Computer Programming 143 – Lecture 25
File Processing I

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

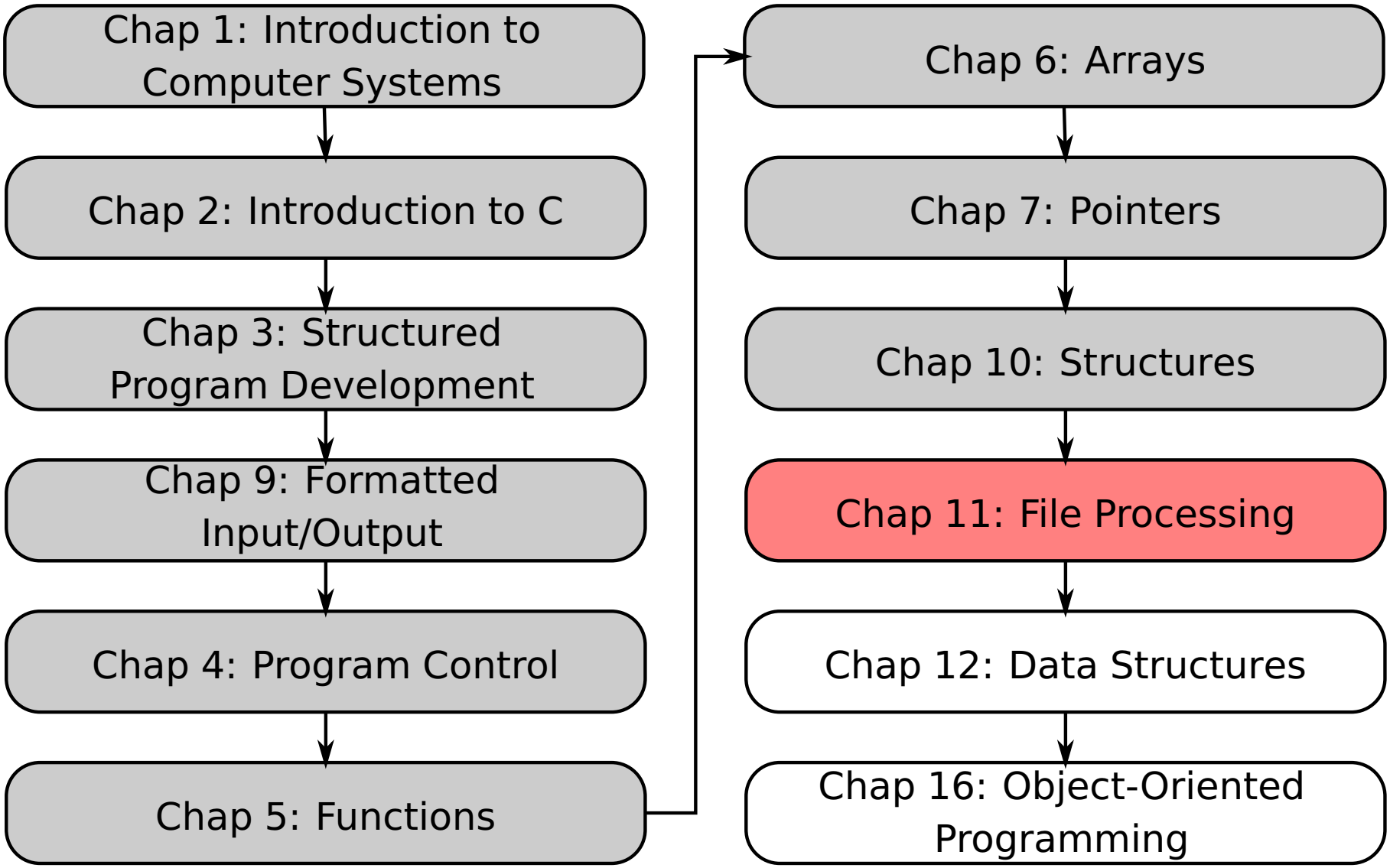
Disclaimer

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

11.1 Introduction

Data files

- Can be created, updated, and processed by C programs.
- Are used for permanent storage of large amounts of data.
  - Storage of data in variables and arrays is only temporary.

