\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Computer Programming Lab

CEN-392

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Practical Exam

Code :-

#include <iostream>

using namespace std;

int Length(char str[])

{

    int len = 0;

    while (str[len] != '\0')

        len++;

    return len;

}

void Replace(char txt[], char find[], char replace[], int indx)

{

    int len1, len2, len3;

    len1 = Length(txt);

    len2 = Length(find);

    len3 = Length(replace);

    int i;

    for (i = indx; i < len1 && txt[i + len2] != '\0'; i++) // Removing The Find String

    {

        txt[i] = txt[i + len2];

    }

    txt[i] = '\0';

    char temp[200];

    len1 = Length(txt);

    for (i = 0; i < len1; i++) // Copying The String To The Temp From indx->end

    {

        temp[i] = txt[i + indx];

    }

    temp[i] = '\0';

    for (i = 0; i < len3; i++) // now adding the replace string from indx

    {

        txt[indx + i] = replace[i];

    }

    indx += i;

    int len4 = Length(temp);

    for (i = 0; i < len4; i++) // now we add the temp string after the replace string

    {

        txt[indx + i] = temp[i];

    }

    txt[indx + i] = '\0';

}

void Find(char txt[], char find[], char replace[], bool Multi)

{

    if (Multi)

        cout << "Multi Replace Selected...\n\n";

    else

        cout << "Single Replace Selected...\n\n";

    fflush(stdin);

    cout << "Enter The Word To Find : ";

    cin.getline(find, 200);

    fflush(stdin);

    cout << "Enter The Word To Replace : ";

    cin.getline(replace, 200);

    int len1, len2;

    len1 = Length(txt);

    len2 = Length(find);

    int i, j;

    bool check = false;

    for (i = 0; i <= len1 - len2; i++)

    {

        for (j = 0; j < len2; j++)

        {

            if (txt[i + j] != find[j])

                break;

        }

        if (j == len2)

        {

            check = true;

            if (Multi)

            {

                Replace(txt, find, replace, i);

            }

            else

            {

                Replace(txt, find, replace, i);

                break;

            }

            len1 = Length(txt);

            len2 = Length(find);

        }

    }

    if (!check)

    {

        cout << "\nWord Not Found In The Text!\n"

             << endl;

    }

    else

    {

        cout << "\nThe Modified String :\n";

        cout << txt << endl;

    }

}

void Input\_Again(char txt[], char find[], char replace[])

{

    cout << "Input Again...\n\n";

    cout << "Enter The Text : \n";

    fflush(stdin);

    cin.getline(txt, 200);

}

void Add\_Bars()

{

    cout << "-----------------------------------------------------------------------" << endl;

}

void Menu()

{

    cout << "\_\_\_\_Operations\_\_\_\_\_" << endl;

    cout << "1.Single Replace" << endl;

    cout << "2.Multiple Replace" << endl;

    cout << "3.Input Again" << endl;

    cout << "4.Exit" << endl;

    cout << "Enter Your Choice : ";

}

bool Options(char txt[], char find[], char replace[])

{

    int opt;

    fflush(stdin);

    cin >> opt;

    Add\_Bars();

    switch (opt)

    {

    case 1:

        Find(txt, find, replace, 0);

        break;

    case 2:

        Find(txt, find, replace, 1);

        break;

    case 3:

        Input\_Again(txt, find, replace);

        break;

    case 4:

        return 0;

    default:

        cout << "Incorrect Input!\nTry Again!" << endl;

        break;

    }

    Add\_Bars();

    return 1;

}

int main()

{

    system("cls");

    cout << "\_\_\_Vicky\_Gupta\_20BCS070\_\_\_\n\n";

    char txt[200], find[200], replace[200];

    cout << "Enter The Text : \n";

    cin.getline(txt, 200);

    cout << "\n\n";

    while (true)

    {

        Menu();

        if (!Options(txt, find, replace))

            break;

    }

    cout << "Exiting..." << endl;

    Add\_Bars();

    return 0;

}

Output :-

Text

Description automatically generated

Text

Description automatically generated