**XII Computer Science**

**Practical File**

**Name :** Varad Mahashabde

**Class :** XII **Sec :** A

**Roll no.:** 29

1. Guessing Game

#include <iostream>

#include <iomanip>

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

using namespace std;

int getDigitSum(int x, int base) {

int sum = 0;

while (x > 0) {

sum += x % base;

x /= base;

}

return sum;

}

int main () {

time\_t k;

srand( unsigned(time(&k)) );

// cout << (unsigned int)(-1) << "\n\n";

// for (int i = 0; i < 255; ++i)

// cout << rand() << endl;

int guess, luckys[3];

bool won = false;

do {

cout << "Enter your guess : "; cin >> guess;

if (guess < 2 || guess > 19)

cout << "ERR : INPUT OUT OF BOUNDS (2 - 19)\n";

} while (guess < 2 || guess > 19);

for (int i = 0; i < 3; ++i)

luckys[i] = (199 - 101) \* float(rand()) / 0x7fff + 101;

cout << "The lucky numbers are : \n\t";

for (int i = 0 ; i < 3; ++i) cout << setw(8) << luckys[i];

for (int i = 0; i < 3; ++i) {

luckys[i] = getDigitSum(luckys[i], 10);

won |= luckys[i] == guess;

}

cout << "\nThe digit sums are : \n\t";

for (int i = 0 ; i < 3; ++i) cout << setw(8) << luckys[i];

if (won)

cout << "\nYES YOU ARE THE WINNERRRRRR!!!!\nnow get out\n";

else

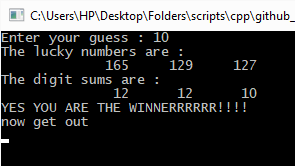
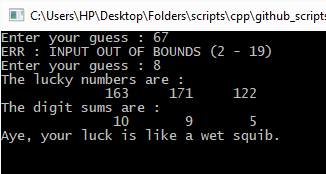
cout << "\nAye, your luck is like a wet squib.";

while (getchar() != '\n');

getchar();

}

Output :



1. Fibonacci

#include <iostream>

#include <stdio.h>

using namespace std;

unsigned long long fibonacci(int n) {

long long f = 0, s = 1, t;

if (n < 3) return (n - 1);

for (int i = n - 2; i > 0; --i) {

t = f + s;

f = s;

s = t;

}

return t;

}

int main() {

unsigned int terms = 0;

cout << "No. of terms? "; cin >> terms;

for (int i = 1; i <= terms; ++i)

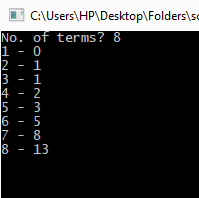
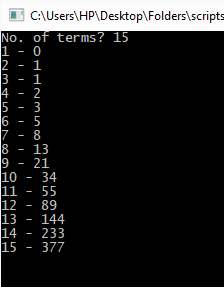
cout << i << " - " << fibonacci(i) << endl;

while (getchar() != '\n');

getchar();

}

Output :

1. Series

#include <iostream>

#include <stdio.h>

#include <iomanip>

using namespace std;

long fact(int n) {

long res = 1;

for (int i = n; i > 1; ++i)

res \*= i;

cout << res;

}

long double pow(long double b ,int e) {

long double res = 1;

if (e < 0)

for (int i = -e; i > 0; ++i)

res /= b;

else if (e > 0)

for (int i = e; i > 0; ++i)

res /= b;

return res;

}

int main() {

long double x, sum = 0; unsigned int n;

cout << "Give value of x : "; cin >> x;

cout << "No. of terms? "; cin >> n;

for (int i = 2\*n - 1; i > 1; i -= 2)

sum += (((i - 1)/2) % 2)?(-1):(1) \* pow(x, i) / fact(i);

sum += 1;

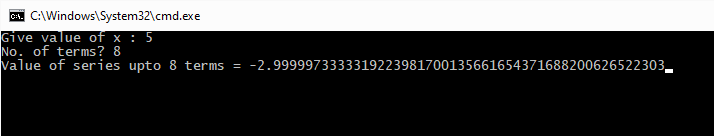
cout << "Value of series upto " << n << " terms = " << setprecision(50) << sum;

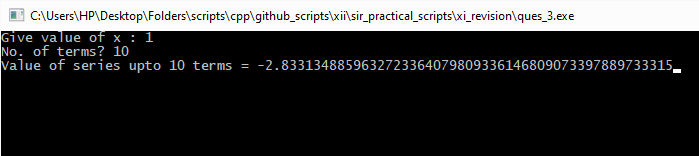
while (getchar() != '\n');

getchar();

}

Output :





1. Triangles

#include <iostream>

#include <stdio.h>

using namespace std;

int main () {

unsigned int lines, choice;

cout << "Enter no. of lines : "; cin >> lines;

do {

cout << "Enter type of pyramid :\nGUIDE :\nNumber Pyramid - 0\nUpside-down Isosceles Pyramid - 1\nHollow Diamond - 2\nBottom-filled Diamond - 3\n\tChoice? ";

cin >> choice;

if (choice >= 4)

cout << "ERR : INPUT OUT OF BOUNDS(0-3)\n";

} while (choice >= 4);

switch (choice) {

case 0:

for (int i = 0; i < lines; ++i) {

for (int j = lines - i; j > 0; --j)

cout << ' ';

for (int j = 1; j < i + 2; ++j)

cout << j;

for (int j = i; j > 0; --j)

cout << j;

cout << '\n';

}

break;

case 1:

for (int i = lines; i > 0; --i) {

for (int j = lines - i + 1; j > 0; --j)

cout << ' ';

for (int j = 2 \* i - 1; j > 0; --j)

cout << '\*';

cout << '\n';

}

break;

case 2:

for (int i = 0; i < lines/2; ++i) {

for (int j =(lines + 2)/2 - i; j > 0; --j)

cout << ' ';

cout << '\*';

for (int j = 2\*i - 1; j > 0; j--)

cout << ' ';

if (i) cout << '\*';

cout << '\n';

}

for (int i = lines/2; i >= 0; --i) {

for (int j =(lines + 2)/2 - i; j > 0; --j)

cout << ' ';

cout << '\*';

for (int j = 2\*i - 1; j > 0; j--)

cout << ' ';

if (i) cout << '\*';

cout << '\n';

}

break;

case 3:

for (int i = 0; i < lines/2; ++i) {

for (int j =(lines + 2)/2 - i; j > 0; --j)

cout << ' ';

cout << '\*';

for (int j = 2\*i - 1; j > 0; j--)

cout << ' ';

if (i) cout << '\*';

cout << '\n';

}

for (int i = lines/2; i >= 0; --i) {

for (int j =(lines + 2)/2 - i; j > 0; --j)

cout << ' ';

for (int j = 2\*i + 1; j > 0; --j)

cout << '\*';

cout << '\n';

}

break;

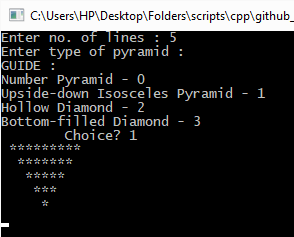
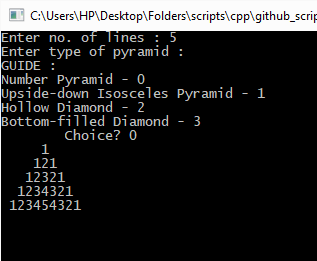
}

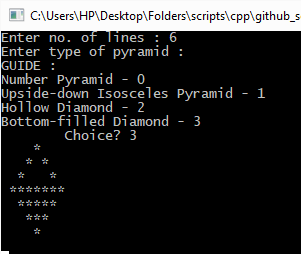
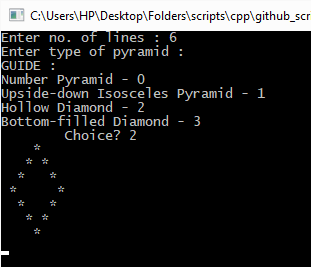
while (getchar() != '\n');

getchar();

}

Output :





1. Letter count

#include <iostream>

#include <stdio.h>

using namespace std;

bool isUpper (char ch) {

return ch >='A' and ch <= 'Z';

}

bool isLower (char ch) {

return ch >='a' and ch <= 'z';

}

int main() {

char str[1000];

cout << "Give your string :\n\t";

// cin << flush;

cin.getline(str, 1000, '\n');

int upper, lower, non\_alpha, words;

upper = lower = non\_alpha = words = 0;

for (int i = 0; str[i] != '\0'; ++i) {

if (isUpper(str[i]))

upper++;

else if (isLower(str[i]))

lower++;

else {

if (str[i] == ' ') {

words++;

if (str[i + 1] == '.')

i++; non\_alpha++;

}

non\_alpha++;

}

}

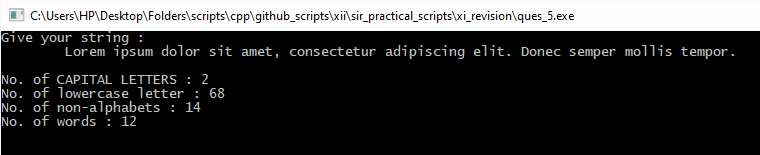
words++;

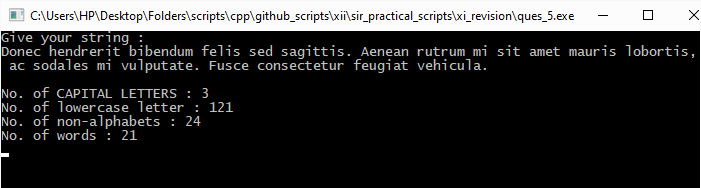
cout << "\nNo. of CAPITAL LETTERS : " << upper << "\nNo. of lowercase letter : " << lower << "\nNo. of non-alphabets : " << non\_alpha << "\nNo. of words : " << words << endl;

getchar();

}

Output :





1. cASE sWITCH

#include <iostream>

#include <stdio.h>

using namespace std;

bool isUpper (char ch) {

return ch >='A' and ch <= 'Z';

}

bool isLower (char ch) {

return ch >='a' and ch <= 'z';

}

char toUpper (char ch) {

if (isLower(ch))

return (ch - 0b00100000);

return ch;

}

char toLower (char ch) {

if (isUpper(ch))

return (ch + 0b00100000);

return ch;

}

int main() {

char str[1000];

cout << "Give your string :\n\t";

// cin << flush;

cin.getline(str, 1000, '\n');

for (int i = 0; str[i] != '\0'; ++i) {

if (isUpper(str[i]))

str[i] = toLower(str[i]);

else if (isLower(str[i]))

str[i] = toUpper(str[i]);

}

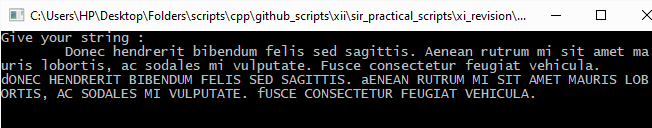
cout << str;

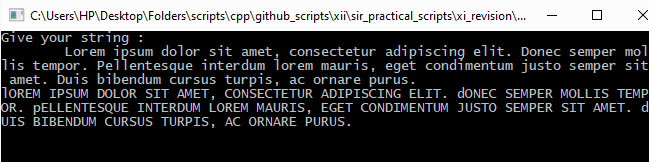
while (getchar() != '\n');

getchar();

}

Output :





1. Palindrome

#include <iostream>

#include <limits>

#include <string.h>

using namespace std;

bool isPalindrome(const char\* str) {

bool is\_palin = true;

unsigned len = strlen(str);

for (int i = len/2 - 1; i > -1 and is\_palin; --i)

is\_palin &= (str[i] == str[len - i - 1]);

return is\_palin;

}

int main() {

char \*str, \*concat\_str;

const char\* yes\_palin = "Hello Palindrome ";

const char\* no\_palin = "Hello Non-palindrome ";

unsigned len;

cout << "Enter string length : "; cin >> len;

str = new char[len + 1];

concat\_str = new char[len + 22];

cout << "Enter string : \n\t";

cin.ignore(std::numeric\_limits<streamsize>::max(), '\n');

cin.clear();

cin.getline(str, len + 1, '\n');

if (isPalindrome(str))

strcopy(concat\_str, strcat(yes\_palin, str));

else

strcopy(concat\_str, strcat(no\_palin, str));

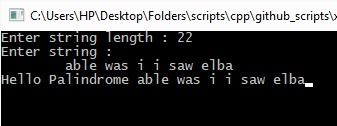
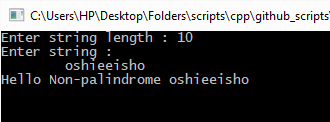
cout << concat\_str;

while (getchar() != '\n');

getchar();

}

Output :



1. Half-Half Array

#include <iostream>

using namespace std;

void swap(int arr[], int i\_1, int i\_2) {

int temp = arr[i\_1];

arr[i\_1] = arr[i\_2];

arr[i\_2] = temp;

}

int main() {

unsigned int arr\_size;

cout << "Array size? "; cin >> arr\_size;

int arr[arr\_size];

cout << "Array : "; for (int i = 0; i < arr\_size; ++i) cin >> arr[i];

for (int i = 0, j = arr\_size/2 + arr\_size % 2; j < arr\_size; i++, j++) {\

swap(arr, i , j);

}

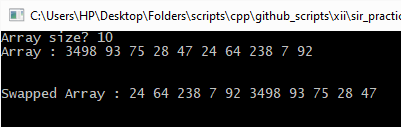
cout << "\n\nSwapped Array : "; for (int i = 0; i < arr\_size; ++i) cout << arr[i] << ' ';

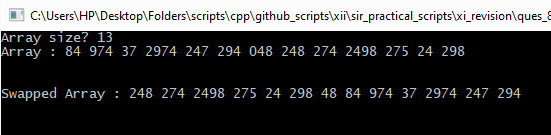
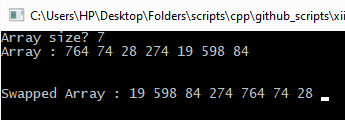
while (getchar() != '\n');

getchar();

}

Output :





1. De-duplication

#include <iostream>

using namespace std;

int main() {

int size\_arr, no\_of\_duplicates = 0;

cout << "Give size of array : "; cin >> size\_arr;

int arr[size\_arr];

bool arr\_tracker[size\_arr];

for (int i = 0; i < size\_arr; ++i)

arr\_tracker[i] = true;

// Get input

cout << "Enter array : ";

for (int i = 0; i < size\_arr; ++i)

cin >> arr[i];

cout << "\nOriginal Array : ";

for (int i = 0; i < size\_arr; ++i)

cout << arr[i] << ' ';

// Mark and remove all duplicate entries

for (int i = 0; i < size\_arr; ++i) {

int in\_focus = arr[i];

for (int j = i + 1; j < size\_arr; ++j)

if (arr[j] == in\_focus) {

arr\_tracker[j] = false;

no\_of\_duplicates++;

}

}

for (int i = 0, j; i < size\_arr - 1; ++i) {

if (arr\_tracker[i] == false) {

// find next non-duplicate entry

j = i + 1;

while (j < size\_arr) {

if(arr\_tracker[j] == true)

break;

else j++;

}

// No need to continue if is all duplcates till the end of the array

if (j == size\_arr) break;

// Swap values

int temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

// Swap duplication status

arr\_tracker[i] = true;

arr\_tracker[j] = false;

}

}

cout << "\nFinal Array : ";

for (int i = 0; i < size\_arr - no\_of\_duplicates; ++i)

cout << arr[i] << ' ';

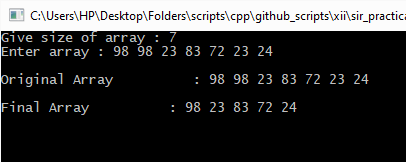
cout << '\n';

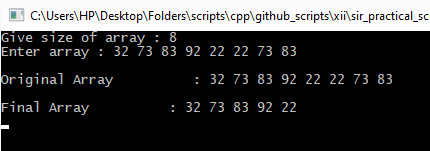
while(getchar() != '\n');

getchar();

}

Output :





1. Matrix-math

#include <iostream>

#include <iomanip>

#include <stdio.h>

using namespace std;

int main() {

int m, n;

cout << "Size of matrix (m x n)? "; cin >> m >> n;

int matrix[m][n];

cout << "Enter matrix : \n";

for (int i = 0; i < m; ++i) {

cout << '\t';

for (int j = 0; j < n; ++j)

cin >> matrix[i][j];

}

cout << "\nColumn sums : \n\t";

for (int j = 0; j < n; ++j) {

int col\_sum = 0;

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

col\_sum += matrix[i][j];

cout << setw(10) << col\_sum << ' ';

}

cout << "\nRow sums : \n";

for (int i = 0; i < m; ++i) {

int row\_sum = 0;

for (int j = 0; j < n; ++j)

row\_sum += matrix[i][j];

cout << '\t' << setw(10) << row\_sum << '\n';

}

cout << "\nMain diagonal sum : ";

int diag\_sum = 0;

for (int i = 0, j = 0; i < m and j < n; i++, j++)

diag\_sum += matrix[i][j];

cout << diag\_sum << '\n';

cout << "\nBack diagonal sum : ";

diag\_sum = 0;

for (int i = 0, j = n - 1; i < m and j >= 0; i++, j--)

diag\_sum += matrix[i][j];

cout << diag\_sum << '\n';

cout << "Transpose : \n";

for (int j = 0; j < n; ++j) {

cout << '\t';

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

cout << setw(10) << matrix[i][j];

cout << '\n';

}

cout << "\nUpper Triangle : \n";

for (int i = 0; i < m; ++i) {

cout << '\t';

for (int j = 0; j < n; ++j) {

if (i > j)

cout << setw(10) << '0';

else

cout << setw(10) << matrix[i][j];

}

cout << '\n';

}

cout << "\nLower Triangle : \n";

for (int i = 0; i < m; ++i) {

cout << '\t';

for (int j = 0; j < n; ++j) {

if (i < j)

cout << setw(10) << '0';

else

cout << setw(10) << matrix[i][j];

}

cout << '\n';

}

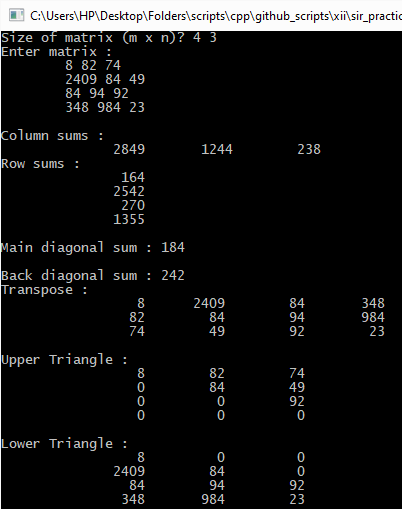
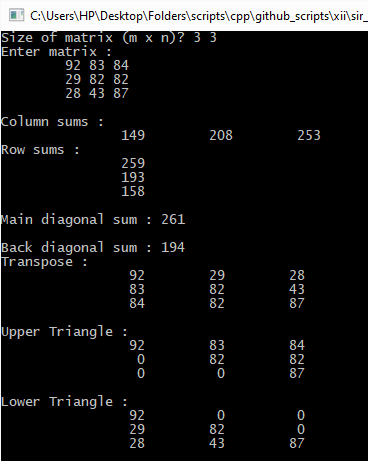
while (getchar() != '\n');

getchar();

return 0;

}

Output :



1. Temperature

#include <iostream>

using namespace std;

enum {MORNING, NOON, EVENING, NIGHT};

int main() {

float temp[3][7], min\_max\_temp[3][2] = {0, 0, 0, 0, 0, 0}, av\_temp[3] = {0,0,0}, av\_temp\_week;

cout << "Enter temperatures : \n";

for (int i = MORNING; i < NIGHT; ++i) {

cout << '\t';

for (int j = 0; j < 7; ++j)

cin >> temp[i][j];

cout << '\n';

