Computer Programming 143 – Lecture 27
File Processing III

Electrical and Electronic Engineering Department
University of Stellenbosch

Prof Johan du Preez
Mr Callen Fisher
Dr Willem Jordaan
Dr Hannes Pretorius
Mr Willem Smit

Copyright & Disclaimer

Copyright

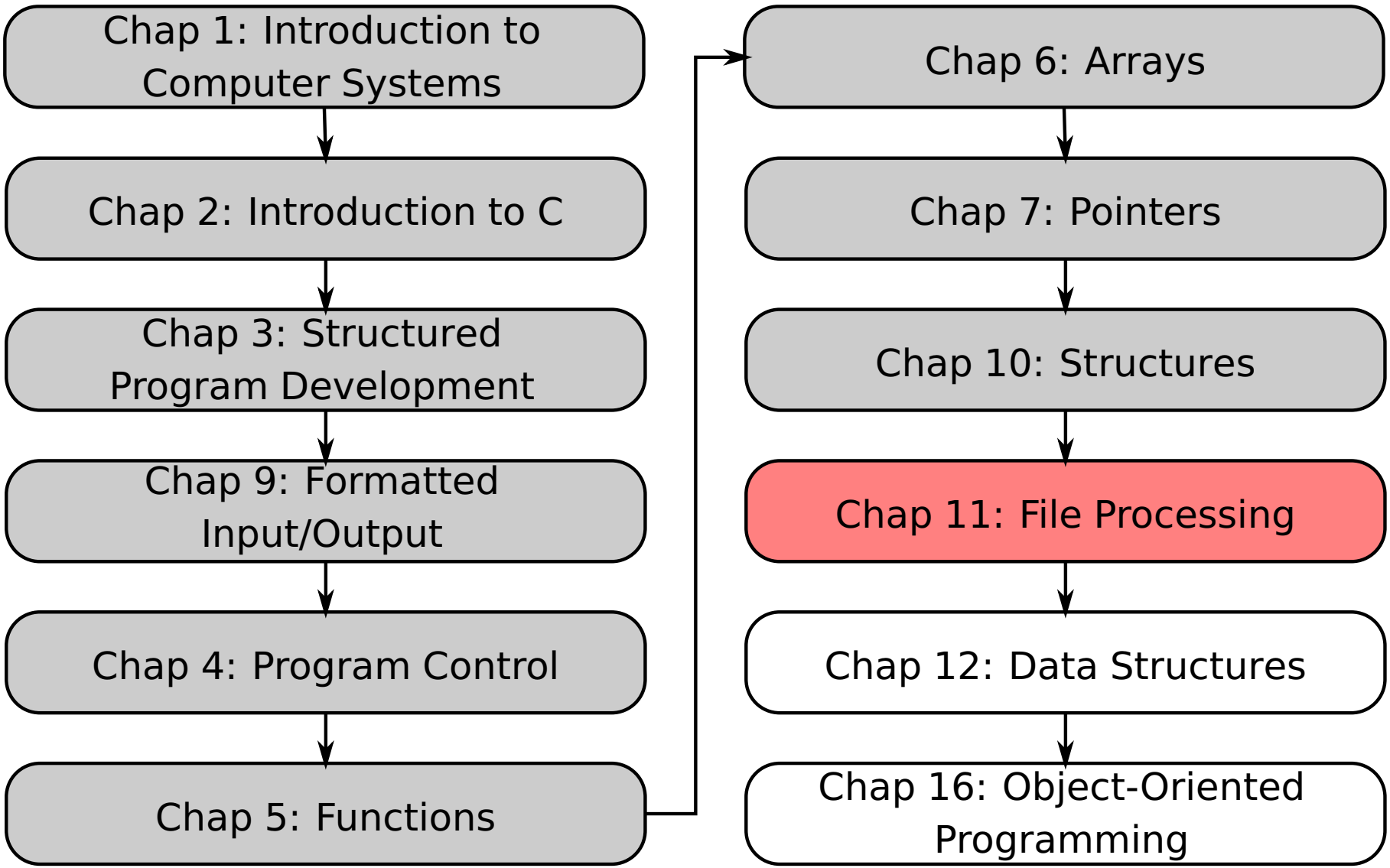
Disclaimer

This content is provided without warranty or representation of any kind. The use of the content is entirely at your own risk and Stellenbosch University (SU) will have no liability directly or indirectly as a result of this content.