File Types in C

Binary files

- Unformatted (stored as “raw bytes”)
  - Data not human readable
- Written and read sequentially or randomly

Text files

- Data is stored as characters (`char`)
- Usually only written and read sequentially

Opening and Closing Files

Opening files

```
FILE *cfPtr;
cfPtr = fopen( someFilename.xyz", "r" );
```

- Declares pointer **cfPtr** that may point to a file
- Opens `someFilename.xyz` and let **cfPtr** point to its beginning
  - **"r"**: File is opened to read and as text file

Closing files

```
fclose( cfPtr );
```

- Closes file to which **cfPtr** points
- Files should always be closed after use

File open modes

Mode	Description
r	Open a file for reading.
w	Create a file for writing. If the file already exists, discard the current contents.
a	Append; open or create a file for writing at end of file.
r+	Open a file for update (reading and writing).
w+	Create a file for update. If the file already exists, discard the current contents.
a+	Append; open or create a file for update; writing is done at the end of the file.
rb	Open a file for reading in binary mode.
wb	Create a file for writing in binary mode. If the file already exists, discard the current contents.
ab	Append; open or create a file for writing at end of file in binary mode.
rb+	Open a file for update (reading and writing) in binary mode.
wb+	Create a file for update in binary mode. If the file already exists, discard the current contents.
ab+	Append; open or create a file for update in binary mode; writing is done at the end of the file.

11.2 Files and Streams

C views each file as a sequence of bytes

- File ends with the *end-of-file* marker
- or file ends at a specified byte

Stream created when a file is opened

- Communication channel between files and programs
- Opening a file returns a pointer to a `FILE` structure
- Three files with associated streams are automatically opened with program execution:
  - `stdin` – standard input (keyboard)
  - `stdout` – standard output (screen)
  - `stderr` – standard error (screen)

FILE structure

- File descriptor
  - Index into an operating system array, the *open file table*
- File Control Block (FCB)
  - Found in every array element, system uses it to administer the file

11.2 Files and Streams

Read/Write functions in standard library

A few useful functions from the standard C library.

- `int fgetc( FILE *stream );`
  - Reads one character from a file
  - Takes a `FILE` pointer as an argument
  - `fgetc( stdin )` equivalent to `getchar()`
- `char *fgets( char *str, int count, FILE *stream );`
  - Reads a line from a file
- `int fscanf( FILE *stream, const char *format, ... );`
  - File processing equivalents of `scanf()`
- `int fprintf( FILE *stream, const char *format, ... );`
  - File processing equivalent of `printf()`

11.3 Creating a Text File for Sequential Access

Additional functions for text files

- `int feof( FILE *stream );`
  - Returns true if *end-of-file* indicator (no more data to process) is set for the specified file
- `void rewind( FILE *stream );`
  - Resets the file position pointer to the beginning of the file

```c
#include <stdio.h>
#include <stdlib.h>
int main( void )
{
   FILE *fPtr; // file pointer
   char fileName[ 30 ]; // string to store file name
   char lineOfText[ 50 ] = "A single line of text";
   int myInt = 5;
   float myFloat = 1.345;
   // Next 2 lines illustrate a point - please do not use!
   fprintf( stdout, "Please enter filename: " );
   fscanf( stdin, "%29s", fileName );
   fPtr = fopen( fileName, "w" ); // opens file for writing
   if ( fPtr == NULL ) {
      printf( "ERROR - File could not be opened!\n" );
   }
   else { // what would the following write to the file?
      fprintf( fPtr, "*");
      fprintf( fPtr, "%s", lineOfText);
      fprintf( fPtr, "*");
      fprintf( fPtr, "\nmyInt = %d\nmyFloat = %.3f\n", myInt, myFloat );
      fclose( fPtr ); // closes file
   }
   return 0; // indicates successful termination
} // end function main
```

```c
#include <stdio.h>
#include <stdlib.h>
int main( void )
{
   FILE *fPtr; // file pointer
   char fileName[ 30 ]; // string to store file name
   char lineOfText[ 30 ], string1[ 20 ], string2[ 20 ];
   float myFloat;
   printf( "Please enter filename: " );
   scanf( "%29s", fileName ); // reads file name from user
   fPtr = fopen( fileName, "r" ); // opens file for reading
   if ( fPtr == NULL ) {
      printf( "ERROR - file could not be opened!\n" );
   }
   else { // what would the following code display?
      fgets( lineOfText, 29, fPtr );
      while ( !feof( fPtr ) ) {
         printf( "%s", lineOfText );
         fgets( lineOfText, 29, fPtr );
      }
      rewind( fPtr );
      printf( "%c\n", fgetc( fPtr ) );
      fgets( lineOfText, 29, fPtr );
      printf( " %s", lineOfText );
      fgets( lineOfText, 29, fPtr );
      printf( "%s", lineOfText );
      fscanf( fPtr, "%19s%19s%f", string1, string2, &myFloat );
      printf( "Value of myFloat = %.3f\n", myFloat );
      fclose( fPtr ); // closes file
   }
   return 0; // indicates successful termination
} // end main
```

Perspective

Today

File Processing I

- File types in C
- Opening and closing files
- Writing and reading from a text file

Next lecture

File Processing II

- Data hierarchy
- Binary files

Homework

1. Study Sections 11.1-11.4 in Deitel & Deitel