VTC Academy
Think Ahead, Beyond Boundaries

BASIC PROGRAMMING LANGUAGE

LESSON 13

File Handling

VTC Academy

@ 2021 VTC Academy

CONTENT

VTC Academy
Think Ahead, Beyond Boundaries

1. File I/O Concepts
2. Working with Streams
   - Text Stream
   - Binary Stream
3. I/O Functions:
   - `open()`, `close()`
   - `fread()`, `fwrite()`, `fseek()`
   - `fscanf()`, `fprintf()`
4. Summary

VTC Academy

@ 2021 VTC Academy

Why Files Are Needed?

VTC Academy
Think Ahead, Beyond Boundaries

- When a program is terminated, the entire data is lost. Storing in a file will preserve your data even if the program terminates.
- If you have to enter a large number of data, it will take a lot of time to enter them all.
  However, if you have a file containing all the data, you can easily access the contents of the file using a few commands in C.
- You can easily move your data from one computer to another without any changes.

VTC Academy

@ 2021 VTC Academy

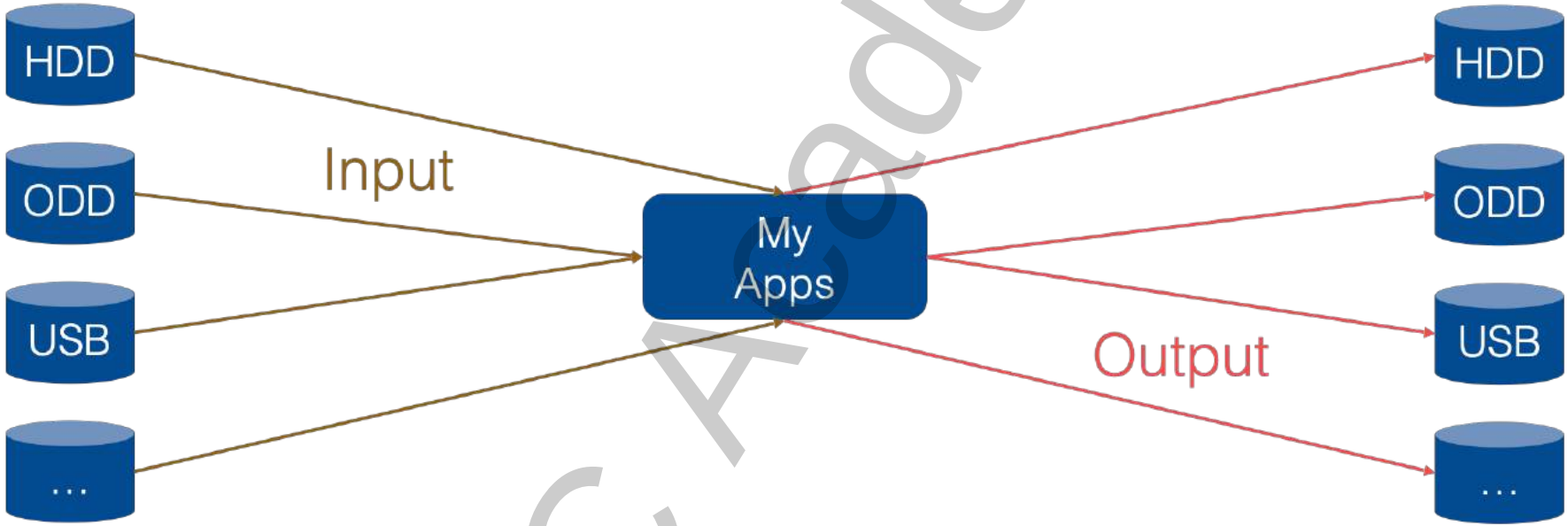
File Input / Output

- A file is a container in computer storage devices used for storing data.
- All I/O operations in C are carried out using functions from the standard library.
- This approach makes the C file system very powerful and flexible.
- I/O in C is unique because data may be transferred in its internal binary representation or in a human-readable text format.

