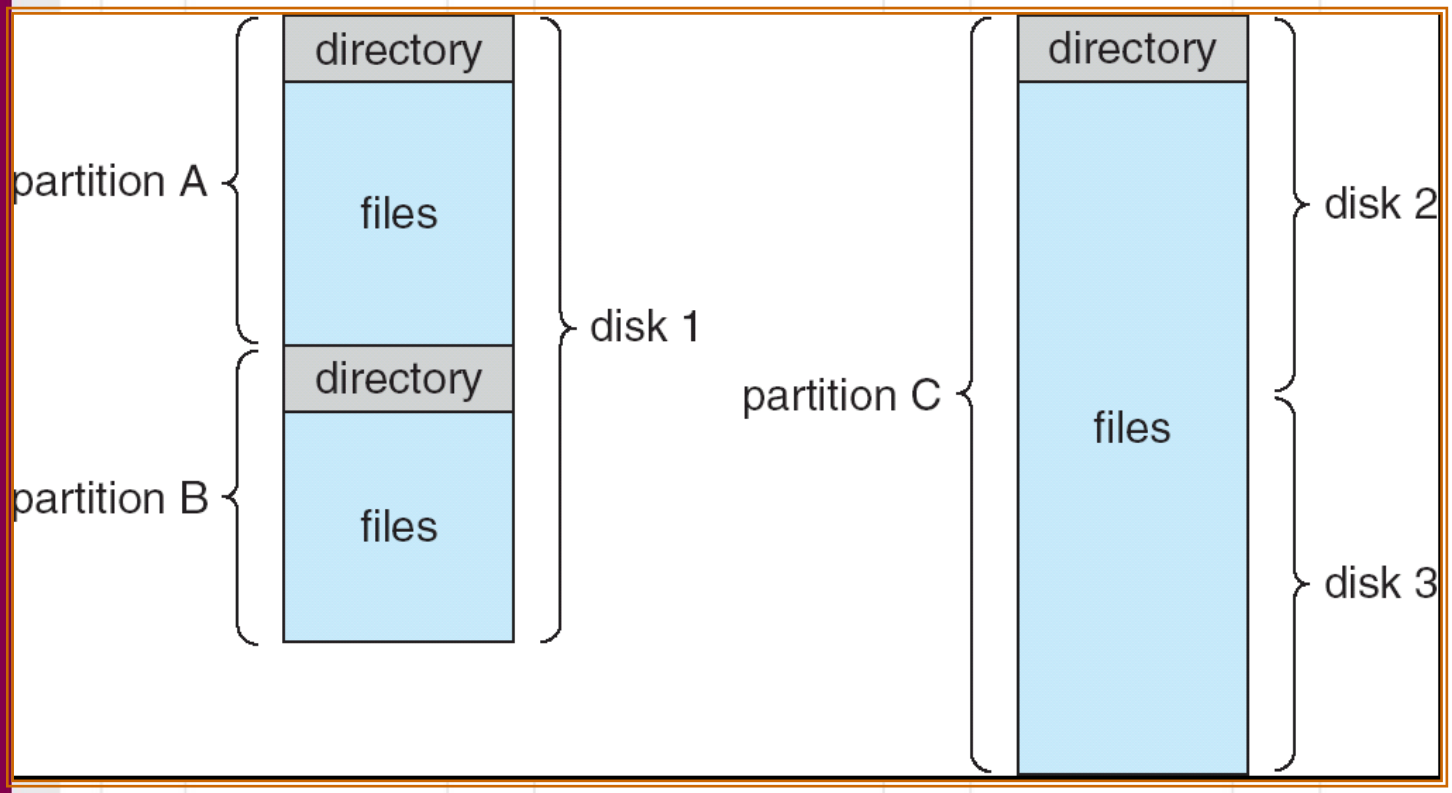
Otsejuurdepääs andmetele failis

sequential access	implementation for direct access
reset	*cp* = 0;
read next	*read cp*; *cp* = *cp* + 1;
write next	*write cp*; *cp* = *cp* + 1;