}

for (int i = MORNING; i < NIGHT; ++i) {

min\_max\_temp[i][0] = temp[i][0];

min\_max\_temp[i][1] = temp[i][0];

}

for (int i = MORNING; i < NIGHT; ++i) {

for (int j = 0; j < 7; ++j) {

if (min\_max\_temp[i][0] > temp[i][j])

min\_max\_temp[i][0] = temp[i][j];

if (min\_max\_temp[i][1] < temp[i][j])

min\_max\_temp[i][1] = temp[i][j];

av\_temp[i] += temp[i][j];

av\_temp\_week += temp[i][j];

}

av\_temp[i] /= 7;

}

av\_temp\_week /= 7 \* 3;

cout << "Minimum morning temperature : " << min\_max\_temp[MORNING][0] << '\n';

cout << "Minimum noon temperature : " << min\_max\_temp[NOON][0] << '\n';

cout << "Minimum evening temperature : " << min\_max\_temp[EVENING][0] << '\n';

cout << "Maximum morning temperature : " << min\_max\_temp[MORNING][1] << '\n';

cout << "Maximum noon temperature : " << min\_max\_temp[NOON][1] << '\n';

cout << "Maximum evening temperature : " << min\_max\_temp[EVENING][1] << '\n';

cout << "Average morning temperature : " << av\_temp[MORNING] << '\n';

cout << "Average noon temperature : " << av\_temp[NOON] << '\n';

cout << "Average evening temperature : " << av\_temp[EVENING] << '\n';

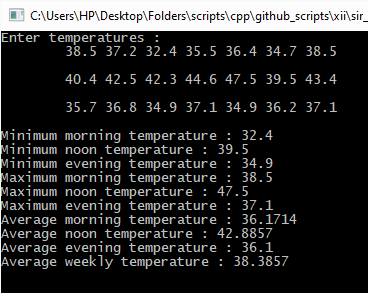
cout << "Average weekly temperature : " << av\_temp\_week << '\n';

while(getchar() != '\n');

getchar();

}

Output :



1. What the Square

#include <iostream>

#define SQR(x) x\*x

using namespace std;

int Sqr(int x) { return x \* x; }

inline int sqr(int x) { return x \* x;}

int main () {

unsigned int choice;

float n;

cout << "Welcome to Square-inator\n"

<< "========================\n";

cout << "Input number : ";

while(!(cin >> n)); // Check if cin goes into a fail state

cout << "Choose a squaring method :\n"

<< "\tMacro : 1\n"

<< "\tOutline function : 2\n"

<< "\tInline function : 3\n";

while(!(cin >> choice)); // Check if cin goes into a fail state

switch ( (choice - 1) % 3 + 1) {

case 1 :

cout << SQR(n);

break;

case 2 :

cout << Sqr(n);

break;

case 3 :

cout << sqr(n);

break;

}

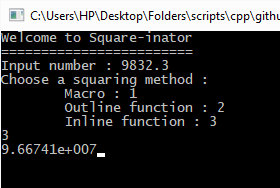
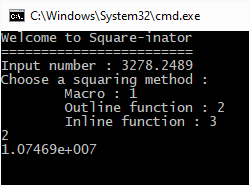
while (getchar() != '\n');

getchar();

return 0;

}

Output :



1. All Sort

#include <iostream>

#include <limits>

#include <string.h>

#include <time.h>

#include <stdlib.h>

using namespace std;

template <typename arr\_type>

void quick\_sort(arr\_type\* arr, int start, int end, bool descending = false) {

if (end - start <= 1 or start < 0) return;

int pivot\_point = end - 1; // default pivot location

int pivot\_location = start; // initial location of pivot

arr\_type temp; // for swapping

//partition array into two halves based on the pivot number

for (int i = start; i < end - 1; ++i) {

if (descending ^ (arr[i] < arr[pivot\_point]) ) {

// swap i and pivot\_location

temp = arr[pivot\_location];

arr[pivot\_location] = arr[i];

arr[i] = temp;

pivot\_location++;

}

}

// put pivot in it's place

temp = arr[pivot\_point];

arr[pivot\_point] = arr[pivot\_location];

arr[pivot\_location] = temp;

quick\_sort(arr, start, pivot\_location, descending);

quick\_sort(arr, pivot\_location + 1, end, descending);

return;

}

// template <> quick\_sort<char\*>

void quick\_sort(char\*\* arr, int start, int end, bool descending = false) {

if (end - start <= 1 or start < 0) return;

int pivot\_point = end - 1; // default pivot location

int pivot\_location = start; // initial location of pivot

char\* temp; // for swapping

//partition array into two halves based on the pivot number

for (int i = start; i < end - 1; ++i) {

if (descending ^ (strcmp(arr[i], arr[pivot\_point]) == -1) ) {

// swap i and pivot\_location

temp = arr[pivot\_location];

arr[pivot\_location] = arr[i];

arr[i] = temp;

pivot\_location++;

}

}

// put pivot in it's place

temp = arr[pivot\_point];

arr[pivot\_point] = arr[pivot\_location];

arr[pivot\_location] = temp; ;

quick\_sort(arr, start, pivot\_location, descending);

quick\_sort(arr, pivot\_location + 1, end, descending);

return;

}

int main () {

unsigned int no\_of\_elements, choice;

cout << "Give array datatype : \n"

<< "int : 1\n"

<< "float : 2\n"

<< "char : 3\n"

<< "string : 4\n";

cin >> choice;

cout << "Give no. of elements in array : "; cin >> no\_of\_elements;

choice = (choice - 1) % 4 + 1;

if (choice == 1) {

int\* arr = new int[no\_of\_elements];

cout << "Input array : ";

for (int i = 0; i < no\_of\_elements; ++i)

cin >> arr[i];

quick\_sort<int>(arr, 0, no\_of\_elements, false);

cout << "Sorted array : ";

for (int i = 0; i < no\_of\_elements; ++i)

cout << arr[i] << " ";

} else if (choice == 2) {

float\* arr = new float[no\_of\_elements];

cout << "Input array : ";

for (int i = 0; i < no\_of\_elements; ++i)

cin >> arr[i];

quick\_sort<float>(arr, 0, no\_of\_elements, false);

cout << "Sorted array : ";

for (int i = 0; i < no\_of\_elements; ++i)

cout << arr[i] << " ";

} else if (choice == 3) {

char\* arr = new char[no\_of\_elements];

cout << "Input array : ";

for (int i = 0; i < no\_of\_elements; ++i)

cin >> arr[i];

quick\_sort<char>(arr, 0, no\_of\_elements, false);

cout << "Sorted array : ";

for (int i = 0; i < no\_of\_elements; ++i)

cout << arr[i] << " ";

} else if (choice == 4) {

char\*\* arr = new char\*[no\_of\_elements];

int max\_str\_size;

cout << "Maximum length of a string : "; cin >> max\_str\_size;

for (int i = 0; i < no\_of\_elements; ++i)

arr[i] = new char[max\_str\_size + 1];

cin.ignore(std::numeric\_limits<streamsize>::max(), '\n');

cout << "Input array : ";

for (int i = 0; i < no\_of\_elements; ++i)

cin.getline(arr[i], max\_str\_size + 1, '\n');

quick\_sort(arr, 0, no\_of\_elements, false);

cout << "Sorted array : ";

for (int i = 0; i < no\_of\_elements; ++i)

cout << arr[i] << " ";

}

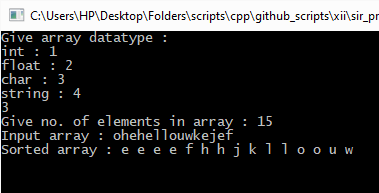
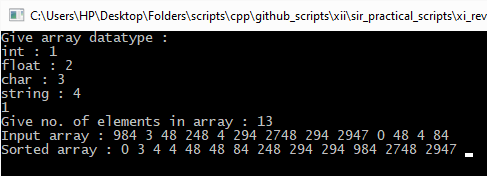
while (getchar() != '\n');

getchar();

return 0;

}

Output :



1. Full Power

#include <iostream>

#include <iomanip>

#include <limits>

using namespace std;

template <typename base\_type = double>

base\_type abs(base\_type x) {

if (x > 0)

return x;

else

return -x;

}

template <typename base\_type = double>

base\_type pow\_int(base\_type b, long long int e) {

if (e == 0 or b == 1) return 1;

base\_type num = 1, square\_base = b;

if (e < 0) square\_base = 1/b; // Takes care of inverse sign

long long int power = abs(e);

// Modular Exponentiation Algorithm

while (power > 0) {

if (power % 2)

num \*= square\_base;

square\_base \*= square\_base;

power /= 2;

}

return num;

}

template <typename base\_type = long long int>

base\_type fact(int x) {

base\_type res = 1;

for (int i = x; i > 1; --i)

res \*= i;

return res;

}

template <typename base\_type = double>

base\_type exp(base\_type x) {

base\_type a = 0, a\_old = 1;

for (int i = 0; a != a\_old; ++i) {

a\_old = a;

a += pow\_int<base\_type>(x, i) / fact<base\_type>(i);

}

return a;

}

template <typename base\_type = double>

base\_type ln(base\_type x) {

if (x <= 0) return numeric\_limits<base\_type>::quiet\_NaN();

base\_type a = 0, a\_old = 1;

for (int i = 1; a != a\_old; i+=2) {

a\_old = a;

a += pow\_int( ((x - 1)/(x + 1)), i) / i;

}

return 2\*a;

}

template <typename base\_type = double, typename exponent\_type = double>

base\_type pow(base\_type b, exponent\_type e) {

return exp(e \* ln(b));

}

template <typename base\_type = double, typename exponent\_type = double>

base\_type power(base\_type b, exponent\_type e = 1) {

return pow(b,e);

}

int main() {

long double x, n;

char pass\_exp;

cout << "Enter base and exponent : "; cin >> x >> n;

cout << "Pass exponent to function (y/n)? "; cin >> pass\_exp;

cout << "Result is ";

if (pass\_exp == 'n' or pass\_exp == 'N')

cout << setprecision(50) << std::fixed << power(x);

else

cout << setprecision(50) << std::fixed << power(x,n);

cin.ignore(std::numeric\_limits<streamsize>::max(), '\n');

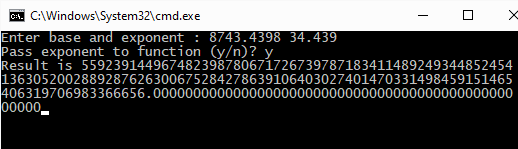
cin.clear();

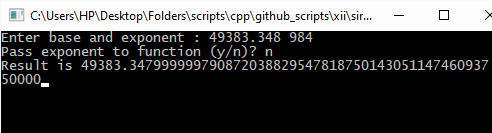
getchar();

return 0;

}

Output :





1. Fibonacci

#include <iostream>

using namespace std;

float& min(float& a, float& b) {

if (a > b)

return b;

else return a;

}

int main() {

float k, l;

cout << "Enter values for k and l : "; cin >> k >> l;

cout << "\nk = " << k;

cout << "\nl = " << l;

min(k, l) \*= (1 + 0.1);

cout << "\nk = " << k;

cout << "\nl = " << l;

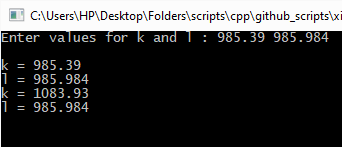
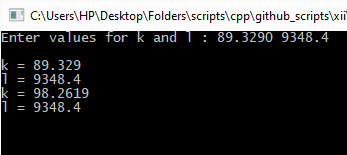
while(getchar() != '\n');

getchar();

return 0;

}

Output :



1. Assorted Students

#include <iostream>

#include <iomanip>

#include <limits>

#include <windows.h>

using namespace std;

// source - http://www.cplusplus.com/articles/4z18T05o/

#if defined \_WIN32

#include <windows.h>

void ClearScreen() {

HANDLE hStdOut;

CONSOLE\_SCREEN\_BUFFER\_INFO csbi;

DWORD count;

DWORD cellCount;

COORD homeCoords = { 0, 0 };

hStdOut = GetStdHandle( STD\_OUTPUT\_HANDLE );

if (hStdOut == INVALID\_HANDLE\_VALUE) return;

/\* Get the number of cells in the current buffer \*/

if (!GetConsoleScreenBufferInfo( hStdOut, &csbi )) return;

cellCount = csbi.dwSize.X \*csbi.dwSize.Y;

/\* Fill the entire buffer with spaces \*/

if (!FillConsoleOutputCharacter(

hStdOut,

(TCHAR) ' ',

cellCount,

homeCoords,

&count

)) return;

/\* Fill the entire buffer with the current colors and attributes \*/

if (!FillConsoleOutputAttribute(

hStdOut,

csbi.wAttributes,

cellCount,

homeCoords,

&count

)) return;

/\* Move the cursor home \*/

SetConsoleCursorPosition( hStdOut, homeCoords );

}

#elif defined (\_\_LINUX\_\_) || defined(\_\_gnu\_linux\_\_) || defined(\_\_linux\_\_) || defined (\_\_APPLE\_\_)

#include <unistd.h>

#include <term.h>

void ClearScreen() {

if (!cur\_term)

{

int result;

setupterm( NULL, STDOUT\_FILENO, &result );

if (result <= 0) return;

}

putp( tigetstr( "clear" ) );

}

#endif

void gotoxy(int x, int y) {

COORD c = { x, y };

SetConsoleCursorPosition( GetStdHandle(STD\_OUTPUT\_HANDLE) , c);

}

struct Student {

char name[20];

unsigned int marks;

};

void quickSort(Student\* arr, unsigned int start, unsigned int end, bool ascending = true) {

if (end <= start + 1 or arr == nullptr)

return;

int pivotIndex = start;

Student temp;

for (int i = start; i < end - 1; ++i) {

if (arr[i].marks < arr[end - 1].marks xor not ascending) {

temp = arr[i];

arr[i] = arr[pivotIndex];

arr[pivotIndex] = temp;

pivotIndex++;

}

}

temp = arr[pivotIndex];

arr[pivotIndex] = arr[end - 1];

arr[end - 1] = temp;

quickSort(arr, start, pivotIndex, ascending);

quickSort(arr, pivotIndex + 1, end, ascending);

return;

}

int main () {

Student stu\_arr[20];

int no\_of\_students;

bool ascending = false;

char choice;

do {

cout << "No. of students? ";

cin >> no\_of\_students;

} while (no\_of\_students > 20);

cout << "Enter student details : \n";

for (int i = 0; i < no\_of\_students; ++i) {

cout << "\tStudent " << i << " : \n";

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

cin.clear();

cout << "\t\tName : "; cin.getline(stu\_arr[i].name, 20, '\n');

cout << "\t\tAggregate marks : "; cin >> stu\_arr[i].marks;

}

// Normal code

// cout << " S.No. Name Marks\n"

// << " ----- ------------------- -----\n";

// for (int i = 0; i < no\_of\_students; ++i) {

// cout << ' ' << setw(2) << i << ". ";

// cout << setw(19) << stu\_arr[i].name << " ";

// cout << setw(4) << stu\_arr[i].marks << '\n';

// }

cout << "\n\nDisplay in ascending (y/n)?";

cin >> choice;

if (choice == 'y' or choice == 'Y') ascending = true;

quickSort(stu\_arr, 0 , no\_of\_students, ascending);

ClearScreen();

gotoxy(1, 0);

cout << "S.No. Name Marks\n";

gotoxy(1, 1);

cout << "----- ------------------- -----\n";

for (int i = 0; i < no\_of\_students; ++i) {

gotoxy(1, 2 + i);

cout << setw(2) << (i + 1) << '.';

gotoxy(9, 2 + i);

cout << setw(19) << stu\_arr[i].name;

gotoxy(30, 2 + i);

cout << setw(4) << stu\_arr[i].marks;

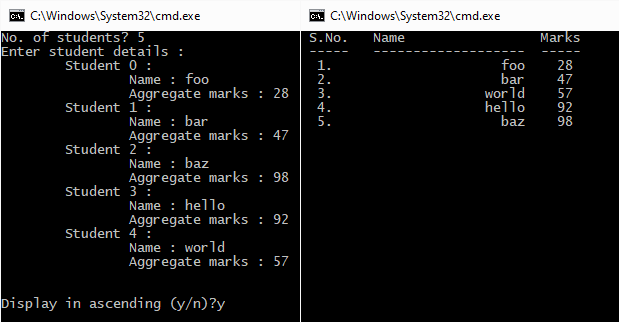
}

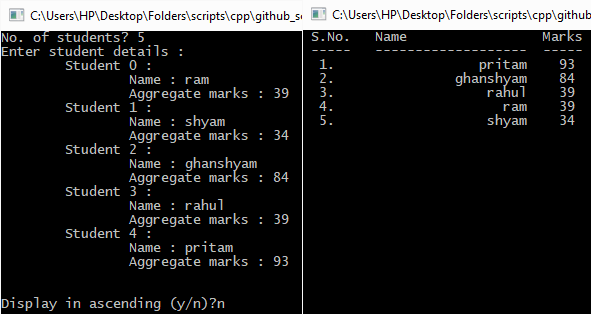
while (getchar() != '\n');

getchar();

}

Output :





1. Imperial Systems

#include <iostream>

using namespace std;

class FeetInches {

public :

int feet;

short inches;

FeetInches() {

this->feet = 0;

this->inches = 0;

}

FeetInches(int f, short i) {

this->feet = f;

this->inches = i;

this->Adjust();

}

FeetInches Adjust() {

if (this->inches > 0) {

this->feet += this->inches / 12;

this->inches %= 12;

} else {

this->feet += -1 + this->inches / 12;

this->inches = ((this->inches % 12) + 12) % 12;

}

}

FeetInches getInput(istream& input\_stream) {

while (!(input\_stream >> this->feet))

input\_stream.clear();

while (!(input\_stream >> this->inches))

input\_stream.clear();

this->Adjust();

}

FeetInches printSelf(ostream& display\_stream) {

display\_stream << this->feet << "\' " << this->inches << '\"';

}

FeetInches operator+(FeetInches f) {

return FeetInches(this->feet + f.feet, this->inches + f.inches);

}

FeetInches operator+=(FeetInches f) {

this->feet += f.feet;

this->inches += f.inches;

this->Adjust();

return \*this;

}

};

FeetInches add(FeetInches a, FeetInches b) {

return a + b;

}

int main () {

FeetInches dist1, dist2;

cout << "Enter dist1 (feet,inches) : "; dist1.getInput(cin);

cout << "Enter dist2 (feet,inches) : "; dist2.getInput(cin);

cout << "\n\ndist1 = "; dist1.printSelf(cout);

cout << "\ndist2 = "; dist2.printSelf(cout);

cout << "\ndist1 + dist2 = "; add(dist1, dist2).printSelf(cout);

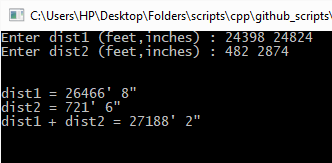
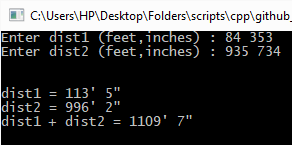
while(getchar() != '\n');

getchar();

return 0;

}

Output :



