Uses and applications of loops, arrays, structures and unions;

Uses of loops;

Uses of loops include cycling through values, adding sums of numbers, repeating functions, and many other things. Two major categories of loop uses are producing output and searching for information.

LOOP USE — PRODUCING OUTPUT

You can reduce a long series of repetitive instructions down to one instruction by using a loop.

For instance, a long series of multiplying numbers can be rewritten and condensed into a For loop. You are able to complete the exact task you want to without explicitly writing repetitive instructions line after line after line.

LOOP USE — SEARCHING FOR INFORMATION

Having an algorithm check through items in a list is incredibly useful. Let’s say that you have a list and you want to find a specific piece of information contained in that list.

Applications of Arrays:

Array is used for different verities of applications. Array is used to store the data or values of

same data type. Below are the some of the applications of array s;

A. Stores Elements of Same Data Type

Array is used to store the number of elements belonging to same data type.

Similarly if we declare the character array then it can hold only character. So in short character

array can store character variables while floating array stores only floating numbers.

B. Array Used for maintaining multiple variable names using single name

Suppose we need to store 5 roll numbers of students then without declaration of array we need

to declare following –

int roll1, roll2, roll3, roll4, roll5;

1. Now in order to get roll number of first student we need to access roll1.

2. Guess if we need to store roll numbers of 100 students then what will be the procedure.

3. Maintaining all the variables and remembering all these things is very difficult.

Consider the Array int roll[5]; Here we are using array which can store multiple values and we

have to remember just single variable name.

C. Array can be used for Sorting Elements

We can store elements to be sorted in an array and then by using different sorting technique we

can sort the elements.

Different Sorting Techniques are:

1. Bubble Sort

2. Insertion Sort

3. Selection Sort

4. Bucket Sort

D. Array can perform Matrix Operation

Matrix operations can be performed using the array. We can use 2-D array to store the matrix.

Matrix can be multi dimensional.

E. Array can be used in CPU Scheduling

CPU Scheduling is generally managed by Queue. Queue can be managed and implemented

using the array. Array may be allocated dynamically i.e at run time. [Animation will Explain more

about Round Robin Scheduling Algorithm | Video Animation]

F. Array can be used in Recursive Function

When the function calls another function or the same function again then the current values are

stores onto the stack and those values will be retrieving when control comes back. This is

similar operation like stack.

Arrays as Function arguments:

Passing array to function:

Array can be passed to function by two ways:

1. Pass Entire array

2. Pass Array element by element

1. Pass Entire array

* Here entire array can be passed as a argument to function.
* Function gets complete access to the original array.
* While passing entire array address of first element is passed to function, any changes made
* inside function, directly affects the Original value.
* Function Passing method : “Pass by Address“

2. Pass Array element by element

* Here individual elements are passed to function as argument.
* Duplicate carbon copy of Original variable is passed to function.
* So any changes made inside function do not affect the original value.
* Function doesn‟t get complete access to the original array element.
* Function passing method is “Pass by Value“

3. Passing entire array to function:

* Parameter Passing Scheme : Pass by Reference
* Pass name of array as function parameter.
* Name contains the base address i.e. ( Address of 0th element )
* Array values are updated in function.
* Values are reflected inside main function also.

3.Uses and applications of structures;

* Organizing Related Data:

Structures allow you to group together variables of different data types that are related to each other. For example, you can define a structure to represent a person with attributes like name, age, and address.

* Creating Complex Data Types:

By combining different data types, structures enable you to create complex data types that suit your specific needs. This is especially useful when dealing with entities that have multiple attributes.

* Passing Structs to Functions:

You can pass structures as parameters to functions, allowing you to work with and manipulate multiple related data in a more organized way.

* Arrays of Structures:

Structures can be used to create arrays, where each element of the array is a structure. This is beneficial when dealing with collections of related data.

Structures are commonly used in file I/O operations. You can define a structure that represents the format of data in a file, allowing you to read and write data in a more organized manner.

* Database Programming:

In database programming, structures are often used to represent records in a database table. Each structure variable corresponds to a row in the table, and the structure members represent the attributes of the record.

* Dynamic Memory Allocation:

Structures can be dynamically allocated using pointers, providing a way to manage memory efficiently for complex data structures.

Uses and applications of unions;

* Memory Optimization:

Unions are useful when you need to represent a data type that can be one of several types, but not all at once. By sharing memory among the different members, unions help conserve memory compared to using separate variables or structures for each type.

* Implementing Variants:

Unions are commonly used to implement variants or tagged unions, where a single variable can represent different types of data.

* Bitfields:

Unions can be used in conjunction with bitfields to efficiently pack multiple values into a single storage unit. This is particularly useful when working with hardware registers or protocol headers.

* Type-Punning:

Unions are sometimes used for type-punning, a technique where you interpret the bits of one type as another. This is useful in low-level programming and certain optimization scenarios.

* Handling Different Data Representations:

Unions can be employed to handle different representations of the same data. For example, you might use a union to represent a value as both a raw binary and a more human-readable format.

* Overlaying Data Structures:

Unions allow overlaying different data structures on top of the same memory space. This can be useful in scenarios where you want to interpret the same memory location differently at different times.

* Interoperability with Other Languages:

Unions can be used to facilitate interoperability with other languages or systems where data layouts might be different. This is especially relevant in systems programming and interfacing with hardware.