The content must not be assumed to provide complete coverage of the particular study material. Content may be removed or changed without notice.

The video is of a recording with very limited post-recording editing. The video is intended for use only by SU students enrolled in the particular module.

Module Overview



Lecture Overview

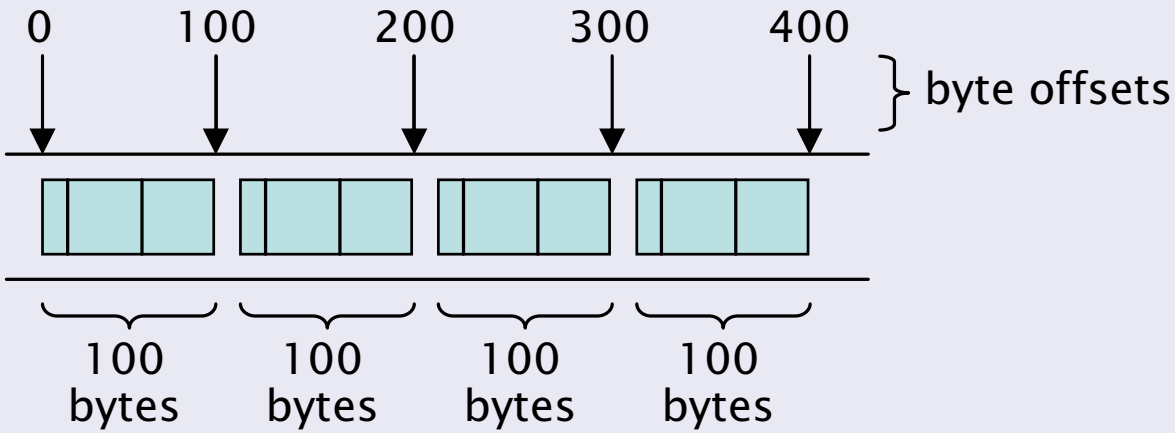
1. Random-Access Files (11.5-11.6)
2. Writing Data to a Random-Access File (11.7)
3. Reading Data from a Random-Access File (11.8)

11.5 Random-Access Files

Random-Access Files

- Access individual records without searching through other records
- Instant access to records in a file
- Data can be inserted without destroying other data
- Data previously stored can be updated or deleted without overwriting