Indeksfailide näide



Tavaline failisüsteem



Teegiga seotud operatsioonid

- Search for a file
- Create a file
- Delete a file
- List a directory
- Rename a file

Teegi korraldus haldamise lihtsustamiseks

- Efficiency – locating a file quickly
- Naming – convenient to users
  - Two users can have same name for different files
  - The same file can have several different names
- Grouping – logical grouping of files by properties, (e.g., all Java programs, all games, …)

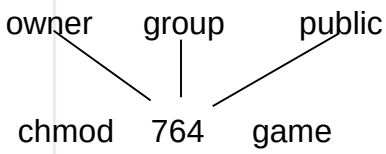
Juurdepääsu nimistu ja rühmitamine

- Mode of access: read, write, execute
- Three classes of users

			RWX
a) **owner access**	7	⇒	1 1 1
b) **group access**	6	⇒	RWX 1 1 0
c) **public access**	4	⇒	RWX 1 0 0

Ask manager to create a group (unique name), say G, and add some users to the group.

- For a particular file (say *game*) or subdirectory, define an appropriate access.



Attach a group to a file

chgrp G game

Kaitse

- File owner/creator should be able to control:
  - what can be done
  - by whom
- Types of access
  - **Read**
  - **Write**
  - **Execute**
  - **Append**
  - **Delete**
  - **List**

Failide praktiline kasutamine C keeles

Failide loomine

Failikirjeldus paikneb struktuuris(andmekogumis) nimega *FILE*.

Iga faili poole pöörduv funktsioon vajab ühe parameetrina viita sellele struktuurile.

Viit deklareeritakse nii:

FILE *viit_failile;

Paljudes näidetes kasutatakse failiviida nime ‘fp’, kuid see pole mitte võtmesõna, vaid tuletatud sõnadest *file pointer*.

Faili kasutamise kirjeldus

- Fail tuleb juurdepääsuks kindlasti avada. Toimub see funktsiooni ‘fopen’ väljakutsega, mille prototüüp on
- FILE* fopen (const char*, const char*);
- viit_failile= fopen (const char*, const char*);
- Esimene parameeter on faili nimi märgistringina. Teine parameeter peab olema märgistring:
- “r” tähendab sisendfaili;
- “w” tähendab väljundfaili (kui selles olid andmed, kirjutatakse nad üle);
- “a” tähendab väljundfaili, mille lõppu lisatakse andmeid.

Töö failidega

Print ja scan käsud saavad juurde f tähe ja failiviida:

fprintf (FILE*, const char*, ...);

fscanf (FILE*, const char*, ...);

Sisendfaili lõpu kontroll toimub funktsiooniga ‘**feof’.** Tema väärtus on tõene, kui ollakse jõutud faili lõppu:

feof (FILE*);

Faili sulgemine toimub funktsiooniga ‘**fclose**’:

fclose (FILE*);

Failide kasutamine C programmis

Faili avamine

Viitadega saab ka failist lugeda ja kirjutada.

```
#include <stdio.h>
#include <stdlib.h>
int main (void)
{FILE *loe= fopen("esimene.txt", "r");
fprintf(loe, "Esimene proov");
fclose(loe); //kuulub iga viisaka failiga töö juurde
}
/* muutuja "loe" kaudu saab avada faili esimene.txt lugemiseks – r, loetakse tekst ja suletakse fail
 */
```

