The University Of Sheffield.

Further Data Types and File Processing

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

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!

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;
```

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

Structure Example

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

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;
```

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)

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

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

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.