**Recursive Solutions**

1. **Design Document**

**Introduction**

Recursion is an important problem solving approach that is an alternative to iteration. Recursion is implemented as a method that calls itself to solve subtasks. During the recursive call the values of the local fields of the method are placed on the method stack until the subtask performed by a recursive call is completed. This computer program performs 3 different task using recursive solutions.

**Data structure**

The program uses a template class that can adapt to the users need. The integer array, ‘items’, holds each integers randomly generated from the computer. This array can hold up to ten words. Meanwhile, the char array ‘word’, holds char read from the input command prompt for permutation and palindrome purpose. The char ‘select’, holds what the user selects from the 3 choices presented.

**Functions**

The Program has one class and an interface class. In the ‘recursive’ class, I have used 9 functions including the constructor. The ‘factorial()’ function takes an integer and returns the factorial computation of the number. The ‘permutation()’ function takes in an array, an integer and the size of the array for recursive computation of transformation.

‘output()’ is a void function used to print out the array sent to it with its size. ‘Palindrome()’ on the other hand takes a char array from the user and checks for the array consistency on either direction. The ‘swap()’ void functions is used to swap template data variables in array. ‘kth()’ array is used to find the smallest kth integer from an array filled with integer numbers with the use of the last function ‘partition()’ for rearranging the array with the first array pivot number.

**The main program**

The main program starts with introducing what the program is intended to be used. Latter, it creates a char object using the class ‘recursive’ with the name ‘rec.’ The program later introduces the main menu where the user can select choices from 1-4. Entering other chars or numbers other than 1-4 results error and the program shows ‘Invalid input, Try again.’ These menus includes:

1. Permutations: this can be achieved with entering number 1. Later, I used ‘switch-case’ statement to call ‘permutations ().’
2. Palindrome: this can be achieved by selecting number 2. Later, I used ‘switch-case’ statement to call ‘palindrome ().’
3. Finding kth smallest value: this can be achieved by entering number 3. Later, I used ‘switch-case’ statement to call ‘kth().’
4. Quit the program: this can be achieved by entering number 4. Later, I used ‘switch-case’ statement to display ‘Good bye’ message and used while loop to stop repeating the loop.
5. **Code list**

**Palindrom\_Permutation\_recursion.cpp: main source**

#include <iostream>

#include <string>

#include "recursiveC.cpp"

using namespace std;

int main()

{

cout << " Recursive Algorithms\n" << " \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n" << endl;

char select = ' ';

const int maxLength = 10;

char word[maxLength];

int items[maxLength];

recursive<char> rec;

cout << "1) Permutations of a word\n"

<< "2) Check if a Palindrome\n"

<< "3) The kth smallest value\n"

<< "4) Exit ";

do

{

cout << "\n\n Choice: ";

cin >> select;

string wordS;

switch (select)

{

case '1':

cout << "\n Enter a sequence of characters:";

cin >> wordS;

while (wordS.length() > maxLength)

{

wordS = "";

cout << "\n The length of the string exceeds the array limit of 10. \n Please reenter a sequence of characters:";

cin >> wordS;

}

for (int i = 0; i < wordS.length(); i++)

word[i] = wordS[i];

rec.permutation(word, 0, wordS.length());

break;

case '2':

cout << "\n Enter a sequence of characters:";

cin >> wordS;

while (wordS.length() > maxLength)

{

wordS = "";

cout << "\n The length of the string exceeds the array limit of 10. \nplease re enter a sequence of characters:";

cin >> wordS;

}

for (int i = 0; i < wordS.length(); i++)

word[i] = wordS[i];

rec.palindrome(word, 0, wordS.length());

break;

case '3':

int k;

cout << "\n Randomly generated numbers: \n =>";

for (int i = 0; i < 10; i++)

{

items[i] = 1 + rand() % 100; // generage number from 1 to 100

cout <<items[i] << " ";

}

cout << "\n Type in the value of k: ";

cin >> k;

rec.kth(items, 0, 10, k);

break;

case '4':

cout << "\nGood bye";

break;

default:

cout << " Invalid input, Try again\n";

break;

}

cin.get();

} while (select != '4');

cin.get();

return 0;

}

**recursiveInterface.h: header file**

#ifndef RECURSIVEINTERFACE

#define RECURSIVEINTERFACE

using namespace std;

template<class ItemType>

class recursiveInterface

{

public:

/\*\* fins the factorial

@param find factorial of numb

@return final factorial of numb.\*/

virtual int factorial(int numb) = 0;

/\*\* find strings

@param arr is the array

@param curr the starting point of the array

@param size size of the string\*/

virtual void permutation(ItemType\* arr, int curr, int size) = 0;

/\*\* outputs the array

@param arr is the array

@param curr the starting point of the array

@param size size of the string\*/

virtual void output(ItemType\* arr, int curr, int size) = 0;

/\*\* check for palindrome

@param arr is the array

@param curr the starting point of the array

@param size size of the string\*/

virtual void palindrome(ItemType\* arr, int curr, int size) = 0;

/\*\* change the places

@param fir pointer of first address

@param sec pointer of second address\*/

virtual void swap(ItemType\* fir, ItemType\* sec) = 0;

/\*\* find kth

@param A is the array

@param start the starting point of the array

@param end max size

@param k kth smallest number \*/

virtual void kth(int A[], int start, int end, int K) = 0;

/\*\* find kth

@param A is the array

@param start the starting point of the array

@param end max size

@return partition intger \*/

virtual int partition(int A[], int start, int end) = 0;