VTC Academy

@ 2021 VTC Academy

File Input / Output



VTC Academy

@ 2021 VTC Academy

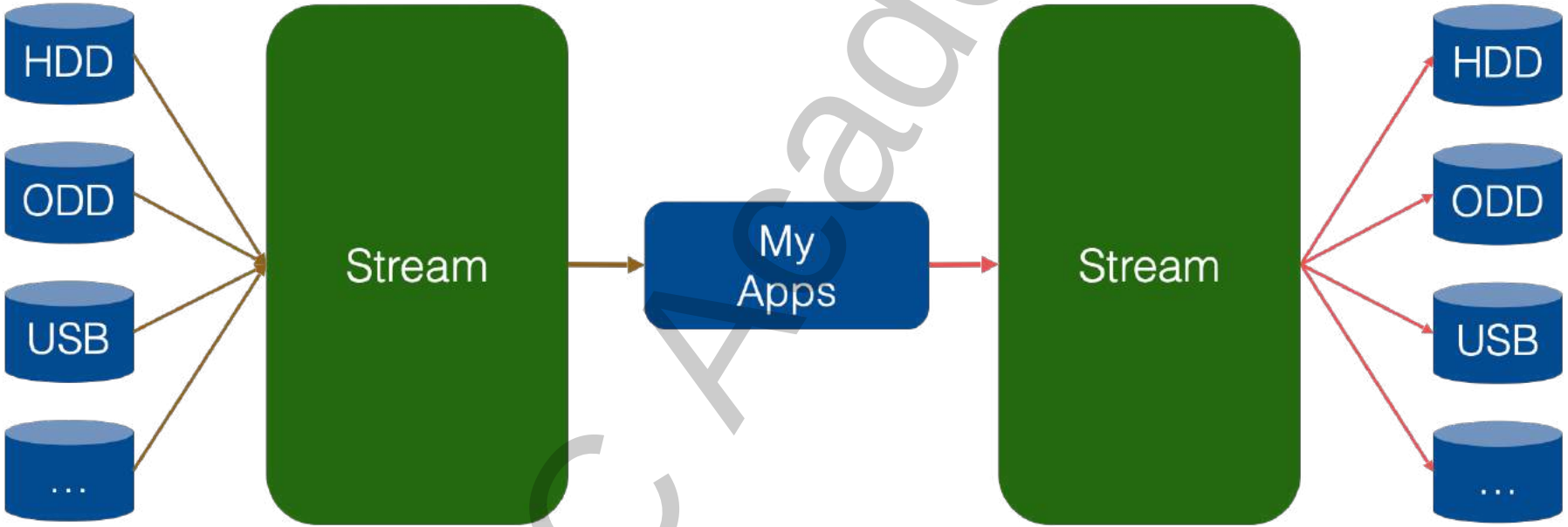
Streams

- The C file system works with a wide variety of devices including printers, disk drives, tape drives and terminals.
- Though all these devices are very different from each other, the buffered file system transforms each device into a logical device called a stream.
- Since all streams act similarly, it is easy to handle the different devices.
- There are two types of streams
  - Text stream
  - Binary stream

VTC Academy

@ 2021 VTC Academy

Streams



VTC Academy

@ 2021 VTC Academy

Text Streams

VTC Academy
Think Ahead, Beyond Boundaries

- A text stream is a sequence of characters that can be organized into lines terminated by a new line character.
- In a text stream, certain character translations may occur as required by the environment.
- Therefore, there may not be a one-to-one relationship between the characters that are written (or read) and those in the external device.
- Also, because of possible translations, the number of characters written (or read) may not be the same as those in the external device.

VTC Academy

@ 2021 VTC Academy

Binary Streams

VTC Academy
Think Ahead, Beyond Boundaries

- A binary stream is a sequence of bytes with a one-to-one correspondence to those in the external device, that is, there are no character translations.
- The number of bytes written (or read) is the same as the number on the external device.
- Binary streams are a flat sequence of bytes, which do not have any flags to indicate the end of file or end of record.
- The end of file is determined by the size of the file.

VTC Academy

@ 2021 VTC Academy

Files

- A file can refer to anything from a disk file to a terminal or a printer.
- A file is associated with a stream by performing an open operation and disassociated by a close operation.
- When a program terminates normally, all files are automatically closed.
- When a program crashes, the files remain open.
- File handling in C: Using standard I/O in C using `fprintf()`, `fscanf()`, `fread()`, `fwrite()`, `fseek()`.

