The University Of Sheffield.

Further Data Types and File Processing

Data structure, strings and how to save and read data files

A WORLD TOP 100 UNIVERSITY

The University Of Sheffield.

Learning Outcomes

- Use data structures and enumerations
- Understand and use dynamic memory allocation
- Using typedef and understanding the concept of abstract data types
- Work with string variables
- Read/write data from/to ascii text files
- Read/write data from to binary files
- Use data structures to read/write record files

The University Of Sheffield.

Review

- Pointers contain variable addresses
  - Address operator &
  - Dereferencing operator *
- Functions call by value and call by reference
- The array variable is a pointer whose value is the address of the first element of the array
- A string is a pointer to an array of characters
- Multidimensional array is essentially an array of pointer arrays.
- Treat with great care…. Memory leaks!

Function Pointers

- Pointer to function
  - Address of the function in memory
  - Starting address of the code
- Quick sort function has prototype (see 2 slides later)
  - **int (**fncompare*)(const void *, const void *)**
  - fncompare is function pointer
  - Tells quicksort to expect pointer to a function receiving 2 pointers to void
- Note without the brackets around *fncompare the declaration becomes a declaration of a function

The University Of Sheffield.

Function Declaration Using Function Pointers

- Declare mysortfunc as
  - mysortfunc(int *data, int size, int (*compare)(void * , void *))
- Call mysortfunc in the following way
  - mysortfunc(data,size,ascending)
  - mysortfunc(data,size,descending)
- Ascending and descending are functions declared as
  - int ascending(void *a, void *b)
  - int descending(void *a, void *b)

The University Of Sheffield.

Array of pointers to functions

- Declare functions
  - void function1(int);
  - void function2(int);
  - void function3(int);
- Declare function pointer array
  - void (*f[3])(int)={function1,function2, function3};
- Called as follows
  - (*f[choice])(myintergerinput);
- Choice and myintegerinput are both integers

A WORLD TOP 100 UNIVERSITY

The University Of Sheffield.

Using the quicksort function

- Implementation of quick sort algorithm
- Qsort function in <stdlib.h>
- Features
  - Pointer to void *
  - Pointer to a function

The University Of Sheffield.

Syntax for the qsort function

- Implementation of the quicksort algorithm to sort the *num* elements of an array pointed by *base*
  - each element has the specified *width* in bytes
  - method used to compare each pair of elements is provided by the caller to this function with *fncompare* parameter (a function called one or more times during the sort process).
- **void qsort (void *** *base*, **size_t** *num*, **size_t** *width*, **int (****fncompare*)**(const void *, const void *));**

The University Of Sheffield.

Example Using qsort

```
/* qsort example */
#include <stdio.h>
#include <stdlib.h>
 int values[] = { 40, 10, 100, 90, 20, 25 };
int compare (const void * a, const void * b){ return ( *(int*)a - *(int*)b ); }

int main ()
{
  int * pItem;
  int n;
  qsort (values, 6, sizeof(int), compare);
   for (n=0; n<6; n++)
  {
  printf ("%d ",values[n]);
   }
  return 0;
}
```

The University Of Sheffield.

Data Types and Structures

- Features for representing data and aggregations of different data types.
  - struct - structures,
  - typedef - type definitions,
  - enum - enumerations
  - union

The University Of Sheffield.

Data Structures

- Arrays and structures are similar
  - pointers to an area of memory that
  - aggregates a collection of data.
- Array
  - All of the elements are of the same type and are numbered.
- Structure
  - Each element or field has its own name and data type.

The University Of Sheffield.

Format of a data structure

```
struct structure-name {
  field-type field-name; /*description*/
  field-type field-name; /*description*/
  .......
}  variable-name;
```


A WORLD TOP 100 UNIVERSITY

The University Of Sheffield.

Declaring Structures and Accessing Fields

- *struct structure-name variable-name;*
- A pointer to a structure
  - *struct structure-name *ptr-variable-name;*
- Accessing a field in a structure
- *variable-name.field-name*
- For a pointer to a structure a field is accessed using the indirection operator ->

  ptr-variable-name->field-name

A WORLD TOP 100 UNIVERSITY

Structure Example

```
struct node {
  char *name;
  char *processor;
       int  num_procs;
  };
```


A WORLD TOP 100 UNIVERSITY

The University Of Sheffield.