Implemented using fixed-length records



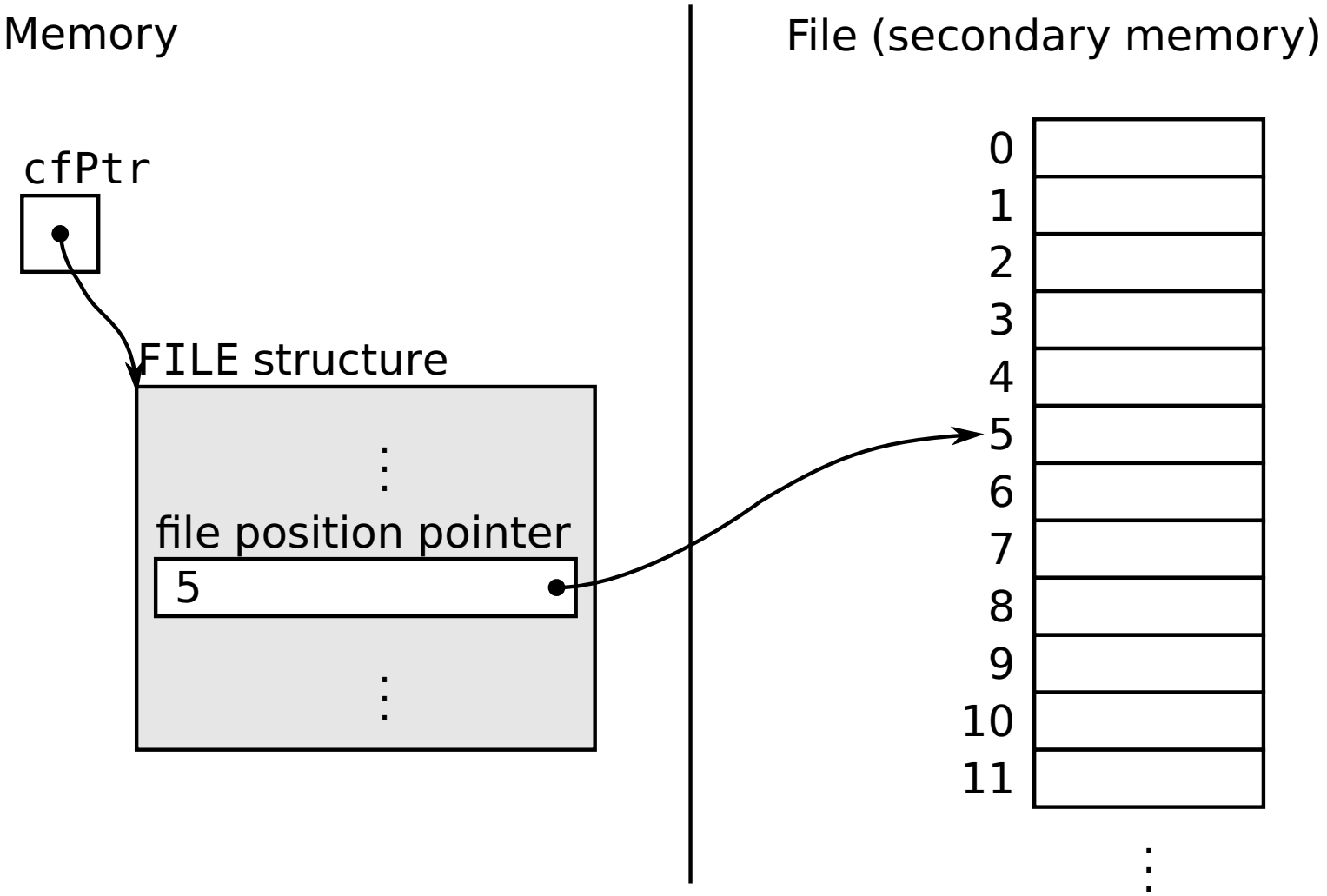
- Text files usually do not have fixed-length records and are therefore handled sequentially

11.5 Random-Access Files

Data in random-access files

- Random-access files are usually binary/unformatted (stored as “raw bytes”)
  - All data of the same type (ints, for example) uses the same amount of space as in memory
  - All records of the same type have a fixed length
  - Data not human readable

11.7 Writing Data to a Random-Access File I



11.7 Writing Data to a Random-Access File II

fseek()

- Sets file position pointer to a specific position

```
int fseek( FILE *stream, long int offset, int whence );
```

  - `stream` – pointer to file
  - `offset` – file position to seek (relative to the position specified in `whence`)
  - `whence` – specifies reference point in file for `offset`
    - `SEEK_SET` – seek starts at beginning of file
    - `SEEK_CUR` – seek starts at current location in file
    - `SEEK_END` – seek starts at end of file

Writing data with `fseek()`

```
struct coord xPnt = { 1.0, 5.0 };
fseek( fPtr, 4 * sizeof( struct coord ), SEEK_SET );
fwrite( &xPnt, sizeof( struct coord ), 1, fPtr );
```

- Positions the file position pointer at the 5th element from the beginning of the file and writes structure `xPnt`.

11.8 Reading Data from a Random-Access File

Reading data with `fseek()`

- Reads a specified number of bytes from a file into memory

```
struct coord xPnt;
fseek( fPtr, 4 * sizeof( struct coord ), SEEK_SET );
fread( &xPnt, sizeof( struct coord ), 1, fPtr );
```

- Can read several fixed-size array elements
  - Provide pointer to array
  - Indicate number of elements to read
- To read multiple elements, specify in third argument

11.6 Creation of Binary File for Random Access I

```
#include <stdio.h>

// clientData structure definition
struct clientData {
    int acctNum; // account number
    char lastName[ 15 ]; // account last name
    char firstName[ 10 ]; // account first name
    double balance; // account balance
}; // end structure clientData

int main( void )
{
    int i; // counter used to count from 1-100

    // create clientData with default information
    struct clientData blankClient = { 0, "", "", 0.0 };

    FILE *cfPtr; // credit.dat file pointer
```