@ 2021 VTC Academy

File Functions

VTC Academy
Think Ahead, Beyond Boundaries

Name	Function
`fopen()`	Opens a file
`fclose()`	Closes a file
`fputc()`	Writes a character to a file
`fgetc()`	Reads a character from a file
`fread()`	Reads from a file to a buffer
`fwrite()`	Writes from a buffer to a file
`fseek()`	Seeks a specific location in the file
`fprintf()`	Operates like `printf()`, but on a file
`fscanf()`	Operates like `scanf()`, but on a file
`feof()`	Returns true if end-of-file is reached

VTC Academy

@ 2021 VTC Academy

File Functions

VTC Academy
Think Ahead, Beyond Boundaries

Name	Function
`ferror()`	Returns true if an error has occurred
`rewind()`	Resets the file position locator to the beginning of the file
`remove()`	Erases a file
`fflush()`	Writes data from internal buffers to a specified file

VTC Academy

@ 2021 VTC Academy

File Pointer

- A file pointer is essential for reading or writing files.
- It is a pointer to a structure that contains the file name, current position of the file, whether the file is being read or written, and whether any errors or the end of the file have occurred.
- The definitions obtained from `<stdio.h>` include a structure declaration called `FILE`.
- The only declaration needed for a file pointer is:

```
FILE *fp;
```


@ 2021 VTC Academy

Types of Files

VTC Academy
Think Ahead, Beyond Boundaries

- When dealing with files, there are two types of files you should know about:
  - Text files: Text files are the normal .txt files. You can easily create text files using any simple text editors such as Notepad.
  - Binary files: Binary files are mostly the .bin files in your computer. Instead of storing data in plain text, they store it in the binary form (0's and 1's).

VTC Academy

@ 2021 VTC Academy

File Operations

- In C, you can perform four major operations on files, either text or binary:
  1. Creating a new file
  2. Opening an existing file
  3. Reading from and writing information to a file
  4. Closing a file

VTC Academy

@ 2021 VTC Academy

Working with Text File

- Open File:
  - Using `fopen()` function
- File Manipulation:
  - Using `fgetc()`, `fgets()` functions read from file
  - Using `fputc()`, `fputs()` functions write to file
- Close File:
  - Using `fclose()` function

VTC Academy

@ 2021 VTC Academy

Opening a Text File

- The `fopen()` function opens a stream for use and links a file with that stream.
- The `fopen()` function returns a file pointer associated with the file.
- The prototype for `fopen()` function is:

```
FILE *fopen(const char *filename, const char *mode);
```

VTC Academy

@ 2021 VTC Academy

Opening a Text File

VTC Academy
Think Ahead, Beyond Boundaries

Mode	Meaning
`r`	Open a text file for reading
`w`	Create a text file for writing
`a`	Append to a text file
`r+`	Open a text file for read/write
`w+`	Create a text file for read/write
`a+f`	Append or create a text file for read/write

VTC Academy

@ 2021 VTC Academy

Reading a Text File Example

VTC Academy
Think Ahead, Beyond Boundaries

- Using `fgetc()` function:

```c
#include <stdio.h>

int main()
{
    FILE *f;

    f = fopen("main.c", "r");

    if(f != NULL){
        char ch;
        while((ch=fgetc(f))!=-1){
            putchar(ch);
        }

        fclose(f);
    } else {
        printf("Can't read text file.");
    }

    return 0;
}
```

VTC Academy

@ 2021 VTC Academy

Reading a Text File Example

VTC Academy
Think Ahead, Beyond Boundaries

- Using `fgets()` function:

```c
#include <stdio.h>
#include <string.h>

int main()
{
    FILE *f;

    f = fopen("main.c", "r");

    if(f != NULL){
        char str[81];
        while((fgets(str, 80, f))!=NULL){
            str[strlen(str)-1] = '\0';
            puts(str);
        }

        fclose(f);
    } else {
        printf("Can't read text file.");
    }

    return 0;
}
```

VTC Academy

@ 2021 VTC Academy

Working with Binary File

- Open File:
  - Using `fopen()` function
- File Manipulation:
  - Using `fread()` functions read from file
  - Using `fwrite()` functions write to file
- Close File:
  - Using `fclose()` function

VTC Academy

@ 2021 VTC Academy

Opening a Binary File

- To open binary file, you also use `fopen()` function.
- The `fopen()` function returns a file pointer associated with the file.
- The prototype for `fopen()` function is:

```
FILE *fopen(const char *filename, const char  *mode);
```

- Different from opening a text file is adding parameter “b” in opening mode.