Declaring and Initialising Structures

```
struct node n1;
struct node *n1ptr;
n1.name="Titania";
n1.processor ="Ultra Sparc III Cu";
n1.num_procs = 80;
n1ptr = &n1;
```


A WORLD TOP 100 UNIVERSITY

Accessing Structure Data

- Direct access

```
printf("The node %s has %d %s processors\n",
        n1.name, n1.num_procs, n1.processor);
```

- Access using a pointer

```
printf("The node %s has %d %s processors\n",
        n1ptr->name, n1ptr->num_procs, n1ptr->processor);
```

- Dereferencing a pointer

```
printf("The node %s has %d %s processors\n",
        (*n1ptr).name, (*n1ptr).num_procs, (*n1ptr).processor);
```

The University Of Sheffield.

Type Definitions

typedef float vec[3];

Defines an array of 3 float variables a particle position may then be defined using:

vec particlepos;

Defined structure types

typedef struct structure-name mystruct;

mystruct mystructvar;

The University Of Sheffield.

- Compile and run the following programs

Program array.c initialising and using arrays with pointers

Program bubblesort.c is a bubble sort example, using call by reference to manipulate data passed into a function

Program arrayref.c uses pointer notation to manipulate arrays

Modify the bubblesort program to use the qsort routine

The University Of Sheffield.

Practical Example

- Compile and run the following programs
  - Numerical Differentiation
    - 2 and four point methods
  - Numerical Integration
    - Trapezium method
    - Simpsons rule (includes lagrange interpolation function)

A WORLD TOP 100 UNIVERSITY

The University Of Sheffield.

Characters and Strings

- A single character defined using the char variable type
- Character constant is an int value enclosed by single quotes
- E.g. ‘a’ represents the integer value of the character a
- A string is a series of characters
- String, string literals and string constants enclosed by double quotes

A WORLD TOP 100 UNIVERSITY

The University Of Sheffield.

Defining Characters and Strings

- Declaring and assigning a single character

```
char c='a';
```

- Strings are arrays of characters
- A pointer to the first character in the array
- The last element of the string character array is the null termination charcter '\0'
- '\0' Denotes theend of a string

A WORLD TOP 100 UNIVERSITY

The University Of Sheffield.

Defining Strings

- char node[]=“iceberg”;
- char *nodeptr=“iceberg”;
- char nodename[180];
- For the first two definitions the null termination is added by the compiler

The University Of Sheffield.

Formatted String Input and Output

sprintf(char *s, const char *format, …..)

Equivalent to printf with the exception that its output is stored in the array s specified in the sprintf function. The prototype for sscanf is ;

sscanf(char *s, const char *format, …).

Equivalent to scanf reads input from the string s specified in the sscanf function.

The University Of Sheffield.

Examples: sprintf and scanf

```
char node[20], s2[80];
char s1[] ="Titania 3.78 7";
float fload, floadout;
int nusers, nusersout;
/*Using sscanf to read data from a string*/
sscanf(s1, "%s%f%d", node, &floadout, &nusersout);
sprintf(s2, "%s %f %d", node, fload, nusers);
```

The University Of Sheffield.

Functions for Character Manipulation

- library **ctype.h**
- **isdigit**, **isalpha**, **islower**, **isupper**, **toupper, tolower** and **isspace**.
- These functions can be used to perform conversions on a single character or for testing that a character is of a certain type.

A WORLD TOP 100 UNIVERSITY

The University Of Sheffield.

String Conversion Functions

- String conversion functions from the general utilities library **stdlib**
- convert strings to float, int long int, double, long, and unsigned long data types respectively.
- **atof**, **atoi**, **atol**, **strtod**, **strtol**, **strtoul**

A WORLD TOP 100 UNIVERSITY

The University Of Sheffield.

String Manipulation

- The string handling library **string.h**
- provides a range of string manipulation functions for copying, concatenating, comparison, tokenizing and for identifying the occurrence and positions of particular characters in a string.
- E.g. **strcpy**, **strlen**, **strcmp** and **strtok**.
- See the examples

The University Of Sheffield.

Practice:String Copy and Concatenation

```
// Create some strings
char szStringX[20] = "Hello, World";
char szStringY[20];
char szStringZ[20] = "Hi, World";
// Copy szStringX to szStringY
strcpy(szStringY,szStringX);
printf("szStringX is %s\n",szStringX);
printf("szStringY is %s\n",szStringY);
// Compare szStringX and szStringY to see if they are the same
if(!strcmp(szStringX,szStringY))
printf("Strings are the same\n");
else printf("Strings are NOT the same\n");
// Compare szStringX and szStringY to see if they are the same
if(!strcmp(szStringX,szStringZ))
        printf("Strings are the same\n");
else printf("Strings are NOT the same\n");
```

The University Of Sheffield.

