Compound data types

Arrays, Pointers, data structures,unions, Dynamic memory

Arrays

- An array is a series of elements of the same type placed in contiguous memory locations that can be individually referenced by adding an index to a unique identifier.
- Format : ***type name[Elements];***
- ***E.g int car[5];***
- Initialisation: int a[3]={1,2,4}; or int a[]={1,2,4}
- Access elements by index: **name[index]**

Arrays

- E.g int a[4]={1,2,35,6}; //a[0]=1,a[3]=6
- Multidimensional arrays: type name[elements][elements]…[elements];

e.g. double vehicle[2][3];

- Arrays in functions

e.g. int car[4];

int add(int car1[]); //declaration

add(car1); //call for the array

Pointers

- A variable which stores a reference to another variable
- Pointers are said to "point to" the variable whose reference they store
- E.g int *p; //declare a pointer

  a=10;

  p=&a; //reference //p=memory address of a

  a=*p; //deference //a =10, value pointed to by pointer p

Pointers

- Discussion
  - int b=8;
  - int *a;
  - a= &b; //assu;me address of b to 100;
  - a++; // address incremented by size of int to 104
  - *(a+4) is equivalent to a[4]
  - *a++ is equivalent to *(p++)   ++ has high precedence over the *
  - int x[4];
  - a=x;

Pointers

- Pointers to pointers

```
Int **a; //*(*a);
```

Dynamic memory

- Create and destroy memory during runtime
- new creates new memory dynamically
- delete destroys/frees the allocated memory
- pointer = new type; //allocate
- delete pointer; //free memory
- pointer =new type[elements]; //allocate
- delete []pointer; //free memory

Data structures

- A data structure is a group of data elements grouped together under one name.
- These data elements, known as *members*, can have different types and different lengths.
- Data structures are declaration format:

Data structures

```
struct structure_name {
    member_type1 member_name1;
    member_type2 member_name2;
    member_type3 member_name3;
    } object_names;
```