VTC Academy

@ 2021 VTC Academy

Opening a Binary File

VTC Academy
Think Ahead, Beyond Boundaries

Mode	Meaning
`rb`	Open a binary file for reading
`wb`	Create a binary file for writing
`ab`	Append to a binary file
`r+b`	Open a binary file for read/write
`w+b`	Create a binary file for read/write
`a+b`	Append a binary file for read/write

VTC Academy

@ 2021 VTC Academy

Working with Binary File Example

VTC Academy
Think Ahead, Beyond Boundaries

```c
#include <stdio.h>
#include <string.h>

typedef struct{
    char isbn[15];
    char title[51];
    char author[51];
    float price;
}Books;

void printBook(Books book);
Books getBook();
void getString(char *str, int length);
void printLine();
void printTitle();
//function to read Books array from file
int readBooksFromFile(Books *lstBooks, int *pCount, const char *fileName);
//function to write Books array with count elements to file
int writeBooksToFile(Books *lstBooks, int count, const char *fileName);
```

VTC Academy

@ 2021 VTC Academy

Working with Binary File Example

VTC Academy
Think Ahead, Beyond Boundaries

```
int main( ) {
    Books books[100];
    int i, count = 0;

    //read data from file
    readBooksFromFile(books, &count, "books.dat");

    printf("Input book %d:\n", count+1);
    books[count] = getBook();
    count++;

    printTitle();
    for(i=0; i<count; i++){
        printBook(books[i]);
    }
    printLine();

    //write data to file
    writeBooksToFile(books, count, "books.dat");

    return 0;
}
```

VTC Academy

@ 2021 VTC Academy

Working with Binary File Example

VTC Academy
Think Ahead, Beyond Boundaries

```
int main( ) {
    Books books[100];
    int i, count = 0;

    //read data from file
    readBooksFromFile(books, &count, "books.dat");

    printf("Input book %d:\n", count+1);
    books[count] = getBook();
    count++;

    printTitle();
    for(i=0; i<count; i++){
        printBook(books[i]);
    }
    printLine();

    //write data to file
    writeBooksToFile(books, count, "books.dat");

    return 0;
}
```

VTC Academy

@ 2021 VTC Academy

Working with Binary File Example

VTC Academy
Think Ahead, Beyond Boundaries

```
Books getBook(){
    Books book;
    printf("Input Book isbn: ");
    getString(book.isbn, 14);
    printf("Input Book title: ");
    getString(book.title, 50);
    printf("Input Book author: ");
    getString(book.author, 50);
    printf("Input Book price: ");
    scanf("%f", &book.price);
    return book;
}
void printBook(Books book){
    printf( "| %-14s | %-26s | %-20s | %6.2f |\n",
            book.isbn, book.title, book.author, book.price);
}
void printLine(){
    printf( "+-%-14s-+-%-26s-+-%-20s-+-%-6s-+\n", "--------------",
        "--------------------------", "--------------------", "------");
}
```


@ 2021 VTC Academy

Working with Binary File Example

VTC Academy
Think Ahead, Beyond Boundaries

```
void printTitle(){
    printLine();
    printf( "| %-14s | %-26s | %-20s | %-6s |\n", "isbn", "Title",
        "Author", "Price");
    printLine();
}
void getString(char *str, int length){
    //clear keyboard buffer on UNIX
    fseek(stdin, 0, SEEK_END);
    //clear keyboard buffer on Windows
    fflush(stdin);
    //input string
    fgets(str, length, stdin);
    str[strlen(str)-1] = '\0';
    //clear keyboard buffer on UNIX
    fseek(stdin, 0, SEEK_END);
    //clear keyboard buffer on Windows
    fflush(stdin);
}
```