};

#endif

**recursive.h: header file**

#ifndef \_RECURSIVE

#define \_RECURSIVE

//#include "stdafx.h"

#include "recursiveInterface.h"

#include <string>

using namespace std;

template<class ItemType>

class recursive : public recursiveInterface<ItemType>

{

public:

recursive();

int factorial(int numb);

void permutation(ItemType\* arr, int curr, int size);

void output(ItemType\* arr, int curr, int size);

void palindrome(ItemType\* arr, int curr, int size);

void swap(ItemType\* fir, ItemType\* sec);

void kth(int A[], int start, int end, int K);

int partition(int A[], int start, int end);

};

#endif

**recursive.cpp: source file**

#include <string>

#include <stdlib.h>

#include "recursiveH.h"

template<class ItemType>

recursive<ItemType>::recursive()

{} // end default constructor

template<class ItemType>

int recursive<ItemType>::factorial(int numb){

if (numb == 0)

return 1;

else

return numb\*factorial(numb - 1);

}

template<class ItemType>

void recursive<ItemType>::swap(ItemType\* fir, ItemType\* sec)

{

char temp = \*fir;

\*fir = \*sec;

\*sec = temp;

}

template<class ItemType>

void recursive<ItemType>::permutation(ItemType\* arr, int curr, int size)

{

if (curr == size - 1)

{

cout << " "; // give tab before the array

output(arr, 0, size);

}

else

{

for (int i = curr; i<size; i++)

{

swap(&arr[curr], &arr[i]);

permutation(arr, curr + 1, size);

swap(&arr[curr], &arr[i]);

}

}

}

template<class ItemType>

void recursive<ItemType>::output(ItemType\* arr, int curr, int size)

{

if (curr >= size)

cout << " ";

else {

cout << arr[curr];

output(arr, curr + 1, size);

}

}

template<class ItemType>

void recursive<ItemType>::palindrome(ItemType\* arr, int curr, int size){

if (curr >= size / 2)

cout << " The word you typed is palindrome";

else

{

if (arr[curr] == arr[size - 1 - curr])

palindrome(arr, curr + 1, size);

else

cout << " The word you typed is not palindrome";

}

}

template<class ItemType>

void recursive<ItemType>::kth(int A[], int start, int end, int K){

int part;

if (start > K || end < K)

{

cout << " "<<K << " is not between " << start << " and " << end<<". Try again";

return;

}

else if (start < end) {

part = partition(A, start, end);

if (part == K - 1){

cout <<" "<<K <<"th smallest element: " << A[part];

}

if (part>K - 1){

kth(A, start, part, K);

}

else{

kth(A, part + 1, end, K);

}

}

return;

}

template<class ItemType>

int recursive<ItemType>::partition(int A[], int start, int end){

int i = start + 1;

int j = i;

int pivot = start; // setting the first element as the pivot

for (; i<end; i++){

if (A[i]<A[pivot]){

int temp = A[i];

A[i]= A[j];

A[j] = temp;

j++;

}

}

if (j <= end){

int temp = A[pivot];

A[pivot] = A[j - 1];

A[j - 1] = temp;

}

return j - 1;

}

1. **User Document**

This program has an easy to use procedure, but it has certain constraints that must be followed for successful usage. This program is used to do 3 important recursive solutions which are finding permutations, checking for palindrome and the kth smallest values; reading the following instructions could be very much of helpful.

Follow this instructions to run the programs:

* To compile the program, simply enter: Palindrom\_Permutation\_recursion.cpp
* To run the program, type in: a.out in the command window
* Once the program starts running input a number from 1-4 to do a specific job you want the program to do.
* Entering numbers or any other characters other than the one mentioned gives out ‘Invalid input, Try again’ error message and asks the user to reenter a number between 1 and 4 again.
* Entering 1 would let the program know that the user wants to find permutations of the strings that the user typed in, while entering 2 makes the program check whether the string entered palindromes. Inputting 3 would make the program generate 10 integers that would be displayed for the user to choose the kth smallest number.
* To quit the program input 4

1. **Test Data Plan**

|  |  |  |
| --- | --- | --- |
| Valid input Values |  |  |
|  | Input: chars 1,2,3,4 for main menu selection. | Does respective choice |
|  | Input: any strings for inserting chars to the array | It will insert chars to the array if it doesn’t exceed. |
| Boundary values |  |  |
|  | Max array size | 10 |
|  |  |  |
| Invalid input values |  |  |
|  | Input: more than 10 characters as a sequence. | The length of the string exceeds the array limit of 10. please reenter a sequence of characters: |
|  | Input: other characters in place of given choices for the main menu. | Error Message: ‘Invalid input, Try again.’ |

1. **Summary**

This program uses two header and two source files for computing and completing the task. The program asks the user for varieties of functionalities including permutation, palindrome and finding the kth smallest number from the array.

Usually, recursion involves a function calling itself. While it may not seem like much on the surface, recursion allows us to write elegant solutions to problems that may otherwise be very difficult to program. The functions used in this project are made with recursive solutions in mind.

Using recursive functions were very helpful as it has certainly helped me to solve and write complex algorithms.

The program could be extended in several ways that would make it more useful for large projects by utilizing permanent storing systems. This can be achieved using a text file or similar data extensions by utilizing libraries like ‘ifstream’ and ‘ofstream.’ Besides utilizing a permanent storage system, we can also use GUI to make it easier and simpler for users.