Failist lugemine

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{ char rida[128];
FILE *sisse=fopen("tervitus.txt", "r");  // r - lugemiseks
    while(!feof(sisse))
        { // fscanf(sisse, "%s", rida);
        //võtab ühe sõna
        fgets(rida, 128, sisse);
        printf("%s", rida); }
fclose(sisse);
return 1; }
```

fsacnf ja fgets on võimalik mõlemat kasutada, kuid mitte samaaegselt

Faili kirjutamine

Näitena oli meil:

*FILE *kirjuta= fopen(“esimene.txt”, “w”);*
fprintf(kirjuta, “Esimene proov”);//faili trükitav tekst
fclose(kirjuta);//kuulub iga viisaka failiga töö juurde

“w” korral siis iga faili avamisega kirjutatakse eelnev sisu üle

*FILE *kirjuta= fopen(“esimene.txt”, “a”);*
fprintf(kirjuta, “Esimene proov”);

“a” korral iga avamise ja kirjutamise kord lisatakse olemasolevale failile.

SOME POSSIBILITIES OF USING FILES

FAILIDE KASUTAMISE LISA VÕIMALUSUSI

Failid

PROGRAMMI TEKSTI JAOTAMINE ERINEVATE FAILIDE VAHEL

Programmi teksti jaotamine erinevate failide vahel

Kindlasti on kasulikum suurem programm jagada nii alamfunktsioonideks kui ka need omakorda jagada teema järgi erinevate failide vahel.
Näiteks: mitme erinevalt moodustatud massiivi väljatrükk oleks kasulik kirjutada omaette alamfunktsioonina. Kuid miks ka mitte eraldi failina.

Näide

Ülesandeks on moodustada massiiv ning trükkida see ekraanile;
Vahetada kasutaja poolt täisarvuna etteantud indeksi väärtusega massiivi rida ja veerg.
Ekraanile trükkida iga samm, mida tehakse massiiviga.
Kahtlemata pika programmitekstina läheb kogu kood liiga raskelt hallatavaks ning lohisevaks.
Alustuseks tuleb moodustada ekraanile välja trükkimiseks eraldi alamfunktsioon ning seejärel see eraldi faili tõsta.

Alamfunktsioonist failiks samm 0

```
#include<stdio.h>
#include<time.h>          //randomi jaoks
#include<stdlib.h>        //randomi jaoks
void valjastus(int n, int a[n][n])    //alamfunktsioon
{    int i,j;
     for(i=0;i<n;i++)
      {for(j=0;j<n;j++)
        { printf(" %d  |",a[i][j]);
         }
         printf("\n");
     }
     return ;
}
int main()                //peaprogramm
{   int i,j;
.... }
```

Alamfunktsioonist failiks sammud 1 ja 2

- Tõsta kogu alamfunktsiooni *valjastus(int n, int a[n][n])* tekst eraldi faili nimega *valjastus.c*
- Kirjuta põhiprogrammi teekide hulka juurde ka lisatav fail.

Enne:

```
#include<stdio.h>
#include<time.h>          //randomi jaoks
#include<stdlib.h>        //randomi jaoks
```

Pärast:

```
#include<stdio.h>
#include<time.h>          //randomi jaoks
#include<stdlib.h>        //randomi jaoks
#include “valjastus.c”       //sinu loodud fail alamfunktsiooniga
                          valjastus(int n, int a[n][n])
```

- Kompileeri, käivita ja kontrolli programmi tööd.

Veel väikeseid meeldetuletusi:

- Ükskord loodud massivi väljatrükki ei pea edaspidi järgmisesse programmi uuesti kopeerima.
- Piisab kui kirjutada teekide juurde lause

#include "valjastus.c" ning tuleb kontrollida, et see lisatav fail alamfunktsiooniga *valjastus()* oleks uue programmiga samas kataloogis.

Edu töö lihtsustamisel!

FAILIDE VÄLINE SORTIMINE I osa

sortimine nelja failiga

Alati saab tekkida olukord, kus sisemälumaht, mis ülesandel kasutada - osutub ebapiisavaks. Seepärast on loodud rida algoritme, mis praktiliselt sisemälu ei vaja. Loomulikult on selline algoritm aeglane, aga näiteks taga-plaan ülesandena võib teda edukalt rakendada.

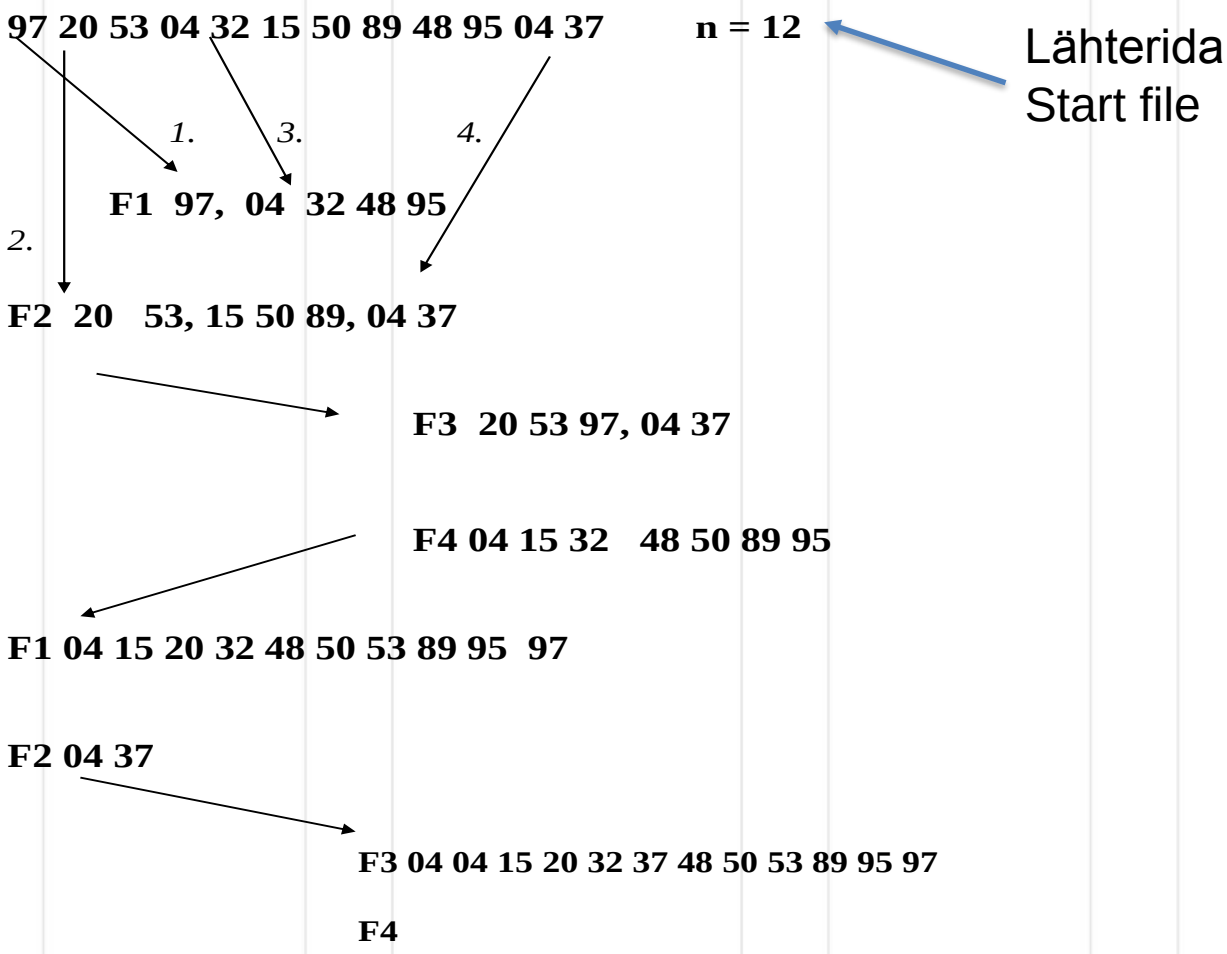
- There can always be a situation where the amount of internal memory that can be used for a task - turns out to be insufficient. Therefore, a number of algorithms have been created that practically do not require internal memory. Of course, such an algorithm is slow, but for example, as a background task, it can be successfully implemented.

Järgmisel slaidil on ühe algoritmi kirjeldus mille alusel tuleks koostada programm.

The next slide describes one of the algorithms that should be used to write the program.

FAILIDE VÄLINE SORTIMINE II osa

sortimine nelja failiga



Täname, et läbisid failimooduli !
Jätka aine omandamist ja
lahenda kodutööd kasutades
faile !

Tutvu ainetega Infotehnoloogia teaduskonna
õppematerjalide kodulehel www.tud.ttu.ee