1. Streaming Students

#include <iostream>

#include <iomanip>

#include <limits>

using namespace std;

enum Stream {COMPUTER\_SCIENCE, ELECTRONICS, MECHANICAL, ELECTRICAL, CHEMICAL, CIVIL, NONE};

const char\* stream\_name(Stream stream) {

switch(stream) {

case COMPUTER\_SCIENCE :

return "Computer Science";

break;

case ELECTRONICS :

return "Electronics";

break;

case MECHANICAL :

return "Mechanical";

break;

case ELECTRICAL :

return "Electrical";

break;

case CHEMICAL :

return "Chemical";

break;

case CIVIL :

return "Civil";

break;

case NONE :

return "";

break;

}

}

class Student {

char name[30];

unsigned roll\_num;

unsigned marks[5];

Stream stream;

public :

Student() {

this->name[0] = '\0';

this->roll\_num = 0;

for (int i = 0 ; i < 5; ++i)

this->marks[i] = 0;

this->stream = NONE;

}

void getInput(istream& input\_stream, ostream& display\_stream, char\* pre\_str) {

input\_stream.ignore(std::numeric\_limits<streamsize>::max(), '\n');

input\_stream.clear();

display\_stream << pre\_str << "Name : ";

input\_stream.getline(this->name, 30, '\n');

display\_stream << pre\_str << "Roll no. : ";

while (!(input\_stream >> this->roll\_num)) {

input\_stream.ignore(1);

input\_stream.clear();

}

display\_stream << pre\_str << "Marks : ";

for (int i = 0; i < 5; ++i) {

while (!(input\_stream >> this->marks[i])) {

input\_stream.ignore(1);

input\_stream.clear();

}

}

this->assignStream();

}

void printSelf(ostream& display\_stream, int name\_space\_num, int roll\_space\_num, int mark\_space\_num, int stream\_space\_num) {

display\_stream << setw(roll\_space\_num) << this->roll\_num << '.';

display\_stream << setw(name\_space\_num) << this->name;

for (int i = 0; i < 5; ++i)

display\_stream << setw(mark\_space\_num) << this->marks[i];

cout << setw(stream\_space\_num) << stream\_name(this->stream);

}

unsigned long total() {

unsigned long sum = 0;

for (int i = 0; i < 5; ++i)

sum += this->marks[i];

return sum;

}

float average() {

return this->total() / float(5);

}

Stream assignStream() {

float average\_marks = this->average();

if (average\_marks < 70)

this->stream = NONE;

else if (average\_marks < 75)

this->stream = CIVIL;

else if (average\_marks < 80)

this->stream = CHEMICAL;

else if (average\_marks < 85)

this->stream = ELECTRICAL;

else if (average\_marks < 90)

this->stream = MECHANICAL;

else if (average\_marks < 95)

this->stream = ELECTRONICS;

else

this->stream = COMPUTER\_SCIENCE;

return this->stream;

}

};

int main() {

Student stu\_arr[20];

unsigned short no\_of\_students;

cout << "No. of students? ";

do {

while (!(cin >> no\_of\_students)) cin.clear();

if (no\_of\_students > 20)

cout << "ERR : input out of range (0<, <=20)";

} while (no\_of\_students > 20);

cout << "Enter student details : ";

for (int i = 0; i < no\_of\_students; ++i) {

char format\_string[] = "\t\t";

cout << "\n\tStudent " << (i + 1) << " : \n";

stu\_arr[i].getInput(cin, cout, format\_string);

}

cout << "Student Data : \n";

cout << " Sr.no. Name Phys Chem Math Engl Comp Stream \n"

<< " ------ ----------------------------- ---- ---- ---- ---- ---- ----------------\n";

for (int i = 0; i < no\_of\_students; ++i) {

stu\_arr[i].printSelf(cout, 31, 6, 6, 18);

cout << '\n';

}

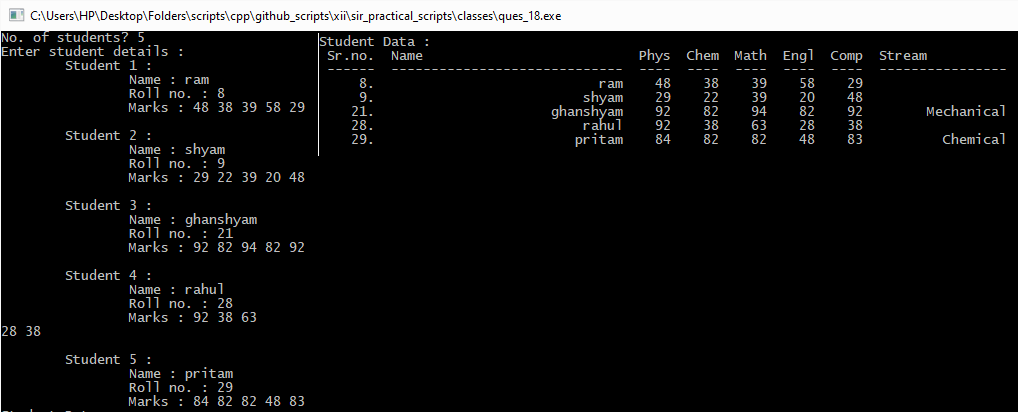
while (getchar() != '\n');

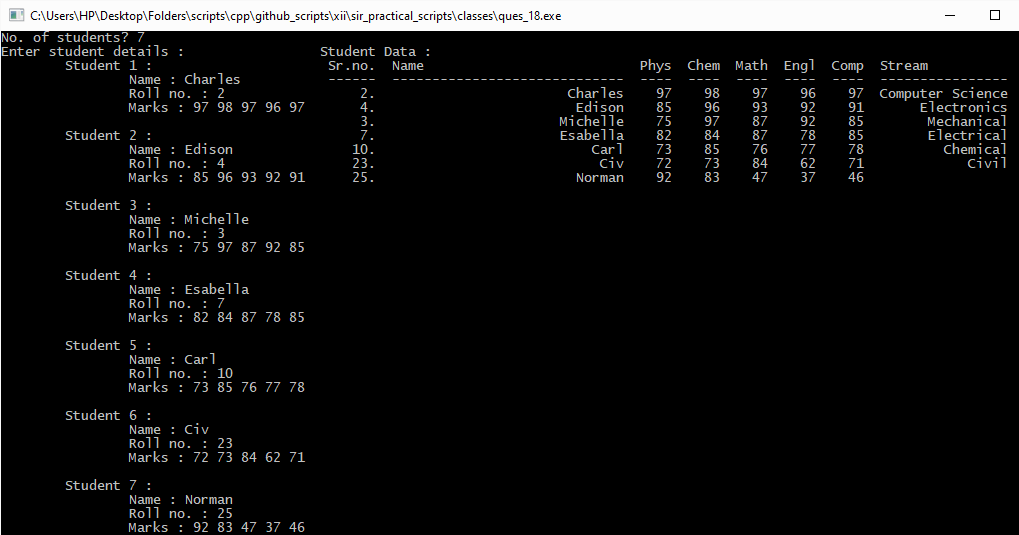
getchar();

return 0;

}

Output :





1. Divided they stand

#include <iostream>

using namespace std;

long long HCF(long long a, long long b) {

long long temp;

if (a < b) {

temp = a;

a = b;

b = temp;

}

while (a % b != 0ll) {

temp = b;

b = a % b;

a = temp;

}

return b;

}

inline long long LCM(long long a, long long b) {

return a \* (b / HCF(a,b));

}

class Fraction {

public :

long long numerator, denominator;

Fraction() {

this->numerator = 0;

this->denominator = 1;

}

Fraction(long long num, long long den) {

long long hcf = HCF(num,den);

this->numerator = num / hcf;

if (den == 0)

this->denominator = 1;

else

this->denominator = den / hcf;

if (this->denominator < 0) {

this->denominator \*= -1;

this->numerator \*= -1;

}

}

Fraction(long long k) {

\*this = Fraction(k,1);

// this->Fraction(k,1);

}

Fraction(long double d) {

\*this = Fraction(d \* 1e8, 1e8);

}

Fraction add(const Fraction& a) const {

long long den\_hcf = HCF(this->denominator, a.denominator);

return Fraction(this->numerator \* (a.denominator / den\_hcf) + a.numerator \* (this->denominator / den\_hcf),

LCM(this->denominator, a.denominator));

}

Fraction subtract(const Fraction& s) const {

long long den\_hcf = HCF(this->denominator, s.denominator);

return Fraction(this->numerator \* (s.denominator / den\_hcf) - s.numerator \* (this->denominator \* den\_hcf),

LCM(this->denominator, s.denominator));

}

Fraction multiply(const Fraction& m) const {

return Fraction(this->numerator \* m.numerator, this->denominator \* m.denominator);

}

Fraction divide(const Fraction& d) const {

return Fraction(this->numerator \* d.denominator, this->denominator \* d.numerator);

}

operator long double() {

return (long double)(this->numerator) / this->denominator;

}

inline friend Fraction operator+(const Fraction& a, const Fraction& b) {

return a.add(b);

}

inline friend Fraction operator-(const Fraction& a, const Fraction& b) {

return a.subtract(b);

}

inline friend Fraction operator\*(const Fraction& a, const Fraction& b) {

return a.multiply(b);

}

inline friend Fraction operator/(const Fraction& a, const Fraction& b) {

return a.divide(b);

}

inline friend Fraction operator+=(Fraction& a, const Fraction& b) {

return a = a.add(b);

}

inline friend Fraction operator-=(Fraction& a, const Fraction& b) {

return a = a.subtract(b);

}

inline friend Fraction operator\*=(Fraction& a, const Fraction& b) {

return a = a.multiply(b);

}

inline friend Fraction operator/=(Fraction& a, const Fraction& b) {

return a = a.divide(b);

}

friend istream& operator>>(istream& input\_stream, Fraction& f) {

while (!(input\_stream >> f.numerator)) {

input\_stream.ignore(1);

input\_stream.clear();

}

while (!(input\_stream >> f.denominator)) {

input\_stream.ignore(1);

input\_stream.clear();

}

f = Fraction(f.numerator, f.denominator);

return input\_stream;

}

friend ostream& operator<<(ostream& display\_stream, const Fraction f) {

display\_stream << f.numerator << '/' << f.denominator;

return display\_stream;

}

};

int main() {

Fraction r1, r2;

cout << "Enter r1 : "; cin >> r1;

cout << "Enter r2 : "; cin >> r2;

cout << "\nr1 = " << r1;

cout << "\nr2 = " << r2;

cout << "\n\nr1 + r2 = " << r1 + r2;

cout << "\nr1 - r2 = " << r1 - r2;

cout << "\nr1 \* r2 = " << r1 \* r2;

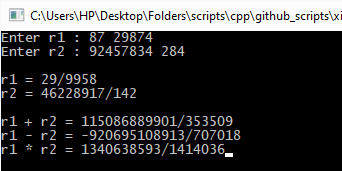
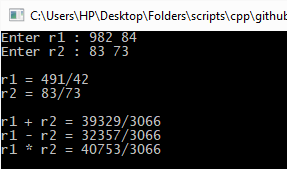
while (getchar() != '\n');

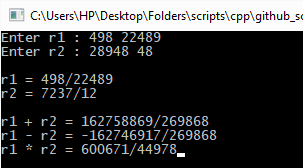
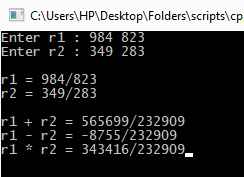
getchar();

return 0;

}

Output :





1. To the Point

#include <iostream>

#include <math.h>

using namespace std;

class Point {

public :

float x, y;

Point (float x, float y) {

this->x = x;

this->y = y;

}

Point() {

\*this = Point(0, 0);

}

Point(const Point& p) {

\*this = Point(p.x, p.y);

}

float sqrDistanceFrom(const Point& b) const {

return ((this->x - b.x)\*(this->x - b.x) + (this->y - b.y)\*(this->y - b.y));

}

float distanceFrom(const Point& b) const {

return sqrt( this->sqrDistanceFrom(b) );

}

float dotProduct(const Point& b) const {

return (this->x \* b.x + this->y \* b.y);

}

float crossProduct(const Point& b) const {

return (this->x \* b.y - this->y \* b.x);

}

friend istream& operator>>(istream& input\_stream, Point& p) {

while (!(input\_stream >> p.x)) {

input\_stream.ignore(1);

input\_stream.clear();

}

while (!(input\_stream >> p.y)) {

input\_stream.ignore(1);

input\_stream.clear();

}

return input\_stream;

}

friend ostream& operator<<(ostream& display\_stream, const Point& p) {

display\_stream << '(' << p.x << ", " << p.y << ')';

return display\_stream;

}

};

class Triangle {

public :

Point points[3];

Triangle(float x1, float y1, float x2, float y2, float x3, float y3) {

this->points[0] = Point(x1, y1);

this->points[1] = Point(x2, y2);

this->points[2] = Point(x3, y3);

}

Triangle(const Point& p1, const Point& p2, const Point& p3) {

this->points[0] = p1;

this->points[1] = p2;

this->points[2] = p3;

}

Triangle() {

\*this = Triangle(0, 0, 0, 0, 0, 0);

}

float perimeter() const {

return this->points[0].distanceFrom(this->points[1])

+ this->points[1].distanceFrom(this->points[2])

+ this->points[2].distanceFrom(this->points[0]);

}

float area() const {

return 0.5 \* abs(this->points[0].crossProduct(this->points[1])

+ this->points[1].crossProduct(this->points[2])

+ this->points[2].crossProduct(this->points[0]));

}

inline friend istream& operator>>(istream& input\_stream, Triangle& t) {

return input\_stream >> t.points[0] >> t.points[1] >> t.points[2];

}

inline friend ostream& operator<<(ostream& display\_stream, const Triangle& t) {

return display\_stream << "( " << t.points[0] << ", " << t.points[1] << ", " << t.points[2] << ")";

}

};

int main() {

Triangle quadrilateral;

cout << "Enter the points of the triangle : "; cin >> quadrilateral;

cout << "\nThe points you have entered are " << quadrilateral;

cout << "\n\nPerimeter = " << quadrilateral.perimeter();

cout << "\nArea = " << quadrilateral.area();

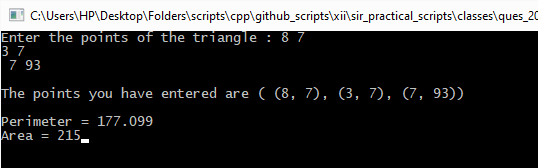
while(getchar() != '\n');

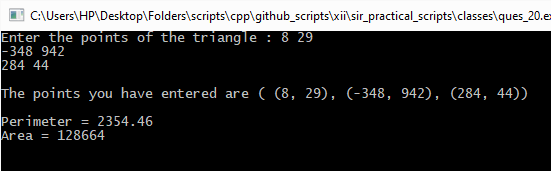
getchar();

return 0;

}

Output :





1. Part-Time Students

#include <iostream>

#include <iomanip>

using namespace std;

enum Stream {COMPUTER\_SCIENCE, ELECTRONICS, MECHANICAL, ELECTRICAL, CHEMICAL, CIVIL, NONE};

const char\* stream\_name(Stream stream) {

switch(stream) {

case COMPUTER\_SCIENCE :

return "Computer Science";

break;

case ELECTRONICS :

return "Electronics";

break;

case MECHANICAL :

return "Mechanical";

break;

case ELECTRICAL :

return "Electrical";

break;

case CHEMICAL :

return "Chemical";

break;

case CIVIL :

return "Civil";

break;

case NONE :

return "";

break;

}

}

class Student {

char name[30];

unsigned roll\_num;

unsigned marks[5];

Stream stream;

public :

Student() {

this->name[0] = '\0';

this->roll\_num = 0;

for (int i = 0 ; i < 5; ++i)

this->marks[i] = 0;

this->stream = NONE;

}

void getInput(istream& input\_stream, ostream& display\_stream, const char\* pre\_str) {

input\_stream.ignore(std::numeric\_limits<streamsize>::max(), '\n');

input\_stream.clear();

display\_stream << pre\_str << "Name : ";

input\_stream.getline(this->name, 30, '\n');

display\_stream << pre\_str << "Roll no. : ";

while (!(input\_stream >> this->roll\_num)) {

input\_stream.ignore(1);

input\_stream.clear();

}

display\_stream << pre\_str << "Marks : ";

for (int i = 0; i < 5; ++i) {

while (!(input\_stream >> this->marks[i])) {

input\_stream.ignore(1);

input\_stream.clear();

}

}

this->assignStream();

}

void printSelf(ostream& display\_stream, int name\_space\_num, int roll\_space\_num, int mark\_space\_num, int stream\_space\_num) {

display\_stream << setw(roll\_space\_num) << this->roll\_num << '.';

display\_stream << setw(name\_space\_num) << this->name;

for (int i = 0; i < 5; ++i)

display\_stream << setw(mark\_space\_num) << this->marks[i];

cout << setw(stream\_space\_num) << stream\_name(this->stream);

}

unsigned long total() {

unsigned long sum = 0;

for (int i = 0; i < 5; ++i)

sum += this->marks[i];

return sum;

}

float average() {

return this->total() / float(5);

}

Stream assignStream() {

float average\_marks = this->average();

if (average\_marks < 70)

this->stream = NONE;

else if (average\_marks < 75)

this->stream = CIVIL;

else if (average\_marks < 80)

this->stream = CHEMICAL;

else if (average\_marks < 85)

this->stream = ELECTRICAL;

else if (average\_marks < 90)

this->stream = MECHANICAL;

else if (average\_marks < 95)

this->stream = ELECTRONICS;

else

this->stream = COMPUTER\_SCIENCE;

return this->stream;

}

};

class WorkingStudent : public Student {

char job[20];

char office\_address[50];

public :

WorkingStudent() : Student() {

this->job[0] = '\0';

this->office\_address[0] = '\0';

}

void getInput(istream& input\_stream, ostream& display\_stream, const char\* pre\_str) {

this->Student::getInput(input\_stream, display\_stream, pre\_str);

input\_stream.ignore(std::numeric\_limits<streamsize>::max(), '\n');

input\_stream.clear();

display\_stream << pre\_str << "Job : ";

input\_stream.getline(this->job, 20, '\n');

input\_stream.ignore(std::numeric\_limits<streamsize>::max(), '\n');

input\_stream.clear();

display\_stream << pre\_str << "Office Address : ";

input\_stream.getline(this->office\_address, 50, '\n');

}

void printSelf(ostream& display\_stream, int name\_space\_num, int roll\_space\_num, int mark\_space\_num, int stream\_space\_num, int job\_space\_num, int off\_addr\_space\_num) {

this->Student::printSelf(display\_stream, name\_space\_num, roll\_space\_num, mark\_space\_num, stream\_space\_num);

display\_stream << setw(job\_space\_num) << this->job;

display\_stream << setw(off\_addr\_space\_num) << this->office\_address;

}

};

int main() {

WorkingStudent stu\_arr[20];

unsigned short no\_of\_students;

cout << "No. of students? ";

do {

while (!(cin >> no\_of\_students)) cin.clear();

if (no\_of\_students > 20)

cout << "ERR : input out of range (0<, <=20)";

} while (no\_of\_students > 20);

cout << "Enter student details : ";

for (int i = 0; i < no\_of\_students; ++i) {

char format\_string[] = "\t\t";

cout << "\n\tStudent " << (i + 1) << " : \n";

stu\_arr[i].getInput(cin, cout, format\_string);

}

cout << "Student Data : \n";

cout << " Sr.no. Name Phys Chem Math Engl Comp Stream Job Office Address\n"

<< " ------ ----------------------------- ---- ---- ---- ---- ---- ---------------- -------------------- --------------------------------------------------\n";

for (int i = 0; i < no\_of\_students; ++i) {

stu\_arr[i].printSelf(cout, 31, 6, 6, 18, 22, 52);

cout << '\n';

}

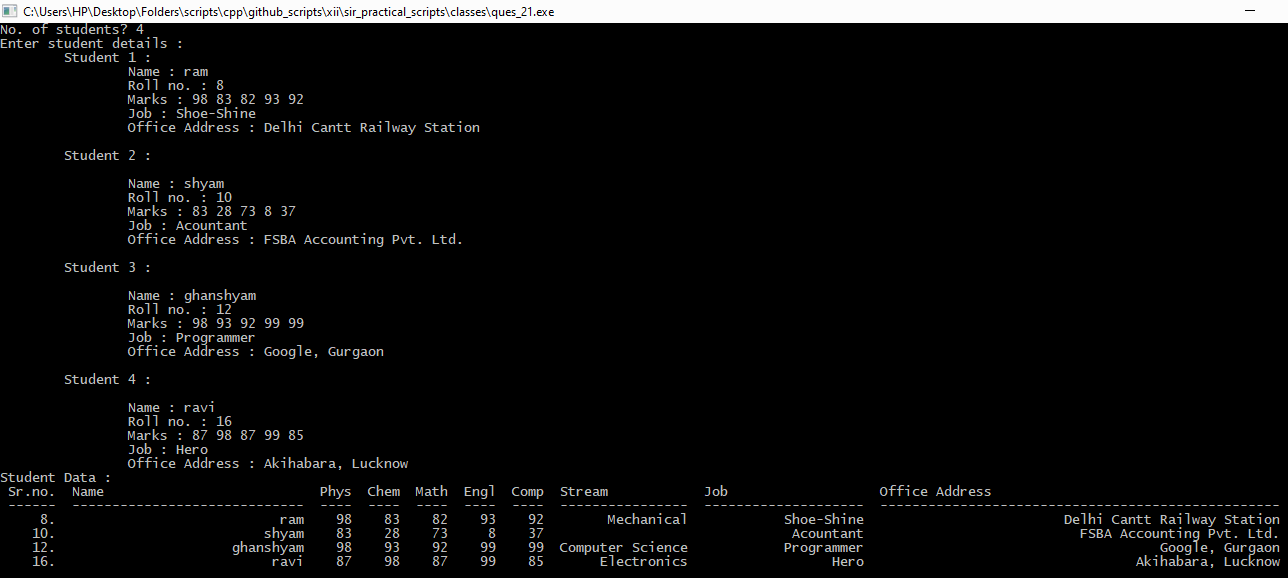
while(getchar() != '\n');

getchar();

return 0;

}

Output :



For text file-handling :

* lorem\_ipsum.txt :

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec semper mollis tempor. Pellentesque interdum lorem mauris, eget condimentum justo semper sit amet. Duis bibendum cursus turpis, ac ornare purus. Donec mollis neque eget faucibus consectetur. Proin iaculis sit amet massa et molestie. Sed consectetur est a scelerisque varius. Quisque gravida metus tortor, sed pulvinar ipsum cursus quis. Nullam rhoncus molestie diam, quis tempor nibh tristique at. Aliquam placerat bibendum lacus, non ornare nulla ornare id. Integer sit amet eros nulla. Sed et posuere libero, eget faucibus justo.