11.6 Creation of Binary File for Random Access II

```
    // fopen opens the file; exits if file cannot be opened
    if ( ( cfPtr = fopen( "credit.dat", "wb" ) ) == NULL ) {
        printf( "File could not be opened.\n" );
    } //end if
    else {
        // output 100 blank records to file
        for ( i = 1; i <= 100; i++ ) {
            fwrite( &blankClient, sizeof( struct clientData ), 1,
                    cfPtr );
        } // end for

        fclose( cfPtr ); // fclose closes the file
    } // end else

    return 0; // indicates successful termination
} // end main
```

11.7 Writing Data in Random-Access File I

```c
#include <stdio.h>

// clientData structure definition
struct clientData {
    int acctNum; // account number
    char lastName[ 15 ]; // account last name
    char firstName[ 10 ]; // account first name
    double balance; // account balance
}; // end structure clientData

int main( void )
{
    FILE *cfPtr; // credit.dat file pointer

    // create clientData with default information
    struct clientData client = { 0, "", "", 0.0 };
```

11.7 Writing Data in Random-Access File II

```
// fopen opens the file; exits if file cannot be opened
if ( ( cfPtr = fopen( "credit.dat", "rb+" ) ) == NULL ) {
    printf( "File could not be opened.\n" );
} // end if
else {
    // require user to specify account number
    printf( "Enter account number"
        " ( 1 to 100, 0 to end input )\n? " );
    scanf( "%d", &client.acctNum );

    // user enters information, which is copied into file
    while ( client.acctNum != 0 ) {
        // user enters last name, first name and balance
        printf( "Enter lastname, firstname, balance\n? " );

        // set record lastName, firstName and balance value
        scanf( "%s%s%lf", client.lastName,
        client.firstName, &client.balance );
```

11.7 Writing Data in Random-Access File III

```
            // seek position in file to user-specified record
            fseek( cfPtr, ( client.acctNum - 1 ) *
            sizeof( struct clientData ), SEEK_SET );

            // write user-specified information in file
            fwrite( &client, sizeof( struct clientData ), 1, cfPtr );

            // enable user to input another account number
            printf( "Enter account number\n? " );
            scanf( "%d", &client.acctNum );
        } // end while

        fclose( cfPtr ); // fclose closes the file
    } // end main

    return 0; // indicates successful termination
} // end main
```

11.8 Reading Data from a Random-Access File I

```c
#include <stdio.h>

// clientData structure definition
struct clientData {
    int acctNum; // account number
    char lastName[ 15 ]; // account last name
    char firstName[ 10 ]; // account first name
    double balance; // account balance
}; // end structure clientData

int main( void )
{
    FILE *cfPtr; // file pointer
    struct clientData client = {0, "", "", 0.0};
```

11.8 Reading Data from a Random-Access File II

```
// fopen opens the file
if((cfPtr = fopen("credit.dat", "rb")) == NULL)
{
        printf("File could not be opened.");
}
else
{
    // print headings
    printf(" %-6s%-16s%-11s%10s\n", "Acct", "Last Name",
    "First Name", "Balance");
```

11.8 Reading Data from a Random-Access File III

```
        // read all records from file
        while(!feof(cfPtr)){
        // read an entry out of file
            int result = fread(
                &client, sizeof(struct clientData), 1, cfPtr);

            // display record
            if(result != 0 && client.accNum != 0){
                printf(" %-6d%-16s%-11s%10.2f\n",
                client.accNum, client.lastName,
                client.firstName, client.balance);
            }
        }
        fclose(cfPtr);
    }
    return 0; // indicates successful termination
} // end main
```

Perspective

Today

File Processing III

- Random-access files
- Write randomly to a binary file
- Read randomly from a binary file

Next lecture

Dynamic Structures I

- Self-referential structures
- Dinamic data allocation
- Linked lists

Homework

1. Study Sections 11.6-11.9 in Deitel & Deitel
2. Do Self Review Exercises 11.1, 11.3 in Deitel & Deitel
3. Do Exercise 11.11 in Deitel & Deitel