File Processing

- file as a sequential stream of bytes with each file terminated by an end-of file marker
- When a file is opened a stream is associated with the file
- Streams open during program execution
  - stdin
  - stdout
  - stderr

The University Of Sheffield.

Sequential File Management

- Streams
  - channels of communication between files and programs.
- Range of functions for streaming data to files
  - fprintf
  - fscanf
  - fgetc
  - fputc

A WORLD TOP 100 UNIVERSITY

The University Of Sheffield.

Opening a File

- When opening a file it is necessary to declare a variable that will be used to reference that file, the standard library provides the FILE structure.
- So a pointer to a FILE is declared using:
  - FILE *myfile;
- File opened using the function fopen
  - returns a pointer to the opened file

The University Of Sheffield.

Using the fopen function

```
if((myfile=fopen("myfilename", "w"))==NULL)
  printf("The file could not be opened!\n");
else
{
  file was opened and is read or written here
}
```


A WORLD TOP 100 UNIVERSITY

The University Of Sheffield.

File open modes

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

The University Of Sheffield.

Writing data using fprintf

- *fprintf(fileptr, “format specifiers”, data list);*
  - fprintf(mfptr, "%6d %20s %6d\n", iRunid, sName, iNode);
- Closing the file
  - fclose(mfptr);

A WORLD TOP 100 UNIVERSITY

The University Of Sheffield.

Reading data using fscanf

- *fscanf(fileptr, “format specifiers”, data list);*

```
while(!feof(mfptr))
{
  printf("%6d %20s %6d\n", sim.id,
sim.name, sim.node);
  fscanf(mfptr, "%d%s%d", &sim.id,
sim.name, &sim.node);
}
```


A WORLD TOP 100 UNIVERSITY

The University Of Sheffield.

A WORLD TOP 100 UNIVERSITY

The University Of Sheffield.

Demonstration

- Method for solving 1st order ODEs with well defined BC's
- Shooting Method
  - Compile and run the code
  - startshooting.c
- Method for solving 1st order ODEs with well defined BC's
- Shooting Method
  - Compile and run the code
  - startshooting.c

The University Of Sheffield.

Exercise

- Adapt the program startshooting.c to read the input parameters from an input file.
- Adapt the program so that it reads the guess q froom the command line
- To read parameters from the command line we use the parameters argc and argv passed into the main function
- Use the following line to convert the command line parameter
- Hint look at vecdp.c in the functions folder

```
If(argc>1)
        q=atof(argv[1]);
```


A WORLD TOP 100 UNIVERSITY

The University Of Sheffield.

Random Access Files

- Transaction processing systems
- Individual records of same length accessed at random
- fwrite
  - Write fixed number of bytes to a file
- fread
  - Read fixed number of bytes from a file

The University Of Sheffield.

Data Declaration

- Example data structure
  - struct mydata { int index; float data;};
- Typical declaration
  - struct mydata blankdata={0, 3.141};

A WORLD TOP 100 UNIVERSITY

The University Of Sheffield.

fwrite example call

- fwrite(&blankdata, sizeof(struct mydata), 1, fileptr)
  - Write data structure myblankdata
  - Specify correct field size
  - Specify number of data items to write (in this case 1)
  - Provide a valid pointer to the file that is opened for writing

The University Of Sheffield.

Fread - Example call

- fread(&blankdata, sizeof(struct mydata), 1, fileptr)
  - Read data structure myblankdata
  - Specify correct field size
  - Specify number of data items to read (in this case 1)
  - Provide a valid pointer to the file that is opened for reading

The University Of Sheffield.

fseek

- fseek sets file pointer to specific position in file
  - int fseek(FILE *stream, long int offset, int whence)
- Offset is number of bytes from location whence
- Whence has one of three values
- SEEK_SET (beginning of file)
- SEEK_CUR (current location)
- SEEK_END (end of file)
- Example call
  - fseek(myfileptr, sizeof(struct mydata)*(index-1),SEEK_SET);

The University Of Sheffield.

Practice

- Study and run the program fileio.c in the extras directory

The University Of Sheffield.

Further Sessions

- Building Applications using Make
- From C to C++
- Boost your programming using the standard template libraries