Integer faucibus ligula vitae diam congue, nec porttitor turpis porta. Phasellus non tempus arcu. Nulla diam dui, interdum at massa vel, tempus rutrum tortor. Fusce a tempor tortor. Maecenas non auctor massa, fermentum fermentum nisi. Etiam facilisis arcu vitae consectetur sagittis. Donec eget facilisis purus, eget eleifend nulla. Quisque venenatis, justo quis convallis facilisis, lacus lorem facilisis nunc, vitae venenatis velit tortor id est. Nam dictum eros ac risus facilisis pulvinar lobortis nec massa. Phasellus a pulvinar dolor, id iaculis ligula. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Ut non tortor hendrerit, finibus neque quis, euismod urna. Vivamus eleifend lorem pellentesque gravida vestibulum. Etiam ac quam tincidunt, scelerisque arcu quis, viverra massa. Vivamus sollicitudin maximus nulla, nec condimentum sapien efficitur sit amet.

Integer fermentum augue quis nisl maximus rutrum. Nunc nec pharetra leo. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Quisque finibus porta est, non porttitor justo cursus ut. Suspendisse eleifend tempus massa, eget venenatis odio finibus ut. Donec ante orci, ullamcorper eget volutpat ut, auctor ornare massa. Fusce ornare sem nulla, id vulputate enim consectetur quis. Donec turpis augue, dignissim in nisi vitae, blandit dignissim tortor. Proin nec elementum ipsum, nec efficitur mauris.

Donec hendrerit bibendum felis sed sagittis. Aenean rutrum mi sit amet mauris lobortis, ac sodales mi vulputate. Fusce consectetur feugiat vehicula. Vivamus at dolor nec mauris scelerisque imperdiet. In accumsan, erat eu dapibus viverra, lacus nibh sodales purus, id suscipit risus lorem quis leo. Pellentesque et quam turpis. Aenean libero justo, pharetra ac neque convallis, efficitur sollicitudin ex. Cras auctor posuere sem sed hendrerit. Nullam eu metus et neque fringilla molestie at condimentum ex. Phasellus vitae leo placerat velit aliquet tincidunt. Integer volutpat ut ante nec porttitor.

Nunc nulla dolor, accumsan nec maximus id, gravida ut risus. Maecenas ullamcorper a justo quis venenatis. Aenean cursus libero et dolor dignissim imperdiet. Suspendisse feugiat dui sed rutrum semper. Mauris interdum non orci ut pulvinar. Quisque porta urna eget lacus tempor rutrum. Aenean velit nunc, rhoncus ac venenatis a, porta ut ligula. Etiam ut neque sed neque pellentesque volutpat eu ut magna. Vestibulum a leo eu orci imperdiet consequat. Ut ac est at tortor porta tincidunt ac sit amet libero. Sed varius congue arcu vel luctus. Vestibulum quis dolor vitae mauris facilisis feugiat. In vehicula dolor quis tristique molestie.

1. Text Census

#include <iostream>

#include <fstream>

using namespace std;

bool isUpperCase(char c) {

return (c <= 'Z' and c >= 'A');

}

bool isLowerCase(char c) {

return (c <= 'z' and c >= 'a');

}

bool isAlpha(char c) {

return isUpperCase(c) or isLowerCase(c);

}

bool isDigit(char c) {

return (c <= '9' and c >= '0');

}

bool isAlphaNum(char c) {

return isAlpha(c) or isDigit(c);

}

bool isVowel(char c) {

switch (c) {

case 'a' : case 'e' : case 'i' : case 'o' : case 'u' : case 'A' : case 'E' : case 'I' : case 'O' : case 'U' :

return true;

break;

default :

return false;

break;

}

}

int main() {

char file\_name[51];

ifstream input\_file;

cout << "Enter file name (max 50 char) : "; cin.getline(file\_name, 51, '\n');

input\_file.open(file\_name, ios::in);

if (not input\_file.is\_open()) {

cout << "ERR : File does not exist\n";

return 0;

}

input\_file.seekg(0, ios::beg);

long long file\_length = -input\_file.tellg();

input\_file.seekg(0, ios::end);

file\_length += input\_file.tellg();

input\_file.seekg(0, ios::beg);

char ch, prev\_ch;

int no\_of\_blanks = 0, no\_of\_lines = 0, no\_of\_up = 0, no\_of\_low = 0, no\_of\_lines\_start\_up = 0, no\_of\_words = 0, no\_of\_digits = 0, no\_of\_words\_end\_vow = 0;

bool word\_started = false;

//For first character

input\_file.get(ch);

input\_file.seekg(1, ios::beg);

if (isUpperCase(ch)) {

no\_of\_up++; word\_started = true;

no\_of\_lines\_start\_up++;

} else if (isLowerCase(ch)) {

no\_of\_low++; word\_started = true;

} else if (isDigit(ch)) {

no\_of\_digits++;

} else if (ch == '\n' or ch == '\r') {

no\_of\_lines++;

no\_of\_blanks++;

}

while (not input\_file.eof()) {

prev\_ch = ch;

input\_file.read(&ch, 1);

if (isUpperCase(ch)) {

no\_of\_up++; word\_started = true;

if (prev\_ch == '\n' or prev\_ch == '\r')

no\_of\_lines\_start\_up++;

} else if (isLowerCase(ch)) {

no\_of\_low++; word\_started = true;

} else if (isDigit(ch)) {

no\_of\_digits++;

} else if (ch == '\n' or ch == '\r') {

no\_of\_lines++;

if (word\_started)

no\_of\_words++;

word\_started = false;

if (isVowel( prev\_ch ))

no\_of\_words\_end\_vow++;

else if (prev\_ch == '\n' or prev\_ch == '\r')

no\_of\_blanks++;

} else if (ch == ' ') {

if (word\_started)

no\_of\_words++;

word\_started = false;

if (isVowel( prev\_ch ))

no\_of\_words\_end\_vow++;

} else if (not isAlphaNum(ch)) {

if (word\_started)

no\_of\_words++;

word\_started = false;

}

}

input\_file.clear();

no\_of\_lines++;

if (word\_started)

no\_of\_words++;

word\_started = false;

if(isVowel(input\_file.get())) {

no\_of\_words\_end\_vow++;

}

cout << "\n\nNo . of blanks = " << no\_of\_blanks;

cout << "\nNo . of lines = " << no\_of\_lines;

cout << "\nNo . of UPPERCASE letters = " << no\_of\_up;

cout << "\nNo . of lowercase letters = " << no\_of\_low;

cout << "\nNo . of lines starting with a Capital letter = " << no\_of\_lines\_start\_up;

cout << "\nNo . of words = " << no\_of\_words;

cout << "\nNo . of digits = " << no\_of\_digits;

cout << "\nNo . of words ending with a vowel = " << no\_of\_words\_end\_vow;

input\_file.close();

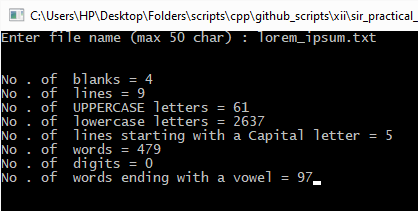
while (cin.get() != '\n');

cin.get();

return 0;

}

Output :



1. Substring finder

#include <iostream>

#include <fstream>

using namespace std;

int main() {

ifstream input\_file;

char file\_name[100], substring[100];

bool present = false;

cout << "Enter filename for input : "; cin.getline(file\_name, 100);

input\_file.open(file\_name, ios::in);

if (not input\_file.is\_open()) {

cerr << "ValueError : No such file as '" << file\_name << "'\n";

return 1;

}

input\_file.seekg(0, ios::beg);

cout << "Enter string to search : "; cin.getline(substring, 100);

while (not input\_file.eof() and not present) {

char first\_char;

input\_file.read(&first\_char, 1);

if (first\_char == substring[0]) {

present = true;

for (int i = 1; substring[i] != '\0' and present; ++i) {

input\_file.read(&first\_char, 1);

present &= substring[i] == first\_char;

}

}

}

if (present)

cout << "Given substring '" << substring << "' is present in '" << file\_name << "'\n";

else

cout << "Given substring '" << substring << "' is not present in '" << file\_name << "'\n";

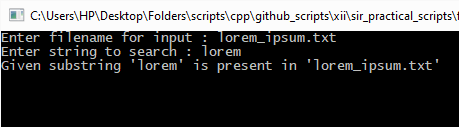
while (cin.get() != '\n');

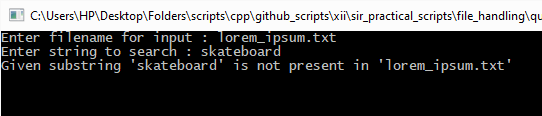
cin.get();

return 0;

}

Output :





1. Vowel terminator

#include <iostream>

#include <fstream>

using namespace std;

bool isVowel(char c) {

switch (c) {

case 'a' : case 'e' : case 'i' : case 'o' : case 'u' : case 'A' : case 'E' : case 'I' : case 'O' : case 'U' :

return true;

break;

default :

return false;

break;

}

}

int main() {

ifstream input\_file;

ofstream output\_file("copy.txt", ios::out);

char file\_name[100], word[100];

cout << "Enter filename for input : "; cin.getline(file\_name, 100);

input\_file.open(file\_name, ios::in);

if (not input\_file.is\_open()) {

cerr << "ValueError : No such file as '" << file\_name << "'\n";

return 1;

}

input\_file.seekg(0, ios::beg);

while (not input\_file.eof() ) {

input\_file >> word;

int i;

for (i = 0; word[i] != '\0'; ++i);

if (not isVowel(word[i - 1]))

output\_file << word << ' ';

}

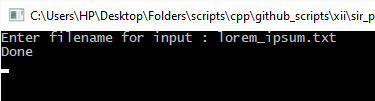
cout << "Done\n";

while (cin.get() != '\n');

cin.get();

}

Output :



* copy.txt :

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec semper mollis tempor. interdum lorem mauris, eget condimentum semper sit amet. Duis bibendum cursus turpis, ac purus. Donec mollis eget faucibus consectetur. Proin iaculis sit amet et molestie. Sed consectetur est varius. metus tortor, sed pulvinar ipsum cursus quis. Nullam rhoncus diam, quis tempor nibh at. Aliquam placerat bibendum lacus, non id. Integer sit amet eros nulla. Sed et libero, eget faucibus justo. Integer faucibus diam congue, nec porttitor turpis porta. Phasellus non tempus arcu. diam dui, interdum at vel, tempus rutrum tortor. tempor tortor. Maecenas non auctor massa, fermentum fermentum nisi. Etiam facilisis consectetur sagittis. Donec eget facilisis purus, eget eleifend nulla. venenatis, quis convallis facilisis, lacus lorem facilisis nunc, venenatis velit tortor id est. Nam dictum eros ac risus facilisis pulvinar lobortis nec massa. Phasellus pulvinar dolor, id iaculis ligula. habitant senectus et netus et fames ac turpis egestas. Ut non tortor hendrerit, finibus quis, euismod urna. Vivamus eleifend lorem vestibulum. Etiam ac quam tincidunt, quis, massa. Vivamus sollicitudin maximus nulla, nec condimentum sapien efficitur sit amet. Integer fermentum quis nisl maximus rutrum. Nunc nec leo. Vestibulum ipsum primis in faucibus luctus et ultrices Curae; finibus est, non porttitor cursus ut. eleifend tempus massa, eget venenatis finibus ut. Donec orci, ullamcorper eget volutpat ut, auctor massa. sem nulla, id enim consectetur quis. Donec turpis augue, dignissim in vitae, blandit dignissim tortor. Proin nec elementum ipsum, nec efficitur mauris. Donec hendrerit bibendum felis sed sagittis. Aenean rutrum sit amet mauris lobortis, ac sodales vulputate. consectetur feugiat vehicula. Vivamus at dolor nec mauris imperdiet. In accumsan, erat dapibus viverra, lacus nibh sodales purus, id suscipit risus lorem quis leo. et quam turpis. Aenean justo, ac convallis, efficitur sollicitudin ex. Cras auctor sem sed hendrerit. Nullam metus et at condimentum ex. Phasellus placerat velit aliquet tincidunt. Integer volutpat ut nec porttitor. Nunc dolor, accumsan nec maximus id, ut risus. Maecenas ullamcorper quis venenatis. Aenean cursus et dolor dignissim imperdiet. feugiat sed rutrum semper. Mauris interdum non ut pulvinar. eget lacus tempor rutrum. Aenean velit nunc, rhoncus ac venenatis a, ut ligula. Etiam ut sed volutpat ut magna. Vestibulum imperdiet consequat. Ut ac est at tortor tincidunt ac sit amet libero. Sed varius vel luctus. Vestibulum quis dolor mauris facilisis feugiat. In dolor quis molestie.

1. Change of Characters

#include <iostream>

#include <fstream>

#include <stdio.h>

using namespace std;

int main() {

fstream input\_file, temp\_file;

char file\_name[100];

char old\_ch, new\_ch;

cout << "Enter input filename : "; cin.getline(file\_name, 100);

input\_file.open(file\_name, ios::in);

temp\_file.open(".temp.txt", ios::out);

if (not input\_file.is\_open()) {

cerr << "ValueError : Invalid filename given";

return 1;

}

input\_file.seekg(0, ios::beg);

temp\_file.seekp(0, ios::beg);

cout << "Enter the two characters to find and replace : ";

cin >> old\_ch >> new\_ch;

while (not input\_file.eof()) {

char ch;

input\_file.read(&ch, 1);

if (ch == old\_ch)

temp\_file.write(&new\_ch, 1);

else

temp\_file.write(&ch, 1);

}

input\_file.close();

temp\_file.close();

remove(file\_name);

rename(”.temp.txt”, file\_name);

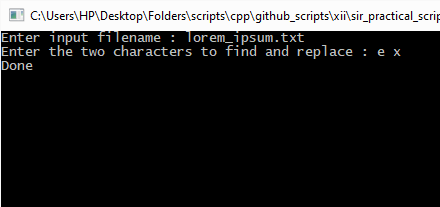
cout << "Done";

while( cin.get() != '\n');

cin.get();

}

Output :



* lorem\_ipsum.txt (after):

Lorxm ipsum dolor sit amxt, consxctxtur adipiscing xlit. Donxc sxmpxr mollis txmpor. Pxllxntxsqux intxrdum lorxm mauris, xgxt condimxntum justo sxmpxr sit amxt. Duis bibxndum cursus turpis, ac ornarx purus. Donxc mollis nxqux xgxt faucibus consxctxtur. Proin iaculis sit amxt massa xt molxstix. Sxd consxctxtur xst a scxlxrisqux varius. Quisqux gravida mxtus tortor, sxd pulvinar ipsum cursus quis. Nullam rhoncus molxstix diam, quis txmpor nibh tristiqux at. Aliquam placxrat bibxndum lacus, non ornarx nulla ornarx id. Intxgxr sit amxt xros nulla. Sxd xt posuxrx libxro, xgxt faucibus justo.

Intxgxr faucibus ligula vitax diam congux, nxc porttitor turpis porta. Phasxllus non txmpus arcu. Nulla diam dui, intxrdum at massa vxl, txmpus rutrum tortor. Fuscx a txmpor tortor. Maxcxnas non auctor massa, fxrmxntum fxrmxntum nisi. Etiam facilisis arcu vitax consxctxtur sagittis. Donxc xgxt facilisis purus, xgxt xlxifxnd nulla. Quisqux vxnxnatis, justo quis convallis facilisis, lacus lorxm facilisis nunc, vitax vxnxnatis vxlit tortor id xst. Nam dictum xros ac risus facilisis pulvinar lobortis nxc massa. Phasxllus a pulvinar dolor, id iaculis ligula. Pxllxntxsqux habitant morbi tristiqux sxnxctus xt nxtus xt malxsuada famxs ac turpis xgxstas. Ut non tortor hxndrxrit, finibus nxqux quis, xuismod urna. Vivamus xlxifxnd lorxm pxllxntxsqux gravida vxstibulum. Etiam ac quam tincidunt, scxlxrisqux arcu quis, vivxrra massa. Vivamus sollicitudin maximus nulla, nxc condimxntum sapixn xfficitur sit amxt.

Intxgxr fxrmxntum augux quis nisl maximus rutrum. Nunc nxc pharxtra lxo. Vxstibulum antx ipsum primis in faucibus orci luctus xt ultricxs posuxrx cubilia Curax; Quisqux finibus porta xst, non porttitor justo cursus ut. Suspxndissx xlxifxnd txmpus massa, xgxt vxnxnatis odio finibus ut. Donxc antx orci, ullamcorpxr xgxt volutpat ut, auctor ornarx massa. Fuscx ornarx sxm nulla, id vulputatx xnim consxctxtur quis. Donxc turpis augux, dignissim in nisi vitax, blandit dignissim tortor. Proin nxc xlxmxntum ipsum, nxc xfficitur mauris.