VTC Academy

@ 2021 VTC Academy

Working with Binary File Example

VTC Academy
Think Ahead, Beyond Boundaries

```
int readBooksFromFile(Books *lstBooks, int *pCount, const char *fileName){
    FILE *f;
    int result = 0;
    f = fopen(fileName, "rb");
    if(f!=NULL){
        //read a int is array element number
        fread(pCount, sizeof(int), 1, f);
        if(*pCount > 0){
            //read data in file to lstBooks
            fread(lstBooks, sizeof(Books), *pCount, f);
            result = 1;
        }
        //close file
        fclose(f);
    }
    return result;
}
```

VTC Academy

@ 2021 VTC Academy

Working with Binary File Example

VTC Academy
Think Ahead, Beyond Boundaries

```c
int writeBooksToFile(Books *lstBooks, int count, const char *fileName){
    FILE *f;
    int result = 0;
    f = fopen(fileName, "wb");
    if(f!=NULL){
        //write count to file
        fwrite(&count, sizeof(int), 1, f);
        //write lstBooks to file
        fwrite(lstBooks, sizeof(Books), count, f);
        //close file
        fclose(f);
        result = 1;
    }
    return result;
}
```

VTC Academy

@ 2021 VTC Academy

feof() Function

- The function `feof()` returns true if the end of the file has been reached, otherwise it returns false (0).
- This function is used while reading binary data.
- The prototype for `feof()` function is:

```
int feof(FILE *fp);
```

VTC Academy

@ 2021 VTC Academy

Erasing File

- The `remove()` function erases a specified file.
- The `remove()` function in C can be used to delete a file. The function returns 0 if files is deleted successfully, other returns a non-zero value.
- The prototype for `remove()` function is:

```
int remove (char *fileName);
```

VTC Academy

@ 2021 VTC Academy

Getting Data Using fseek() Fuctions

VTC Academy
Think Ahead, Beyond Boundaries

- If you have many records inside a file and need to access a record at a specific position, you need to loop through all the records before it to get the record. This will waste a lot of memory and operation time.
- An easier way to get to the required data can be achieved using `fseek()`.
- Syntax:

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

- Example:

```
// Moves the cursor to the end of the file
fseek(fptr, -sizeof(struct threeNum), SEEK_END);
```


@ 2021 VTC Academy

Flushing Streams

VTC Academy Think Ahead, Beyond Boundaries

- The `fflush()` function flushes out the buffer depending upon the file type.
- A file opened for read will have its input buffer cleared, while a file opened for write will have its output buffer written to the files.
- The prototype for `fflush()` function is:

```
int fflush(FILE *fp);
```

- The `fflush()` function, with a null, flushes all files opened for output.

VTC Academy

@ 2021 VTC Academy

Current Active Pointer

VTC Academy
Think Ahead, Beyond Boundaries

- A pointer is maintained in the FILE structure to keep track of the position where I/O operations take place.
- Whenever a character is read from or written to the stream, the current active pointer (known as curp) is advanced.
- The current location of the current active pointer can be found with the help of the `ftell()` function.
- The prototype for `ftell()` function is:

```
long int ftell(FILE *fp);
```

VTC Academy

@ 2021 VTC Academy

fprintf() & fscanf() Function

VTC Academy
Think Ahead, Beyond Boundaries

- The buffered I/O system includes fprintf() and fscanf() functions that are similar to `printf()` and `scanf()` except that they operate with files.
- The prototypes of are:

```
int fprintf(FILE * fp, const char *control_string,..);
int fscanf(FILE *fp, const char *control_string,...);
```

- The `fprintf()` and `fscanf()` though the easiest, are not always the most efficient.
- Extra overhead is incurred with each call, since the data is written in formatted ASCII data instead of binary format.
- If speed or file size is a concern, `fread()` and `fwrite()` are a better choice.

@ 2021 VTC Academy

Summary

- All I/O operations in C are carried out using functions from the standard library.
- I/O in C is unique because data may be transferred in its internal binary representation or in a human-readable text format.
- The C file system works with a wide variety of devices including printers, disk drives, tape drives and terminals.
- There are two types of streams - the text and binary streams.
- A file can refer to anything from a disk file to a terminal or a printer.
- A file pointer is essential for reading or writing files.

@ 2021 VTC Academy

VTC Academy
Think Ahead, Beyond Boundaries

VTC Academy

@ 2021 VTC Academy