Design and Analysis of Algorithms

REPORT

Topic : Vector Implementation in C

NAME : Vaibhav V Pawar
SRN : PES1201701131
CLASS : B

Vector :

Vector are same as dynamic array which has the ability to resize itself when the element is inserted or deleted from the array . Vectors are stored in contiguous storage so that they can be accessed and traversed using iterators . In vector data is inserted at end and insertion may take different time as they may be a need of extending an array . Removing the element from the array may take constant time or linear time depending on the implementation , If resizing is required for deletion then linear time and if resizing not required constant time

Some of the functions of Vectors are :

- vector::begin() :
  - Returns the iterator pointer to first element in the vector .
- vector::end() :
  - Returns the iterator pointer to last element in the vector .
- Vector::front() :
  - Return the starting element in the vector .
- vector::back() :
  - Return the last element in the vector .
- vector::size() :
  - Returns the size of vector .
- vector::push_back() :
  - Used to add element at last in vector
- .vector::pop_back():
  - Used to delete element from last in vector .
- vector::max_element() :
  - Returns an iterator pointer to max element in vector .
- vector::min_element() :
  - Returns an iterator pointer to min element in vector

.And many more functions are available in vector in c++. C++ has a library called vector which has all the functions in it

Now let us talk about the project and functions available in it .

Structure of Vector :

There are two structures one for integers and one for characters .
The one for integers is :

```
typedef struct vector_int {

    int capacity;
    int elements;
    int *arr;
    void (*push_back)(struct vector_int* vec , int ele);
    bool (*pop_back)(struct vector_int* vec);
    int (*size)(struct vector_int* vec);

} vector_int;
```

And one for characters is :

```
typedef struct vector_char {

    int capacity;
    int elements;
    char *arr;
    void (*push_back)(struct vector_char* vec , char ele);
    bool (*pop_back)(struct vector_char* vec);
    int (*size)(struct vector_char* vec);

} vector_char;
```

1) The variable capacity is used to know the actual capacity of the vector . When we insert a new element into the vector , there are two possibilities, one ,number of elements in the vector are less than the actual capacity of vector . second , the vector is full and need to be resized ie capacity is equal to number of elements .

2) The variable elements is used to know the number of elements present in vector

3) The “ arr ” is an array of integer in an integer vector and array of characters in character vector .

4) The variable push_back is a function pointer variable . Which helps to add a new element in vector with the help of some other functions .

5) The pop_back variable is same as push_back but used to delete last element from the vector .

6) The variable " size " is used to get the size of a vector .

Adding Element to vector :

In vector elements are added at the end .
There are two scenario for adding an element in to an vector .

Scenario one : When capacity is more then the number of elements present in vector. At that time the element is added at the last place in *constant time* as we already know the previous numbers of elements present in vector , using that we will get to know where to place new element in vector .

Scenario two: When the vector is fully filled and a new element should be added . At that time The vector capacity is increased by the factor of two ie the resultant capacity of the vector will be twice the previous capacity . Which will give room to many many new elements . In this case time complexity of adding element at last will be *directly proportional to number of elements* as the array has to be reallocated and all the elements from previous vector has to be copied to new array . which may consume time.

In the project there are two functions to add element in vector one is for integer and another for character . And both these functions are assigned to function pointers present in vector structure in "vector_init " function ,so there is no need of calling them with different names .
In some of the cases the increasing factor my change depending upon their convenience . It can be 1.5 time or 2 time .

Ex : consider v and vc are vector integer and vector character respectively , now for adding element in vector we would do
v.push_back(&v,integer_number)
vc.push_back(&vc , character)

Deleting last element from vector :

Deleting element in vector is a *constant time* operation . As we have all the required details of number of elements available in the vector we can just reduce the number of elements variable in vector so that while traversing we will not reach that place .

Deleting can be done in another way . That is , as we go on deleting elements if we find a situation where number of elements are less than half the capacity at that point of time we can reduce the capacity of vector by half so that there should not be any unwanted usage of memory. This would be again *linearly dependent on number of elements* in vector

Getting the Beginning and end elements iterator pointer :

This function is just *constant time* operations as we need only the beginning element iterator pointer . Beginning as constant as we already know where the data is in the structure we can just return the value .
And for the iterator pointer of the last element is also constant time as we know the number of elements present in vector and we also know the beginning iterator , so simply we can reach that place and return the iterator pointer .

Getting the First and Last values in vector :

It is similar to that of getting beginning and end iterator pointer , but it will not return iterator pointer instead it return the values present in first and last position of the vector .

Getting the max_element and min_element in vector :

These are the functions which will give us the min and max element iterator in the vector . These both are the linear in nature ie their time complexity is *linearly dependent of number of elements* in vector . It is as simple as finding min and max element in an array.

In these functions the user has the ability to find min and max between some range of element like max_element of starting five elements . The user have to send an iterator pointer of beginning and ending point to find min or max in the given range . If the user gives the range which is larger than number of elements that are available in the vector at that time max or min of the full array elements are found .

Implementation :

There are two header files one file “ vector.h ”contains all the functions and structure required for integers and the another file “ vector_char.h ” contains all the functions and structure required for characters to implement vector .

In “vector.h” file there is an import of “vector_char.h” file ie a header file is importing another header file so that the code should be clean and neat .

vec_imp .c is a file which contain all the functions required to perform all the operations mentioned .

An init function is present so that it can assign all the necessary functions names to function pointer and all the values to some constant like, capacity at the beginning is equal to 1 and number of elements it equal to zero .

The implementation file contains all the functions related to integer anwell as characters .

Lastly , there is a client file in which the user will write a program using vectors by importing the vector.h file and the user can use all the available functions present. After creating every vector variable the user have to call init function for every variable so that all the necessary assignments can be done at beginning .