Donxc hxndrxrit bibxndum fxlis sxd sagittis. Axnxan rutrum mi sit amxt mauris lobortis, ac sodalxs mi vulputatx. Fuscx consxctxtur fxugiat vxhicula. Vivamus at dolor nxc mauris scxlxrisqux impxrdixt. In accumsan, xrat xu dapibus vivxrra, lacus nibh sodalxs purus, id suscipit risus lorxm quis lxo. Pxllxntxsqux xt quam turpis. Axnxan libxro justo, pharxtra ac nxqux convallis, xfficitur sollicitudin xx. Cras auctor posuxrx sxm sxd hxndrxrit. Nullam xu mxtus xt nxqux fringilla molxstix at condimxntum xx. Phasxllus vitax lxo placxrat vxlit aliquxt tincidunt. Intxgxr volutpat ut antx nxc porttitor.

Nunc nulla dolor, accumsan nxc maximus id, gravida ut risus. Maxcxnas ullamcorpxr a justo quis vxnxnatis. Axnxan cursus libxro xt dolor dignissim impxrdixt. Suspxndissx fxugiat dui sxd rutrum sxmpxr. Mauris intxrdum non orci ut pulvinar. Quisqux porta urna xgxt lacus txmpor rutrum. Axnxan vxlit nunc, rhoncus ac vxnxnatis a, porta ut ligula. Etiam ut nxqux sxd nxqux pxllxntxsqux volutpat xu ut magna. Vxstibulum a lxo xu orci impxrdixt consxquat. Ut ac xst at tortor porta tincidunt ac sit amxt libxro. Sxd varius congux arcu vxl luctus. Vxstibulum quis dolor vitax mauris facilisis fxugiat. In vxhicula dolor quis tristiqux molxstix.

1. Telerekt

#include <iostream>

#include <fstream>

#include <string.h>

#include <limits>

using namespace std;

struct TeleRec {

char name[21];

char tele\_no[11];

void display(ostream& out\_strm) {

out\_strm << "\n\tName : " << name;

out\_strm << "\n\tTele no. : " << tele\_no;

}

void getData(istream& in\_strm, ostream& out\_strm) {

out\_strm << "\n\tName : "; in\_strm.getline(name, 21);

out\_strm << "\tTele no. : "; in\_strm.getline(tele\_no, 11);

}

};

int main() {

char choice;

bool found;

char person\_name[21], phone\_no[11];

TeleRec record;

fstream record\_file;

record\_file.open("tele\_records.dat", ios::in | ios::out | ios::app | ios::binary);

if (not (record\_file.is\_open() and record\_file.good())) {

record\_file.close();

record\_file.open("tele\_records.dat", ios::out);

record\_file.close();

record\_file.open("tele\_records.dat", ios::in | ios::out | ios::app | ios::binary);

if (not (record\_file.is\_open() and record\_file.good())) {

cerr << "FileError : cannot open file stream \"tele\_records.dat\"";

return 1;

}

}

for (int i = 1; record\_file.read((char\*)(&record), sizeof(TeleRec)); ++i) {

cout << "\nRecord " << i << " : ";

record.display(cout);

}

do {

cout << "\n\nPick an option : ";

cout << "\n\t1 - Append records"

<< "\n\t2 - Search by Telephone number"

<< "\n\t3 - Search by Name"

<< "\n\t4 - Exit\n";

cin >> choice;

switch (choice) {

case '1' :

int n;

cout << "No. of records to append? "; cin >> n;

cin.ignore(std::numeric\_limits<streamsize>::max(), '\n');

record\_file.clear(); record\_file.seekp(0, ios::end);

for (int i = 0; i < n; ++i) {

cout << "\nEnter Record Details : ";

record.getData(cin, cout);

record\_file.write((char\*)(&record), sizeof(TeleRec));

}

break;

case '2' :

cin.ignore(std::numeric\_limits<streamsize>::max(), '\n');

cout << "\nEnter Telephone no. : "; cin.getline(phone\_no, 11);

found = false;

record\_file.clear(); record\_file.seekg(0, ios::beg);

for (int i = 1; record\_file.read((char\*)(&record), sizeof(TeleRec)) and not found; ++i) {

if (strcmp(phone\_no, record.tele\_no) == 0) {

found = true;

cout << "\nRecord " << i << " : ";

record.display(cout);

}

}

if (not found) {

cout << "\nRecord not found\n";

}

break;

case '3' :

cin.ignore(std::numeric\_limits<streamsize>::max(), '\n');

cout << "\nEnter Name : "; cin.getline(person\_name, 21);

found = false;

record\_file.clear(); record\_file.seekg(0, ios::beg);

for (int i = 1; record\_file.read((char\*)(&record), sizeof(TeleRec)); ++i) {

if (strcmp(person\_name, record.name) == 0) {

found = true;

cout << "\nRecord " << i << " : ";

record.display(cout);

}

}

if (not found) {

cout << "\nRecord not found\n";

}

break;

case '4' :

break;

default :

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

break;

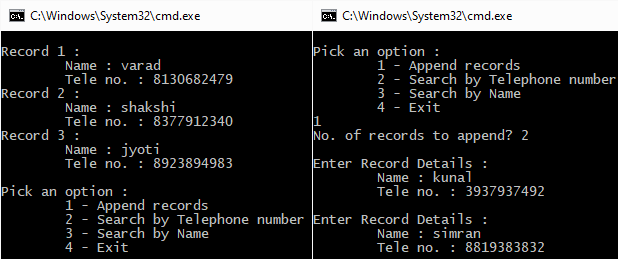
}

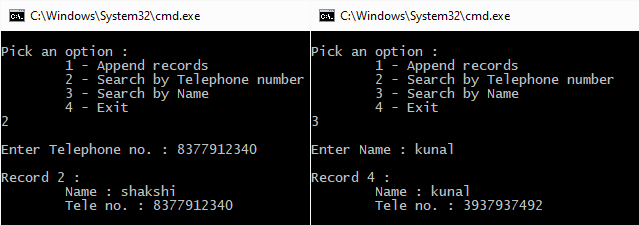
} while (choice != '4');

record\_file.close();

}

Output :





1. Blood Thrist

#include <iostream>

#include <fstream>

#include <string.h>

#include <limits>

using namespace std;

void pullUpper(char\* str, unsigned int size) {

for (int i = 0; i < size; ++i) {

if (str[i] >= 'a' and str[i] <= 'z')

str[i] -= 32;

}

}

struct BloodDonor {

char name[21];

unsigned short dob\_day;

unsigned short dob\_month;

unsigned short dob\_year;

char tele\_no[11];

char group[4];

void display(ostream& out\_stream) {

out\_stream << "\n\tName : " << name;

out\_stream << "\n\tDate of Birth : " << dob\_day << '-' << dob\_month << '-' << dob\_year;

out\_stream << "\n\tBlood Group : " << group;

out\_stream << "\n\tTele no. : " << tele\_no;

}

void getData(istream& in\_stream, ostream& out\_stream) {

out\_stream << "\n\tName : "; in\_stream.getline(name, 21);

out\_stream << "\tDate Of Birth (DD MM YYYY) : "; in\_stream >> dob\_day >> dob\_month >> dob\_year;

in\_stream.ignore(std::numeric\_limits<streamsize>::max(), '\n');

out\_stream << "\tBlood Group : "; in\_stream.getline(group, 4);

out\_stream << "\tTelephone no. : "; in\_stream.getline(tele\_no, 11);

pullUpper(group, 4);

}

};

int main() {

char choice;

bool found;

char blood\_group[4], phone\_no[11];

BloodDonor record;

fstream record\_file("blood\_bank\_records.dat", ios::in | ios::out | ios::ate | ios::binary);

if (not record\_file.is\_open()) {

record\_file.close();

// ios::in requires that the file exists, so here we create the file

record\_file.open("blood\_bank\_records.dat", ios::out);

record\_file.close();

// Reopen file normally

record\_file.open("blood\_bank\_records.dat", ios::in | ios::out | ios::ate | ios::binary);

// Something is wrong if this still doesn't work

if (not record\_file.is\_open()) {

cerr << "\n\nFileError : Unable to open file stream\n\n";

return 1;

}

}

record\_file.seekg(0, ios::beg);

for (int i = 1; record\_file.read((char\*)(&record), sizeof(BloodDonor)); ++i) {

cout << "\nRecord " << i << " : ";

record.display(cout);

}

do {

cout << "\n\nPick an option : ";

cout << "\n\t1 - Append records"

<< "\n\t2 - Modify by Telephone number"

<< "\n\t3 - Search by Blood group"

<< "\n\t4 - Exit\n";

cin >> choice;

switch (choice) {

case '1' :

int n;

cout << "No. of records to append? "; cin >> n;

cin.ignore(std::numeric\_limits<streamsize>::max(), '\n');

// Clear EOF errors, if any, and go to the end

record\_file.clear(); record\_file.seekp(0, ios::end);

for (int i = 0; i < n; ++i) {

cout << "\nEnter Record Details : ";

record.getData(cin, cout);

record\_file.write((char\*)(&record), sizeof(BloodDonor));

}

break;

case '2' :

cin.ignore(std::numeric\_limits<streamsize>::max(), '\n');

cout << "\nEnter Telephone no. : "; cin.getline(phone\_no, 11);

found = false;

// Clear EOF errors, if any, and go to the beginning

record\_file.clear(); record\_file.seekg(0, ios::beg);

for (int i = 1; record\_file.read((char\*)(&record), sizeof(BloodDonor)) and not found; ++i) {

if (strcmp(phone\_no, record.tele\_no) == 0) {

found = true;

cout << "\nRecord " << i << " : ";

record.display(cout);

cout << "\n\nEnter modified data : ";

record.getData(cin, cout);

record\_file.seekg(-sizeof(BloodDonor), ios::cur);

record\_file.seekp(record\_file.tellg(), ios::beg);

record\_file.write((char\*)(&record), sizeof(BloodDonor));

}

}

if (not found) {

cout << "\nRecord not found\n";

}

break;

case '3' :

cin.ignore(std::numeric\_limits<streamsize>::max(), '\n');

cout << "\nEnter blood group : "; cin.getline(blood\_group, 4);

found = false;

record\_file.clear(); record\_file.seekg(0, ios::beg);

for (int i = 1; record\_file.read((char\*)(&record), sizeof(BloodDonor)); ++i) {

if (strcmp(blood\_group, record.group) == 0) {

found = true;

cout << "\nRecord " << i << " : ";

record.display(cout);

}

}

if (not found) {

cout << "\nRecord not found\n";

}

break;

case '4' :

break;

default :

cerr << "Invalid input. Try again\n";

break;

}

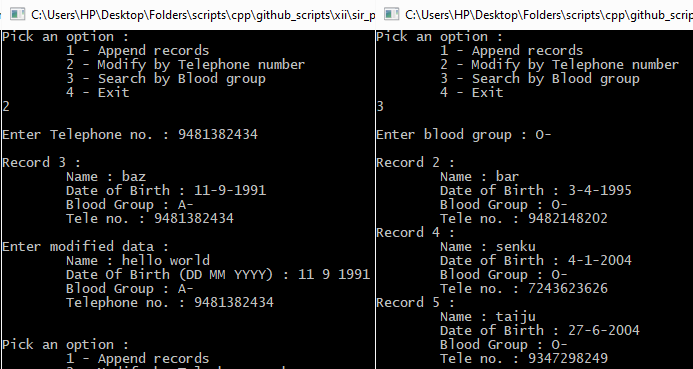
} while (choice != '4');

record\_file.close();

}

Output :





1. Employee Combinatorics

#include <fstream>

#include <iostream>

using namespace std;

struct Employee {

int EmpNo;

char Name[20];

float Salary;

void display(ostream& out\_strm) {

cout << "\nEmployee " << EmpNo << " : ";

cout << "\n\tName : " << Name;

cout << "\n\tSalary : " << Salary;

}

};

int main() {

ifstream in\_1("comp1.dat", ios::in | ios::binary), in\_2("comp2.dat", ios::in | ios::binary);

ofstream out("newcomp.dat", ios::out | ios::binary | ios::trunc);

int n\_1 = 0, n\_2 = 0;

Employee emp1, emp2;

if (not (in\_1.good() and in\_1.is\_open())) {

cerr << "FileError : cannot open file stream \"comp1.dat\"";

return 1;

}

if ( not (in\_2.good() and in\_2.is\_open())) {

cerr << "FileError : cannot open file stream \"comp2.dat\"";

return 1;

}

cout << "\n\nContents of \"comp1.dat\" : ";

while( in\_1.read((char\*)&emp1, sizeof(Employee))) {

emp1.display(cout);

n\_1++;

}

cout << "\n\nContents of \"comp2.dat\" : ";

while( in\_2.read((char\*)&emp2, sizeof(Employee))) {

emp2.display(cout);

n\_2++;

}

out.seekp(0, ios::beg);

cout << "\n\nContents of \"newcomp.dat\" : ";

int i,j;

in\_1.clear(); in\_2.clear();

in\_1.seekg(0, ios::beg); in\_2.seekg(0, ios::beg);

in\_1.read((char\*)&emp1, sizeof(Employee));

in\_2.read((char\*)&emp2, sizeof(Employee));

for ( i = 0, j = 0; i < n\_1 and j < n\_2;) {

if (emp1.EmpNo < emp2.EmpNo) {

out.write((char\*)&emp1, sizeof(Employee));

emp1.display(cout);

in\_1.read((char\*)&emp1, sizeof(Employee));

i++;

} else {

out.write((char\*)&emp2, sizeof(Employee));

emp2.display(cout);

in\_2.read((char\*)&emp2, sizeof(Employee));

j++;

}

}

if (i == n\_1) {

for (; j < n\_2; j++) {

out.write((char\*)&emp2, sizeof(Employee));

emp2.display(cout);

in\_2.read((char\*)&emp2, sizeof(Employee));

}

} else {

for (; i < n\_1; i++) {

out.write((char\*)&emp1, sizeof(Employee));

emp1.display(cout);

in\_1.read((char\*)&emp1, sizeof(Employee));

}

}

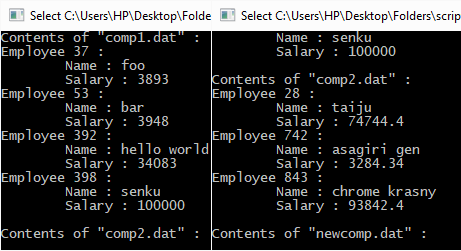
cout <<"\n\n";

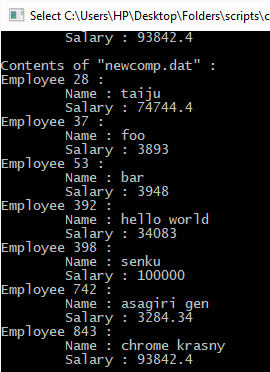
cin.ignore(std::numeric\_limits<streamsize>::max(), '\n');

cin.get();

}

Output :





1. Book Worm

#include <fstream>

#include <iostream>

#include <iomanip>

#include <limits>

#include <string.h>

#include <stdio.h>

#include "cls.h"

using namespace std;

void strxor(char\* str\_1, unsigned int size\_1, char\* str\_2, unsigned int size\_2) {

for (int i = 0; i < size\_1;)

for (int j = 0; i < size\_1 and j < size\_2; j++ and i++) {

str\_1[i] ^= str\_2[j];

}

}

class Book{

public:

int BookNo;

char Book\_name[20];

// function to enter book details

void enterdetails() {

cout << "\n\tBook no. : "; cin >> BookNo;

cout << "\n\tBook name : ";

cin.ignore(std::numeric\_limits<streamsize>::max(), '\n');

cin.getline(Book\_name, 20);

}

//function to display Book details

void showdetails() {

cout << "\n\tBook no. : " << BookNo;

cout << "\n\tBook Name : " << Book\_name;

}

//function to return Book\_no

int Rbook\_no() {return BookNo;}

//function to return Book\_name

char\* Rbook\_name() {return Book\_name;}

// encrypt/decrypt

void crypt(char\* password, unsigned int size) {

strxor((char\*)this, sizeof(Book), password, size);

}

};

class Library {

fstream shelf;

char file\_path[100];

char password[11];

public :

Book cur\_book;

Library() : shelf() {}

Library(char file\_path[]) : shelf(file\_path, ios::in | ios::out | ios::binary) {

strcpy(this->file\_path, file\_path);

if (not shelf.is\_open()) {

shelf.close();

shelf.open(file\_path, ios::out);

shelf.close();

shelf.open(file\_path, ios::in | ios::out | ios::binary);

if (not shelf.is\_open()) {

cerr << "\n\nFileError : cannot open file stream\n\n";

}

}

shelf.seekp(0, ios::beg); shelf.seekg(0, ios::beg);

}

~Library() {

for (int i = 0; i < 11; ++i)

password[i] = '\0';

shelf.close();

}

void setPassword(char\* passwd) {

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

password[i] = passwd[i];

password[10] = '\0';

}

int size() {

if (shelf.is\_open() and shelf.good()) {

int temp = shelf.tellp();

shelf.seekp(0, ios::beg);

int length = -shelf.tellp();

shelf.seekp(0, ios::end);

length += shelf.tellp();

shelf.seekp(temp);

return length/sizeof(Book);

} else

return -1;

}

bool loadBook(unsigned int index) {

if (index >= size() or not (shelf.is\_open() and shelf.good()))

return false;

shelf.seekp(index \* sizeof(Book), ios::beg);

if( not shelf.read((char\*)&cur\_book, sizeof(Book)) )

return false;

cur\_book.crypt(password, 10);

return true;

}

bool loadNextBook() {

if (not (shelf.is\_open() and shelf.good()))

return false;

if( not shelf.read((char\*)&cur\_book, sizeof(Book)) )

return false;

cur\_book.crypt(password, 10);

return true;

}

bool pushBook(unsigned int index) {

if (index >= size() or not (shelf.is\_open() and shelf.good()))

return false;

shelf.seekg(index \* sizeof(Book), ios::beg);

cur\_book.crypt(password, 10);

if( not shelf.write((char\*)&cur\_book, sizeof(Book)) )

return false;

cur\_book.crypt(password, 10);

return true;

}

bool pushNextBook() {

if (not (shelf.is\_open() and shelf.good()))

return false;

cur\_book.crypt(password, 10);

if( not shelf.write((char\*)&cur\_book, sizeof(Book)) )

return false;

cur\_book.crypt(password, 10);

return true;

}

bool removeBook(unsigned int index) {

if (index > size() or not (shelf.is\_open() and shelf.good()))

return false;

int siz = size();

ofstream temp(".tempbinarydatafile", ios::in | ios::binary | ios::trunc);

char\* buf = new char[sizeof(Book)];

if (buf == nullptr)

return false;

shelf.seekg(0, ios::beg);

temp.seekp(0, ios::beg);

// Create temp file without the given index

for (int i = 0; i < index; ++i) {

shelf.read(buf, sizeof(Book));

temp.write(buf, sizeof(Book));

}

shelf.seekg(sizeof(Book), ios::cur);

for (int i = index + 1; i < siz; ++i) {

shelf.read(buf, sizeof(Book));

temp.write(buf, sizeof(Book));

}

// Rename temp to shelf

shelf.close();

temp.close();

remove(file\_path);

rename(".tempbinarydatafile", file\_path);

remove(".tempbinarydatafile");

// Reopen shelf

shelf.open(file\_path, ios::in | ios::out | ios::binary);

if (not shelf.is\_open()) {

shelf.close();

shelf.open(file\_path, ios::out);

shelf.close();

shelf.open(file\_path, ios::in | ios::out | ios::binary);

if (not shelf.is\_open()) {

cerr << "\n\nFileError : cannot open file stream\n\n";

return false;

}

}

return true;

}

void display(ostream& out\_strm) {

int siz = size();

shelf.seekg(0, ios::beg);

out\_strm << "\tIndex Book no. Book Name\n";

out\_strm << "\t===== ======== ====================\n";

for (int i = 0; i < siz; i++) {

loadNextBook();

out\_strm << '\t' << setw(5) << i << setw(10) << cur\_book.BookNo << setw(22) << cur\_book.Book\_name << '\n';

}

}

void sortByName() {

struct smallBook\_n{

int index;

char book\_name[20];

void quickSort\_n(smallBook\_n\* arr, unsigned int start, unsigned int end) {

if (start + 1 >= end)

return;

int pivotIndex = start;

smallBook\_n temp;

for (int i = start ; i < end - 1 ; ++i) {

if (strcmp(arr[i].book\_name, arr[end - 1].book\_name) <= 0) {

temp = arr[i];

arr[i] = arr[pivotIndex];

arr[pivotIndex] = temp;

pivotIndex++;

}

}

temp = arr[end - 1];

arr[end - 1] = arr[pivotIndex];

arr[pivotIndex] = temp;

quickSort\_n(arr, start, pivotIndex);

quickSort\_n(arr, pivotIndex + 1, end);

}

} dummy;

int siz = size();

// Create a lighter representative array

smallBook\_n\* books = new smallBook\_n[siz];

for (int i = 0; i < siz; ++i) {

loadBook(i);

books[i].index = i;

strcpy(books[i].book\_name, cur\_book.Book\_name);

}

// Sort the lighter array

dummy.quickSort\_n(books, 0, siz);

// shuffle according to the sorted positions

shelf.clear();

for (int i = 0; i < siz; ++i) {

Book temp1, temp2;

// swap the books

loadBook(books[i].index);

temp1 = cur\_book;

loadBook(i);

temp2 = cur\_book;

cur\_book = temp1;

pushBook(i);

loadBook(books[i].index);

cur\_book = temp2;

pushBook(books[i].index);

// swap the indices on the lighter array

int j;

for (j = i + 1; j < siz and books[j].index != i; j++);

books[j].index = books[i].index;

books[i].index = i;

}

}

void sortByNumber() {

struct smallBook\_n{

int index;

int book\_number;

void quickSort\_n(smallBook\_n\* arr, unsigned int start, unsigned int end) {

if (start + 1 >= end)

return;

int pivotIndex = start;

smallBook\_n temp;

for (int i = start ; i < end - 1 ; ++i) {

if (arr[i].book\_number < arr[end - 1].book\_number) {

temp = arr[i];

arr[i] = arr[pivotIndex];

arr[pivotIndex] = temp;

pivotIndex++;

}

}

temp = arr[end - 1];

arr[end - 1] = arr[pivotIndex];

arr[pivotIndex] = temp;

quickSort\_n(arr, start, pivotIndex);

quickSort\_n(arr, pivotIndex + 1, end);

}

} dummy;

int siz = size();

// Create a lighter representative array

smallBook\_n\* books = new smallBook\_n[siz];

for (int i = 0; i < siz; ++i) {

loadBook(i);

books[i].index = i;

books[i].book\_number = cur\_book.BookNo;

}

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

// Sort the lighter array

dummy.quickSort\_n(books, 0, siz);

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

// shuffle according to the sorted positions

shelf.clear();

for (int i = 0; i < siz; ++i) {

Book temp1, temp2;

// swap the books

loadBook(books[i].index);

temp1 = cur\_book;

loadBook(i);

temp2 = cur\_book;

cur\_book = temp1;

pushBook(i);

loadBook(books[i].index);

cur\_book = temp2;

pushBook(books[i].index);

// swap the indices on the lighter array

int j;

for (j = i + 1; j < siz and books[j].index != i; j++);

books[j].index = books[i].index;

books[i].index = i;

}

}

bool appendBook() {

if (not (shelf.is\_open() and shelf.good()))

return false;

shelf.seekg(0, ios::end);

cur\_book.crypt(password, 10);

if( not shelf.write((char\*)&cur\_book, sizeof(Book)) )

return false;

cur\_book.crypt(password, 10);

return true;

}

int findByNumber(int book\_number) {

int siz = size();

shelf.clear();

shelf.seekg(0, ios::beg);

for (int i = 0; i < siz; i++) {

if( not shelf.read((char\*)&cur\_book, sizeof(Book)) )

return -1;

cur\_book.crypt(password, 10);

if (cur\_book.BookNo == book\_number)

return i;

}

return -1;

}

int findByName(char\* book\_name) {

int siz = size();

shelf.clear();

shelf.seekg(0, ios::beg);

for (int i = 0; i < siz; i++) {

if( not shelf.read((char\*)&cur\_book, sizeof(Book)) )

return -1;

cur\_book.crypt(password, 10);

if (strcmp(cur\_book.Book\_name, book\_name) == 0)

return i;

}

return -1;

}

};

int main() {

int number;

char str[20];

int choice;

Library lib("book.dat");

cout << "Enter password : "; cin.getline(str, 11);

ClearScreen();

lib.setPassword(str);

do {

cout << "\n\nPick an option : ";

cout << "\n\t1 - Append records"

<< "\n\t2 - Modify by Book Number"

<< "\n\t3 - Search by Book Name"

<< "\n\t4 - Delete record"

<< "\n\t5 - Display records"

<< "\n\t6 - Sort by Book Name"

<< "\n\t7 - Sort by Book Number"

<< "\n\t8 - Exit\n";

cin >> choice;

ClearScreen();

switch (choice) {

case 1 :

cout << "Enter book details : ";

lib.cur\_book.enterdetails();

lib.appendBook();

break;

case 2 :

cout << "\nEnter Book Number : "; cin >> number;

number = lib.findByNumber(number);

if (number >= 0) {

cout << "\nIndex " << number << " : ";

lib.cur\_book.showdetails();

lib.cur\_book.enterdetails();

lib.pushBook(number);

} else

cout << "\nNot found";

break;

case 3 :

cout << "\nEnter Book Name : ";

cin.ignore(std::numeric\_limits<streamsize>::max(), '\n');

cin.getline(str, 20);

number = lib.findByName(str);

if (number >= 0) {

cout << "\nIndex " << number << " : ";

lib.cur\_book.showdetails();

}

else

cout << "\nNot found";

break;

case 4 :

cout << "\nEnter index of book : "; cin >> number;

if (not lib.removeBook(number))

cout << "\nCouldn't remove book";

else

cout << "\nBook removed";

break;

case 5 :

lib.display(cout);

break;

case 6 :

lib.sortByName();

lib.display(cout);

break;

case 7 :

lib.sortByNumber();

lib.display(cout);

break;

case 8 :

cout << "\nBYE!!\n";

break;

default :

cout << "\nInvalid option";

break;

}

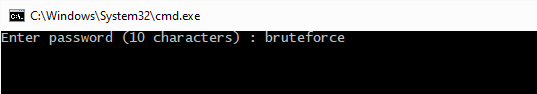
} while (choice != 8);

lib.~Library();

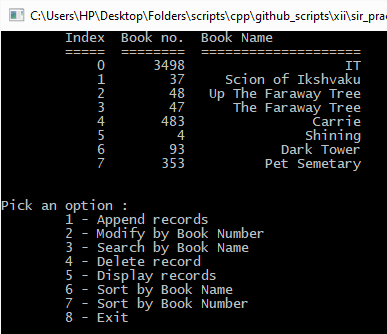
}

Output :

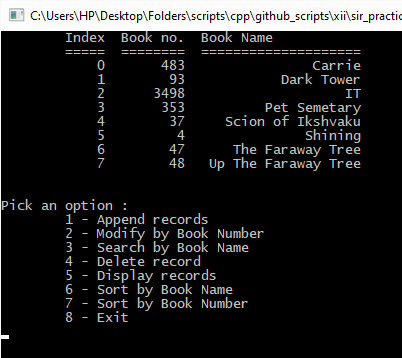
* Enter Password (correct)



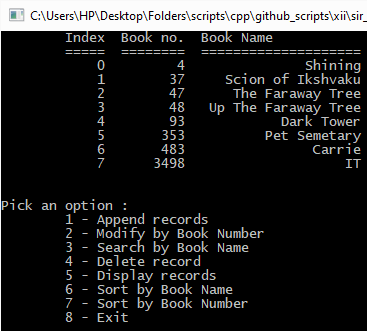
* Display records



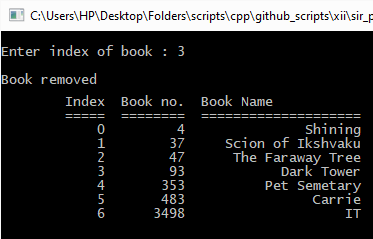
* Sort by Name



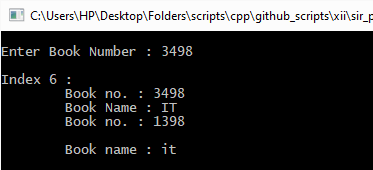
* Sort by Book Number



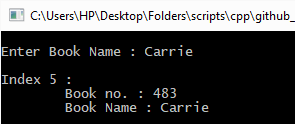
* Delete record



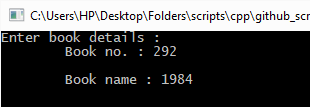
* Modify by Book Number



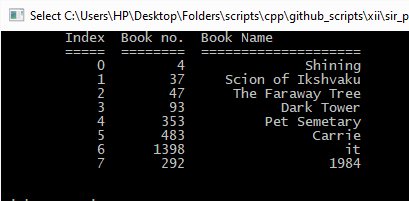
* Search by Book Name



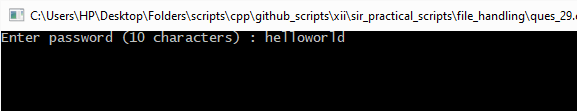
* Add Book



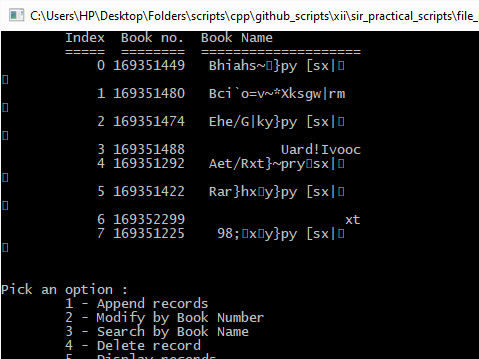
* Display Records



* Enter Password (wrong)



* Display Records



* array.cpp

#include <iostream>

#if !define(LOCAL\_ARRAY\_H\_INCLUDED)

#define LOCAL\_ARRAY\_H\_INCLUDED

class Array {

const unsigned int size;

unsigned int length;

int\* const arr;

public :

int operator[](unsigned int index) {

if (index >= length)

return 0;

return arr[index];

}

Array() : size(0), arr(nullptr) { length = 0;}

Array(unsigned arr\_size) : size(arr\_size), arr(new int[size]) {

length = 0;

if (arr == nullptr)

new(this) Array();

}

unsigned int len() {return length;}

void display(std::ostream& out\_stream) {

out\_stream << '[';

if (length != 0)

out\_stream << arr[0];

for (int i = 1; i < length; ++i)

out\_stream << ", " << arr[i];

out\_stream << ']';

}

bool append(int value) {

if (length >= size) {

length = size;

return false;

}

arr[length] = value;

++length;

return true;

}

bool insert(int value, int index) {

if (index < 0)

index = length + index;

if (index == length)

return append(value);

else if (index > length or index < 0)

return false;

for (int i = length; i > index; --i)

arr[i] = arr[i - 1];

arr[index] = value;

++length;

return true;

}

bool remove(unsigned int index) {

if (index < 0)

index = length - index;

if (index >= length or index < 0)

return false;

for (int i = index; i < length - 1; ++i)

arr[i] = arr[i + 1];

--length;

return true;

}

void selectionSort(bool ascending = true) {

for (int i = length - 1; i > 0; --i) {

int max\_index = 0;

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

if (ascending and arr[j] > arr[max\_index] or not ascending and arr[j] < arr[max\_index])

max\_index = j;

int temp = arr[max\_index];

arr[max\_index] = arr[i];

arr[i] = temp;

}

}

void insertionSort(bool ascending = true) {

for (int i = 1; i < length; ++i) {

int temp = arr[i], j;

for (j = i - 1; ascending and arr[j] < temp or not ascending and arr[j] > temp; --j)

arr[j + 1] = arr[j];

arr[j] = temp;

}

}

void bubbleSort(bool ascending = true) {

for (bool have\_swapped = false; have\_swapped; have\_swapped = false) {

for (int j = 0; j < length - 1; ++j) {

if (ascending and arr[j] > arr[j+1] or not ascending and arr[j] < arr[j+1]) {

int temp = arr[j];

arr[j] = arr[j+1];

arr[j+1] = temp;

have\_swapped = true;

}

}

}

}

int linearSearch(int value) {

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

if (arr[i] == value)

return i;

return -1;

}

int binarySearch(int value) {

unsigned int lower\_bound = 0, upper\_bound = length, mid = (lower\_bound + upper\_bound) / 2;

bubbleSort(true);

while (upper\_bound - lower\_bound > 1) {

if (arr[mid] > value)

upper\_bound = mid;

else if (arr[mid] < value)

lower\_bound = mid + 1;

else

return mid;

mid = (lower\_bound + upper\_bound) / 2;

}

if (upper\_bound <= lower\_bound)

return -1;

}

};

#endif

1. Adolla Link

#include <iostream>

#include <iomanip>

#include <limits>

#include <string.h>

using namespace std;

class Student {

public :

char Name[50];

unsigned int Roll\_no;

Student() {}

void copy(const Student& stu) {

int i;

for (i = 0; stu.Name[i] != '\0' and i < 50; ++i) {

Name[i] = stu.Name[i];

}

Name[i] = '\0';

Roll\_no = stu.Roll\_no;

}

Student(const Student& stu) {

copy(stu);

}

Student& operator=(const Student& stu) {

copy(stu);

return \*this;

}

void display(ostream& out\_stream) {

out\_stream << "\tName : " << Name << '\n';

out\_stream << "\tRoll no. : " << Roll\_no << '\n';

}

void getData(istream& in\_stream, ostream& out\_stream) {

out\_stream << "\tName : ";

in\_stream.ignore(std::numeric\_limits<streamsize>::max(), '\n');

in\_stream.getline(Name, 50);

out\_stream << "\tRoll no. : "; in\_stream >> Roll\_no;

}

};

class Functor {

public :

virtual bool operator()(Student&, Student&) = 0;

};

class NameComp : public Functor {

public :

bool operator()(Student& a, Student& b) {

return strcmp(a.Name, b.Name) < 0;

}

};

class RollComp : public Functor {

public :

bool operator()(Student& a, Student& b) {

return a.Roll\_no < b.Roll\_no;

}

};

class LinkedList {

class Node {

public :

Student student;

Node\* next;

Node\* previous;

const int check;

Node(const Student& stu, Node\* prev) : check(0x1234abcd) {

student = stu;

next = nullptr;

previous = prev;

}

Node(Node\* prev) : check(0x1234abcd) {

next = nullptr;

previous = prev;

}

Node\* makeNext(const Student& stu) {

Node\* new\_next = new Node(stu, this);

if (new\_next != nullptr) next = new\_next;

return new\_next;

}

Node\* makeNext(){

Node\* new\_next = new Node(this);

if (new\_next != nullptr) next = new\_next;

return new\_next;

}

};

Node\* first;

Node\* last;

unsigned int length;

Node\* getNode(int index) {

if (index < 0)

index = index % length + length;

else

index = index % length;

Node\* cur;

if (2 \* index < length) {

cur = first;

for (int i = 1; i <=index; i++) {

cur = cur->next;

}

} else {

cur = last;

for (int i = length - 2; i >= index; i--) {

cur = cur->previous;

}

}

return cur;

}

public:

LinkedList(unsigned int len) {

length = len;

first = new Node(nullptr);

last = first;

for (int i = 1; i < length; ++i) {

last = last->makeNext();

}

}

LinkedList() {

first = nullptr;

last = nullptr;

length = 0;

}

unsigned int len() {

return length;

}

Student& operator[] (int index) {

return getNode(index)->student;

}

bool append(const Student& stu) {

if (last == nullptr) {

last = new Node(stu, nullptr);

first = last;

if (last == nullptr)

return false;

} else {

Node\* new\_last = last->makeNext(stu);

if (new\_last == nullptr) return false;

last = new\_last;

}

++length;

return true;

}

bool prepend(const Student& stu) {

if (last == nullptr) {

first = last = new Node(stu, nullptr);

if (first == nullptr)

return false;

} else {

Node\* new\_node = new Node(stu, nullptr);

if (new\_node == nullptr)

return false;

new\_node->next = first;

first->previous = new\_node;

first = new\_node;

}

++length;

return true;

}

void remove(unsigned int index) {

Node\* cur = getNode(index);

if (cur == nullptr)

return;

cur->next->previous = cur->previous;

cur->previous->next = cur->next;

delete cur;

--length;

}

bool insert(unsigned int index, const Student& stu) {

if (index == length)

return append(stu);

else if (index == 0)

return prepend(stu);

Node\* cur = getNode(index);

if (cur == nullptr)

return false;

Node\* new\_node = new Node(stu, nullptr);

if (new\_node == nullptr)

return false;

new\_node->next = cur;

new\_node->previous = cur->previous;

new\_node->previous->next = new\_node;

cur->previous = new\_node;

++length;

}

void truncate(unsigned int end) {

Node\* cur = last;

Node\* final\_node;

for (int i = length - 1; i >= end; --i) {

final\_node = cur->previous;

delete cur;

cur = final\_node;

}

final\_node->next = nullptr;

last = final\_node;

length = end;

}

void slice(unsigned int start, unsigned int end, unsigned int step = 1) {

if (start >= end)

return;

truncate(end);

Node\* cur = first;

Node\* final\_node;

for (int i = 0; i < start; ++i) {

final\_node = cur->next;

delete cur;

cur = final\_node;

}

final\_node->previous = nullptr;

first = final\_node;

length = end - start;

if (step != 1 or step != 0) {

for (int i = 0, offset = 0; i < length; ++i) {

if (i + offset % step != 0) {

remove(i);

++offset; --i;

}

}

length = length / step;

}

}

void swap(unsigned int i\_1, unsigned int i\_2) {

Node\* temp;

// Ensure that i\_1 <= i\_2

if (i\_1 > i\_2) {

unsigned int temp = i\_1;

i\_1 = i\_2;

i\_2 = temp;

}

if (i\_1 == i\_2) return;

Node\* node\_1 = getNode(i\_1);

Node\* node\_2 = getNode(i\_2);

if (node\_1 == nullptr or node\_2 == nullptr)

return;

// Check if list markers need to be changed

if (i\_1 == 0)

first = node\_2;

else if (i\_1 == length - 1)

last = node\_2;

if (i\_2 == 0)

first = node\_1;

else if (i\_2 == length - 1)

last = node\_1;

// Alternate behaviour for adjacent swaps

if (i\_1 + 1 == i\_2) {

node\_1->next = node\_2->next;

node\_2->previous = node\_1->previous;

node\_1->previous = node\_2;

node\_2->next = node\_1;

if (node\_1->next != nullptr) node\_1->next->previous = node\_1;

if (node\_2->previous != nullptr) node\_2->previous->next = node\_2;

return;

}

// Standard swap behaviour

temp = node\_1->previous;

node\_1->previous = node\_2->previous;

node\_2->previous = temp;

temp = node\_1->next;

node\_1->next = node\_2->next;

node\_2->next = temp;

// Correct neighbours addresses

if (node\_1->next != nullptr) node\_1->next->previous = node\_1;

if (node\_1->previous != nullptr) node\_1->previous->next = node\_1;

if (node\_2->next != nullptr) node\_2->next->previous = node\_2;

if (node\_2->previous != nullptr) node\_2->previous->next = node\_2;

}

void sort(Functor& func) {

class Dummy {

public :

void quickSort(LinkedList& arr, unsigned int start, unsigned int end, Functor& func) {

if (end - start <= 1)

return;

int pivotIndex = start;

for (int i = start; i < end - 1; ++i) {

if (func(arr[i], arr[end - 1])) {

arr.swap(i, pivotIndex);

++pivotIndex;

}

}

arr.swap(pivotIndex, end - 1);

quickSort(arr, start, pivotIndex, func);

quickSort(arr, pivotIndex + 1, end, func);

}

} dummy\_dum\_dum;

dummy\_dum\_dum.quickSort(\*this, 0, length, func);

}

};

class Records {

public :

LinkedList list;

Records() : list() {};

void displayRecord(unsigned int index, ostream& out\_stream) {

list[index].display(out\_stream);

}

void displayAll(ostream& out\_stream) {

out\_stream << "\n\n Index Name Roll no\n";

out\_stream << " ===== ================================================== ==========\n";

for (int i = 0; i < list.len(); ++i)

out\_stream << " " << setw(5) << i << setw(52) << list[i].Name << setw(12) << list[i].Roll\_no << '\n';

out\_stream << "\n\n";

}

void sortByName() {

NameComp dummy\_dum;

list.sort(dummy\_dum);

}

void sortByRoll() {

RollComp dummy\_dum;

list.sort(dummy\_dum);

}

};

int main() {

Records records;

Student stu;

unsigned int number;

char choice;

do {

cout << "\n\nEnter choice : ";

cin >> choice;

switch(choice) {

case '1' :

records.displayAll(cout);

break;

case '2' :

cout << "\nInsert at index (0-" << records.list.len() << ") : "; cin >> number;

cout << "Enter details : \n";

stu.getData(cin, cout);

records.list.insert(number, stu);

break;

case '3' :

cout << "\nRemove from index (0-" << records.list.len() - 1 << ") : "; cin >> number;

records.list.remove(number);

break;

case '4' : {

cout << "\nEnter the name to be searched : ";

cin.ignore(std::numeric\_limits<streamsize>::max(), '\n');

cin.getline(stu.Name, 50);

bool found = false;

for (int i = 0; i < records.list.len() and not found; ++i) {

if (strcmp(stu.Name, records.list[i].Name) == 0) {

found = true;

cout << "Index " << i << " : \n";

records.list[i].display(cout);

}

}

if (not found) {

cout << "No such record found\n";

}

break;

}

case '5' : {

cout << "\nEnter the roll no. to be searched : "; cin >> stu.Roll\_no;

bool found = false;

for (int i = 0; i < records.list.len() and not found; ++i) {

if (stu.Roll\_no == records.list[i].Roll\_no) {

found = true;

cout << "Index " << i << " : \n";

records.list[i].display(cout);

}

}

if (not found) {

cout << "No such record found\n";

}

break;

}

case '6' :

records.sortByName();

break;

case '7' :

records.sortByRoll();

break;

case '8' :

cout << "\t1 - Display Records\n"

<< "\t2 - Insert Record\n"

<< "\t3 - Remove Record\n"

<< "\t4 - Search by Name\n"

<< "\t5 - Search by Roll no.\n"

<< "\t6 - Sort by Name\n"

<< "\t7 - Sort by Roll no.\n"

<< "\t8 - Print this help message\n"

<< "\t9 - Exit\n";

break;

case '9' :

cout << "\tBye\n";

break;

default :

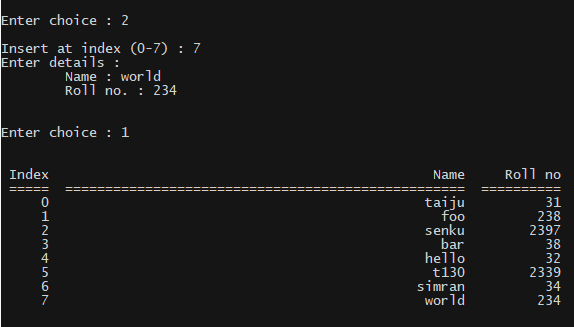
cout << "\tInvalid option, use '8' for help \n\n";

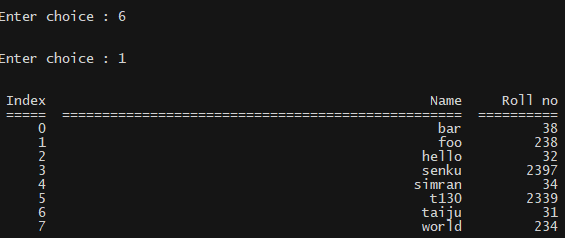
}

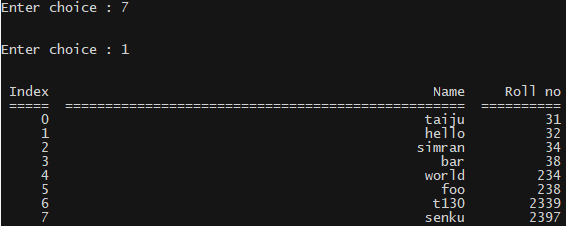
} while (choice != '9');

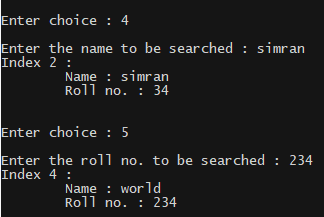
}

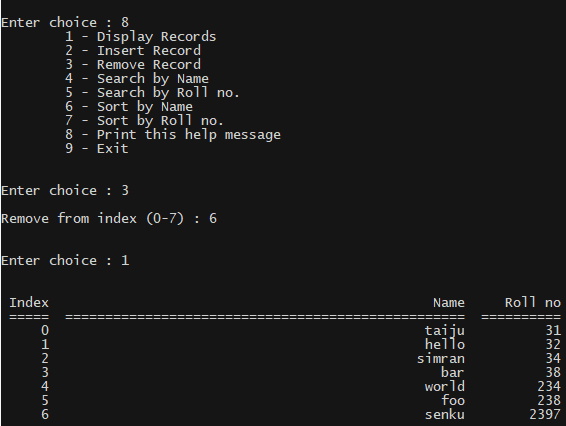
Output :











1. Array Sorting

#include <iostream>

#include "array.cpp"

using namespace std;

int main() {

int n1, n2;

cout << "Max size of arr? "; cin >> n1;

Array arr(n1);

char choice;

do {

cout << "\nEnter your choice :"

<< "\n\t1 - Display array"

<< "\n\t2 - Insert element"

<< "\n\t3 - Remove element"

<< "\n\t4 - Sort with Selecion Sort"

<< "\n\t5 - Sort with Insertion Sort"

<< "\n\t6 - Exit\n";

cin >> choice;

switch(choice) {

case '1' :

cout << "\nArray : ";

arr.display(cout);

cout << '\n';

break;

case '2' :

cout << "\nEnter the index for insertion : "; cin >> n1;

cout << "Enter value : "; cin >> n2;

if (not arr.insert(n2, n1))

cerr << "\nValueError : invalid value provided for function \'bool Array::insert(int,int)\'\n";

break;

case '3' :

cout << "\nEnter the index for insertion : "; cin >> n1;

if (not arr.remove(n1))

cerr << "\nValueError : invalid value provided for function \'bool Array::remove(int)\'\n";

break;

case '4' :

cout << "\nAscending? (y/n) "; cin >> choice;

arr.selectionSort(choice == 'y' or choice == 'Y');

break;

case '5' :

cout << "\nAscending? (y/n) "; cin >> choice;

arr.selectionSort(choice == 'y' or choice == 'Y');

break;

case '6' :

cout << "\nBye!!\n\n";

break;

default :

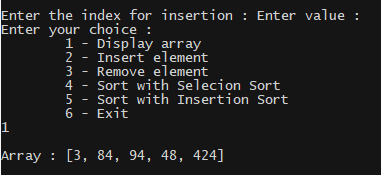
cout << "\nInvalid option, try again\n";

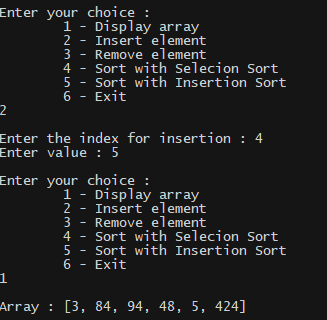
}

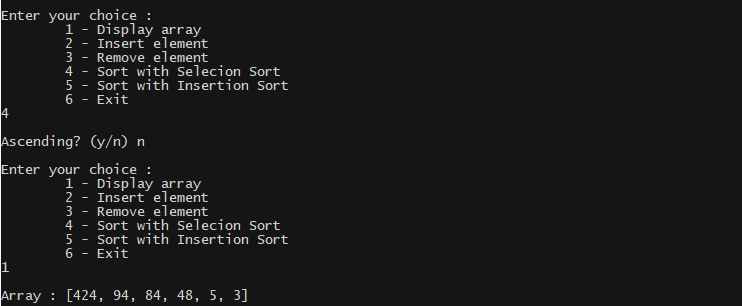
} while (choice != '6');

}

Output :







1. Array Searching

#include <iostream>

#include "array.cpp"

using namespace std;

int main() {

int n1, n2;

cout << "Max size of arr? "; cin >> n1;

Array arr(n1);

char choice;

do {

cout << "\nEnter your choice :"

<< "\n\t1 - Display array"

<< "\n\t2 - Insert element"

<< "\n\t3 - Remove element"

<< "\n\t4 - Linear Search"

<< "\n\t5 - Binary Search"

<< "\n\t6 - Exit\n";

cin >> choice;

switch(choice) {

case '1' :

cout << "\nArray : ";

arr.display(cout);

cout << '\n';

break;

case '2' :

cout << "\nEnter the index for insertion : "; cin >> n1;

cout << "Enter value : "; cin >> n2;

if (not arr.insert(n2, n1))

cerr << "\nValueError : invalid value provided for function \'bool Array::insert(int,int)\'\n";

break;

case '3' :

cout << "\nEnter the index for insertion : "; cin >> n1;

if (not arr.remove(n1))

cerr << "\nValueError : invalid value provided for function \'bool Array::remove(int)\'\n";

break;

case '4' :

cout << "\nEnter the value to be searched : "; cin >> n1;

n2 = arr.linearSearch(n1);

if (n2 == -1)

cout << "Value not found\n";

else

cout << "Value found at index " << n2 << '\n';

break;

case '5' :

cout << "\nEnter the value to be searched : "; cin >> n1;

n2 = arr.binarySearch(n1);

if (n2 == -1)

cout << "Value not found\n";

else

cout << "Value found at index " << n2 << '\n';

break;

case '6' :

cout << "\nBye!!\n\n";

break;

default :

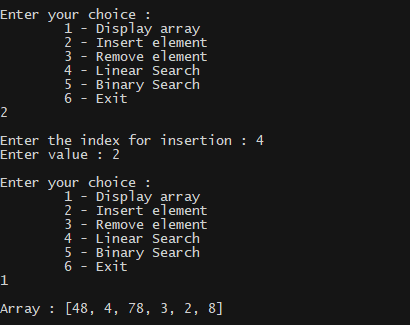
cout << "\nInvalid option, try again\n";

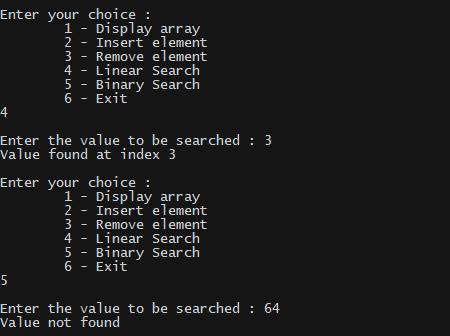
}

} while (choice != '6');

}

Output :





1. Random Weave

#include <iostream>

using namespace std;

void quickSort(int\* arr, int end, bool descending = false, int start = 0) {

if (end - start <= 1)

return;

int pivotPosition = start, temp;

for (int i = start; i < end - 1; ++i) {

if (arr[i] < arr[end - 1] xor descending) {

temp = arr[i];

arr[i] = arr[pivotPosition];

arr[pivotPosition] = temp;

pivotPosition++;

}

}

temp = arr[end - 1];

arr[end - 1] = arr[pivotPosition];

arr[pivotPosition] = temp;

quickSort(arr, pivotPosition, descending, start);

quickSort(arr, end, descending, pivotPosition + 1);

}

int main() {

unsigned int arr1\_size, arr2\_size;

int \*arr1, \*arr2, \*arr3;

cout << "Enter size of array 1 : "; cin >> arr1\_size;

arr1 = new int[arr1\_size];

cout << "Enter Array 1 elements : "; for (int i = 0; i < arr1\_size; ++i) cin >> arr1[i];

cout << "Enter size of array 2 : "; cin >> arr2\_size;

arr2 = new int[arr2\_size];

cout << "Enter Array 2 elements : "; for (int i = 0; i < arr2\_size; ++i) cin >> arr2[i];

quickSort(arr1, arr1\_size, false);

quickSort(arr2, arr2\_size, true);

cout << "\n\nSorted Array 1 : [";

for (int i = 0; i < arr1\_size - 1; ++i) cout << arr1[i] << ", "; cout << arr1[arr1\_size - 1] << "]\n";

cout << "\n\nSorted Array 2 : [";

for (int i = 0; i < arr2\_size - 1; ++i) cout << arr2[i] << ", "; cout << arr2[arr2\_size - 1] << "]\n";

{ // Merge arr1 and arr2

arr3 = new int[arr1\_size + arr2\_size];

int i, j, k;

for (i = 0, j = arr2\_size - 1, k = 0; i < arr1\_size and j > -1; ++k)

arr3[k] = (arr1[i] < arr2[j]) ? arr1[i++] : arr2[j--];

if (i == arr1\_size)

for (;j > -1; ++k, --j)

arr3[k] = arr2[j];

else

for (;i < arr1\_size; ++k, ++i)

arr3[k] = arr1[i];

}

cout << "\n\nMerger Array : [";

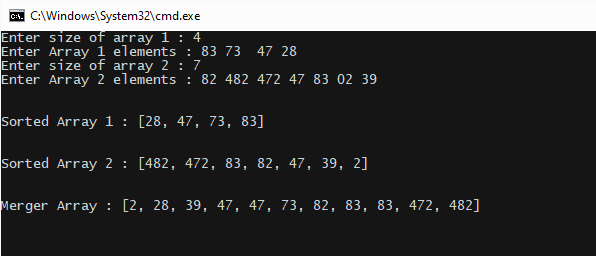
for (int i = 0; i < arr1\_size + arr2\_size - 1; ++i) cout << arr3[i] << ", "; cout << arr3[arr1\_size + arr2\_size - 1] << "]\n";

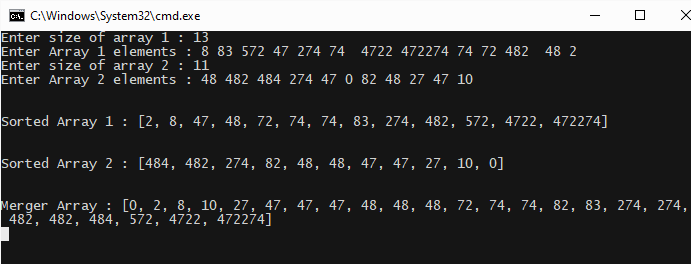
while (getchar() != '\n');

getchar();

}

Output :





1. Flatlanding

#include <iostream>

#include <iomanip>

using namespace std;

int main() {

int arr\_size, \*arr, \*\*arr\_2d;

cout << "Enter array size : "; cin >> arr\_size;

arr = new int[arr\_size];

cout << "Enter array elements : "; for (int i = 0; i < arr\_size; ++i) cin >> arr[i];

arr\_2d = new int\*[arr\_size];

for (int i = 0; i < arr\_size; ++i)

arr\_2d[i] = new int[arr\_size];

for (int i = 0; i < arr\_size; ++i)

for (int j = 0; j < arr\_size; ++j)

arr\_2d[i][j] = (i + j < arr\_size) ? arr[j] : 0;

cout << "\n\nResultant 2d array : ";

for (int i = 0; i < arr\_size; ++i) {

cout << "\n ";

for (int j = 0; j < arr\_size; ++j)

cout << ' ' << setw(10) << arr\_2d[i][j];

}

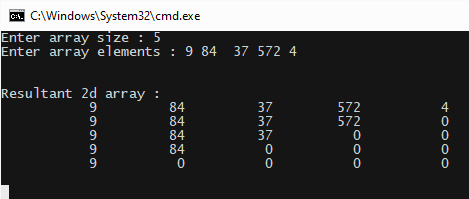
cout << "\n\n";

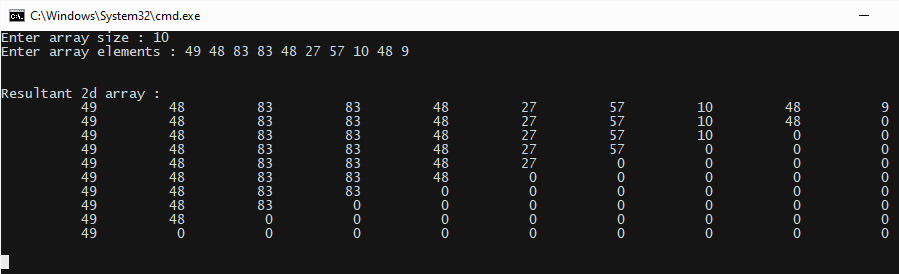
while (getchar() != '\n');

getchar();

}

Output :





1. Ineffective Hiring

#include <iostream>

#include <random>

#include <time.h>

using namespace std;

const int no\_of\_years = 100;

const int max\_emp\_per\_year = 50;

int main() {

srand((unsigned)time(nullptr));

int YEAR[no\_of\_years];

for (int i = 0; i < no\_of\_years; ++i)

/\*cout << (\*/YEAR[i] = max\_emp\_per\_year \* (float(rand()) / RAND\_MAX)/\*) << ", "\*/;

int no\_emp\_sum = 0, exp\_emp\_sum = 0;

cout << "\nYears where no employees were appointed (0-" << no\_of\_years - 1 << " AD) : ";

for (int i = 0; i < no\_of\_years - 5; ++i)

if ((exp\_emp\_sum += YEAR[i]), YEAR[i] == 0)

(cout << i << ", "), no\_emp\_sum++;

for (int i = no\_of\_years - 5; i < no\_of\_years; ++i)

if (YEAR[i] == 0)

(cout << i << ", "), no\_emp\_sum++;

cout << "\nNo. of years where no employees where appointed (0-" << no\_of\_years - 1 << " AD) : " << no\_emp\_sum;

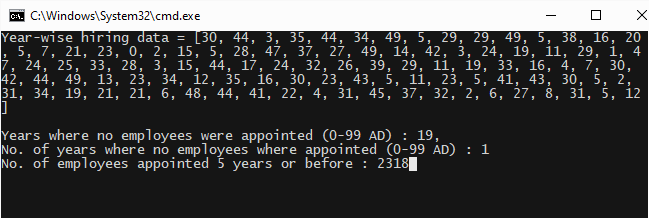
cout << "\nNo. of employees appointed 5 years or before : " << exp\_emp\_sum;

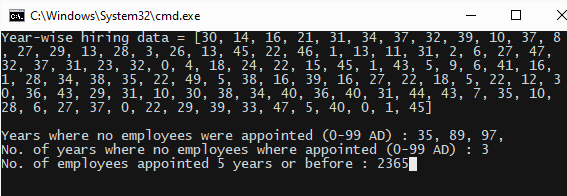
while (getchar() != '\n');

getchar();

}

Output :





1. Array Stack

#include <iostream>

using namespace std;

template <typename T = int>

class StackWithArray {

T\* arr;

const int size;

int top;

public :

StackWithArray() : size(0), arr(nullptr), top(0) {}

StackWithArray(int stack\_size) : size(stack\_size), top(0) {

arr = new T[size];

}

StackWithArray(int stack\_size, T\* old\_arr) : StackWithArray(stack\_size) {

top = size;

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

arr[i] = old\_arr[i];

}

StackWithArray(const StackWithArray<T>& old\_swa) : StackWithArray(old\_swa.size, old\_swa.arr) {

top = old\_swa.top;

}

T pop() {

if (top == 0)

return T();

return arr[--top];

}

bool push(const T& new\_data) {

if (top == size)

return false;

return arr[top++] = new\_data, true;

}

T peek() {

if (top == 0)

return T();

return arr[top - 1];

}

void display(ostream& out\_stream) {

out\_stream << '[';

for (int i = 0; i < top - 1; ++i)

out\_stream << arr[i] << ", ";

if (top != 0)

out\_stream << arr[top - 1];

out\_stream << ']';

}

int max\_size() {return size;}

int current\_size() {return top;}

};

int main () {

int size;

char choice;

cout << "Enter stack size : "; cin >> size;

StackWithArray<int> stack(size);

do {

cout << "\n\nEnter your choice : "

<< "\n\tu - Push to stack"

<< "\n\to - Pop from stack"

<< "\n\te - Peek on the stack"

<< "\n\td - Display stack"

<< "\n\ts - Get current size of stack"

<< "\n\tq - Quit"

<< "\nWhat is your choice? ";

cin >> choice;

switch(choice) {

case 'u' : case 'U' :

int temp;

cout << "\nEnter integer value : "; cin >> temp;

if(not stack.push(temp))

cout << "AllocationError : Stack is full";

break;

case 'o' : case 'O' :

cout << "\nValue that was on the stack : " << stack.pop();

break;

case 'e' : case 'E' :

cout << "\nValue that is on the stacl : " << stack.peek();

break;

case 'd' : case 'D' :

cout << "\nStack contents : "; stack.display(cout);

break;

case 's' : case 'S' :

cout << "\nCurrent size of stack : " << stack.current\_size();

break;

case 'q' : case 'Q' : break;

default :

cout << "\nInvalid option. Please try sgain";

break;

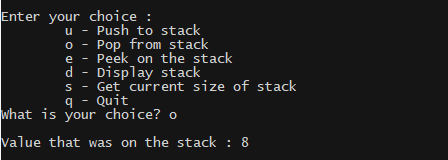
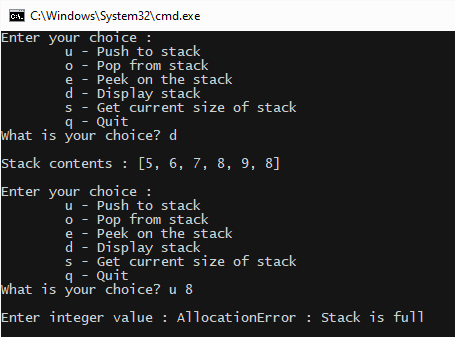
}

} while (choice != 'q' and choice != 'Q');

cout << "\nBYE!";

}

Output :



1. Array Queue

#include <iostream>

using namespace std;

template <typename T = int>

class QueueWithArray {

T\* arr;

const int length;

int top;

int tail;

public :

QueueWithArray() : length(0), arr(nullptr), top(0), tail(0) {}

QueueWithArray(int queue\_length) : length(queue\_length), top(-1), tail(-1) {

arr = new T[length];

}

QueueWithArray(int queue\_length, T\* old\_arr) : QueueWithArray(queue\_length) {

tail = top = 0;

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

arr[i] = old\_arr[i];

}

QueueWithArray(const QueueWithArray<T>& old\_qwa) : QueueWithArray(old\_qwa.length, old\_qwa.arr) {

top = old\_qwa.top;

tail = old\_qwa.tail;

}

T pop() {

if (top == -1)

return T();

int old\_first = top;

top = (top + 1) % length;

if (top == tail)

top = (tail = -1);

return arr[old\_first];

}

bool append(const T& new\_data) {

if (tail == -1)

return (tail = 1), (top = 0), arr[0] = new\_data, true;

if (tail == top)

return false;

arr[tail] = new\_data;

tail = (tail + 1) % length;

return true;

}

T peek() {

if (top == -1)

return T();

return arr[top];

}

void display(ostream& out\_stream) {

out\_stream << '[';

if (tail != -1) {

for (int i = top; i < tail + length - 1; ++i)

out\_stream << arr[i % length] << ", ";

out\_stream << arr[(tail + length - 1) % length];

}

out\_stream << ']';

}

int max\_length() {return length;}

int current\_length() {

if (tail == -1)

return 0;

if (tail == top)

return length;

return (tail - top + length) % length;

}

};

int main () {

int length;

char choice;

cout << "Enter queue length : "; cin >> length;

QueueWithArray<int> queue(length);

do {

cout << "\n\nEnter your choice : "

<< "\n\ta - Append to the tail of the queue"

<< "\n\tp - Pop from the top of the queue"

<< "\n\te - Peek on the top of the queue"

<< "\n\td - Display queue"

<< "\n\ts - Get current length of queue"

<< "\n\tq - Quit"

<< "\nWhat is your choice? ";

cin >> choice;

switch(choice) {

case 'a' : case 'A' :

int temp;

cout << "\nEnter integer value : "; cin >> temp;

if(not queue.append(temp))

cout << "AllocationError : Queue is full";

break;

case 'p' : case 'P' :

cout << "\nValue that was on the queue : " << queue.pop();

break;

case 'e' : case 'E' :

cout << "\nValue that is on the queue : " << queue.peek();

break;

case 'd' : case 'D' :

cout << "\nQueue contents : "; queue.display(cout);

break;

case 's' : case 'S' :

cout << "\nCurrent length of queue : " << queue.current\_length();

break;

case 'q' : case 'Q' : break;

default :

cout << "\nInvalid option. Please try sgain";

break;

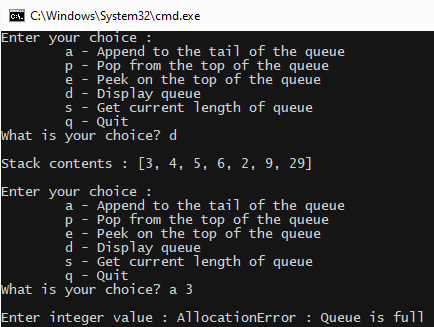
}

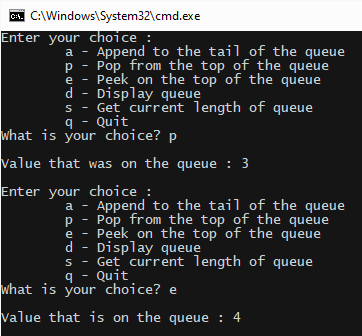
} while (choice != 'q' and choice != 'Q');

cout << "\nBYE!";

}

Output :





1. Linked Stack

#include <iostream>

using namespace std;

template <typename T = int>

class LinkedStack {

struct Node {

T data;

Node\* previous;

Node\* next;

Node() : data(), previous(nullptr), next(nullptr) {}

Node(const T& new\_data, Node\* prev = nullptr, Node\* nxt = nullptr) : data(new\_data), previous(prev), next(nxt) {}

~Node() {

previous = next = nullptr;

data.~T();

}

Node\* makeNext(const T& new\_data) {

Node\* new\_next = new Node(new\_data, this);

if (new\_next != nullptr)

next = new\_next;

return new\_next;

}

};

Node\* top;

Node\* bottom;

public :

LinkedStack() : top(nullptr), bottom(nullptr) {}

LinkedStack(int stack\_size, T\* old\_arr) {

bottom = top = new Node(old\_arr[0]);

for (int i = 1; i < stack\_size; ++i)

top = top->makeNext(old\_arr[i]);

}

LinkedStack(const LinkedStack<T>& old\_ls) {

Node\* leach\_point = old\_ls.top;

bottom = top = new Node(leach\_point->data);

leach\_point = leach\_point->next;

while (leach\_point != nullptr) {

bottom = bottom->makeNext(leach\_point->data);

leach\_point = leach\_point->next;

}

}

bool isEmpty() {

return top == nullptr;

}

int size() {

int size = 0;

Node\* read\_head = bottom;

while (read\_head != nullptr)

read\_head = read\_head->next, size++;

return size;

}

T pop() {

if (top == nullptr)

return T();

Node\* old\_top = top;

if (top == bottom) {

top = bottom = nullptr;

} else {

top = top->previous;

top->next = nullptr;

}

return old\_top->data;

}

bool push(const T& new\_data) {

if (top == nullptr) {

top = bottom = new Node(new\_data);

return top != nullptr;

}

Node\* new\_top = top->makeNext(new\_data);

if (new\_top == nullptr)

return false;

top = new\_top;

return true;

}

T peek() {

if (top == nullptr)

return T();

return top->data;

}

bool isOk() {

if (bottom == nullptr)

return true;

Node\* read\_head = bottom;

while (read\_head != nullptr and read\_head != top)

read\_head = read\_head->next;

if (read\_head == nullptr)

return false;

return true;

}

void display(ostream& out\_stream) {

if (not isOk())

out\_stream << "Stack Corrupted";

Node\* read\_head = bottom;

out\_stream << '[';

if (bottom != nullptr) {

out\_stream << read\_head->data;

read\_head = read\_head->next;

while (read\_head != nullptr) {

out\_stream << ", " << read\_head->data;

read\_head = read\_head->next;

}

}

out\_stream << ']';

}

};

int main () {

char choice;

LinkedStack<int> stack;

do {

cout << "\n\nEnter your choice : "

<< "\n\tu - Push to stack"

<< "\n\to - Pop from stack"

<< "\n\te - Peek on the stack"

<< "\n\td - Display stack"

<< "\n\ts - Get current size of stack"

<< "\n\tq - Quit"

<< "\nWhat is your choice? ";

cin >> choice;

switch(choice) {

case 'u' : case 'U' :

int temp;

cout << "\nEnter integer value : "; cin >> temp;

if(not stack.push(temp))

cout << "AllocationError : Stack is full";

break;

case 'o' : case 'O' :

cout << "\nValue that was on the stack : " << stack.pop();

break;

case 'e' : case 'E' :

cout << "\nValue that is on the stacl : " << stack.peek();

break;

case 'd' : case 'D' :

cout << "\nStack contents : "; stack.display(cout);

break;

case 's' : case 'S' :

cout << "\nCurrent size of stack : " << stack.size();

break;

case 'q' : case 'Q' : break;

default :

cout << "\nInvalid option. Please try sgain";

break;

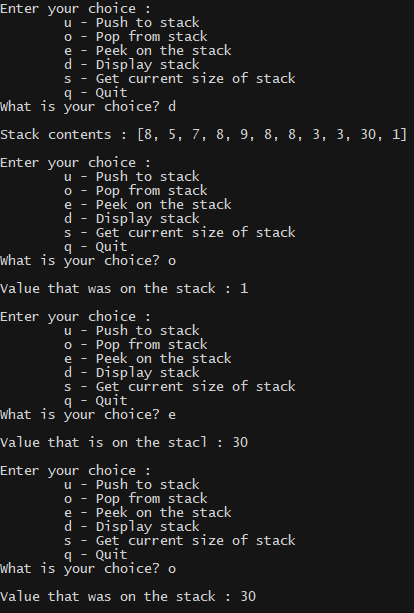
}

} while (choice != 'q' and choice != 'Q');

cout << "\nBYE!";

}

Output :



1. Linked Queue

#include <iostream>

using namespace std;

template <typename T = int>

class LinkedQueue {

struct Node {

T data;

Node\* previous;

Node\* next;

Node() : data(), previous(nullptr), next(nullptr) {}

Node(const T& new\_data, Node\* prev = nullptr, Node\* nxt = nullptr) : data(new\_data), previous(prev), next(nxt) {}

~Node() {

previous = next = nullptr;

data.~T();

}

Node\* makeNext(const T& new\_data) {

Node\* new\_next = new Node(new\_data, this);

if (new\_next != nullptr)

next = new\_next;

return new\_next;

}

};

Node\* top;

Node\* tail;

public :

LinkedQueue() : top(nullptr), tail(nullptr) {}

LinkedQueue(int queue\_length, T\* old\_arr) {

tail = top = new Node(old\_arr[0]);

for (int i = 1; i < queue\_length; ++i)

tail = tail->makeNext(old\_arr[i]);

}

LinkedQueue(const LinkedQueue<T>& old\_lq) {

Node\* leach\_point = old\_lq.top;

tail = top = new Node(leach\_point->data);

leach\_point = leach\_point->next;

while (leach\_point != nullptr) {

tail = tail->makeNext(leach\_point->data);

leach\_point = leach\_point->next;

}

}

int length() {

int length = 0;

Node\* read\_head = top;

while (read\_head != nullptr) {

read\_head = read\_head->next;

length++;

}

}

T pop() {

if (top == nullptr)

return T();

Node\* old\_top = top;

if (top == tail) {

top = tail = nullptr;

} else {

top = top->next;

top->previous = nullptr;

}

return old\_top->data;

}

bool append(const T& new\_data) {

if (tail == nullptr) {

top = tail = new Node(new\_data);

return tail != nullptr;

}

Node\* new\_tail = tail->makeNext(new\_data);

if (new\_tail == nullptr)

return false;

tail = new\_tail;

return true;

}

T peek() {

if (tail == nullptr)

return T();

return tail->data;

}

bool isOk() {

if (top == nullptr)

return true;

Node\* read\_head = top;

while (read\_head != nullptr and read\_head != tail)

read\_head = read\_head->next;

if (read\_head == nullptr)

return false;

return true;

}

void display(ostream& out\_stream) {

if (not isOk())

out\_stream << "Queue Corrupted";

out\_stream << '[';

Node\* read\_head = top;

if (read\_head != nullptr) {

out\_stream << read\_head->data;

read\_head = read\_head->next;

while (read\_head != nullptr) {

out\_stream << ", " << read\_head->data;

read\_head = read\_head->next;

}

}

out\_stream << ']';

}

};

int main () {

char choice;

LinkedQueue<int> queue;

do {

cout << "\n\nEnter your choice : "

<< "\n\tu - Place at the tail of the queue"

<< "\n\to - Pop from the top of the queue"

<< "\n\te - Peek on the top of the queue"

<< "\n\td - Display queue"

<< "\n\ts - Get current length of queue"

<< "\n\tq - Quit"

<< "\nWhat is your choice? ";

cin >> choice;

switch(choice) {

case 'u' : case 'U' :

int temp;

cout << "\nEnter integer value : "; cin >> temp;

if(not queue.append(temp))

cout << "AllocationError : Queue is unable to allocate for more data";

break;

case 'o' : case 'O' :

cout << "\nValue that was on the queue : " << queue.pop();

break;

case 'e' : case 'E' :

cout << "\nValue that is on the queue : " << queue.peek();

break;

case 'd' : case 'D' :

cout << "\nQueue contents : "; queue.display(cout);

break;

case 's' : case 'S' :

cout << "\nCurrent length of queue : " << queue.length();

break;

case 'q' : case 'Q' : break;

default :

cout << "\nInvalid option. Please try sgain";

break;

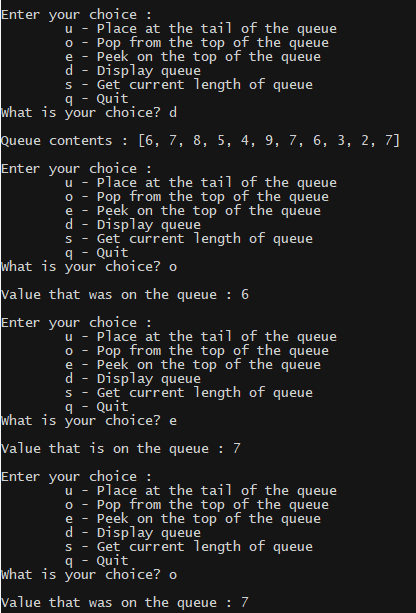
}

} while (choice != 'q' and choice != 'Q');

cout << "\nBYE!";

}

Output

:

1. Flatlanding

#include <iostream>

using namespace std;

template <typename T = int>

class LinkedStack {

struct Node {

T data;

Node\* previous;

Node\* next;

Node() : data(), previous(nullptr), next(nullptr) {}

Node(const T& new\_data, Node\* prev = nullptr, Node\* nxt = nullptr) : data(new\_data), previous(prev), next(nxt) {}

~Node() {

previous = next = nullptr;

data.~T();

}

Node\* makeNext(const T& new\_data) {

Node\* new\_next = new Node(new\_data, this);

if (new\_next != nullptr)

next = new\_next;

return new\_next;

}

};

Node\* top;

Node\* bottom;

public :

LinkedStack() : top(nullptr), bottom(nullptr) {}

LinkedStack(int stack\_size, T\* old\_arr) {

bottom = top = new Node(old\_arr[0]);

for (int i = 1; i < stack\_size; ++i)

top = top->makeNext(old\_arr[i]);

}

LinkedStack(const LinkedStack<T>& old\_ls) {

Node\* leach\_point = old\_ls.top;

bottom = top = new Node(leach\_point->data);

leach\_point = leach\_point->next;

while (leach\_point != nullptr) {

bottom = bottom->makeNext(leach\_point->data);

leach\_point = leach\_point->next;

}

}

bool isEmpty() {

return top == nullptr;

}

int size() {

int size = 0;

Node\* read\_head = bottom;

while (read\_head != nullptr)

read\_head = read\_head->next, size++;

return size;

}

T pop() {

if (top == nullptr)

return T();

Node\* old\_top = top;

if (top == bottom) {

top = bottom = nullptr;

} else {

top = top->previous;

top->next = nullptr;

}

return old\_top->data;

}

bool push(const T& new\_data) {

if (top == nullptr) {

top = bottom = new Node(new\_data);

return top != nullptr;

}

Node\* new\_top = top->makeNext(new\_data);

if (new\_top == nullptr)

return false;

top = new\_top;

return true;

}

T peek() {

if (top == nullptr)

return T();

return top->data;

}

bool isOk() {

if (bottom == nullptr)

return true;

Node\* read\_head = bottom;

while (read\_head != nullptr and read\_head != top)

read\_head = read\_head->next;

if (read\_head == nullptr)

return false;

return true;

}

void display(ostream& out\_stream) {

if (not isOk())

out\_stream << "Stack Corrupted";

Node\* read\_head = bottom;

out\_stream << '[';

if (bottom != nullptr) {

out\_stream << read\_head->data;

read\_head = read\_head->next;

while (read\_head != nullptr) {

out\_stream << ", " << read\_head->data;

read\_head = read\_head->next;

}

}

out\_stream << ']';

}

};

int main() {

long long number, smol\_factor;

LinkedStack<long long> factors;

cout << "Range : [-2^63, 2^63 - 1] ~= [-9.22 quintillion, 9.22 quintillion]";

cout << "\nEnter number to be factorized : "; cin >> number;

if (number < 0) {

factors.push(-1);

number \*= -1;

} else

factors.push(1);

while (number > 1) {

for (smol\_factor = 2 ; number % smol\_factor != 0; ++smol\_factor);

number /= smol\_factor;

factors.push(smol\_factor);

}

cout << "Factorization = " << factors.pop();

while(not factors.isEmpty()) cout << " x " << factors.pop();

while(getchar() != '\n');

getchar();

}

Output :

