До попереднього завдання (клас для роботи з масивами стрічок) додати такі можливості:

1. Перевантажити оператори введення/виведення масиву стрічок.
2. Перевантажити логічні оператори (порівняння елементів масиву): <, <=, >, >=, !=, ==. Логічні оператори для порівняння повинні працювати аналогічно стандартним операторам.
3. Перезавантажити оператори індексування та (), щоб була можливість заміняти окремі елементи масиву, наприклад, масив[індекс](нове\_значення) — arr[2]("aedwe")

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include<Windows.h>

#include<math.h>

#include<string.h>

#include<stdlib.h>

#include<direct.h>

#include<io.h>

#include<time.h>

#include<conio.h>

using namespace std;

class String

{

private:

char\*\* arr;

int k;

public:

String()

{

arr = NULL;

k = 0;

}

String(const String& obj)

{

arr = new char\* [obj.k];

k = obj.k;

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

{

arr[i] = new char[strlen(obj.arr[i]) + 1];

strcpy(arr[i], obj.arr[i]);

}

}

~String()

{

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

{

if (arr[i] != NULL)

delete[] arr[i];

}

if (arr != NULL)

delete[] arr;

}

void Delete\_memory()

{

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

{

if (arr[i] != NULL)

delete[] arr[i];

}

if (arr != NULL)

delete[] arr;

}

char\* String\_for\_anything()

{

char\* tmp = NULL;

int k = 0;

do

{

char s;

s = \_getch();

cout << s;

if (s == 13)

break;

tmp = (char\*)realloc(tmp, sizeof(char) \* (k + 1));

tmp[k] = s;

k++;

} while (true);

tmp = (char\*)realloc(tmp, sizeof(char) \* (k + 1));

tmp[k] = '\0';

cout << endl;

return tmp;

}

void Set\_String()

{

Delete\_memory();

cout << "Введіть скільки рядків хочете написати ->";

cin >> k;

String\_memory(k);

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

{

cout << "Введіть " << i + 1 << " рядок:\n";

char\* tmp = String\_for\_anything();

arr[i] = new char[strlen(tmp) + 1];

strcpy(arr[i], tmp);

delete[] tmp;

}

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

}

void Print\_String()

{

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

{

cout << arr[i] << endl;

}

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

}

void String\_memory(int s)

{

arr = new char\* [s];

}

void Sort()

{

bool b;

do {

b = false;

for (int i = 0; i < k - 1; i++)

{

if (strcmp(arr[i], arr[i + 1]) == 1)

{

char\* temp = new char[strlen(arr[i]) + 1];

strcpy(temp, arr[i]);

if (arr[i] != NULL)

delete[] arr[i];

arr[i] = new char[strlen(arr[i + 1]) + 1];

strcpy(arr[i], arr[i + 1]);

if (arr[i + 1] != NULL)

delete[] arr[i + 1];

arr[i + 1] = new char[strlen(temp) + 1];

strcpy(arr[i + 1], temp);

b = true;

delete[] temp;

}

}

} while (b);

cout << "Відсортовано!" << endl;

}

bool Is\_Ne\_Is(char\* str)

{

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

{

if (strcmp(arr[i], str) == 0)

return true;

}

return false;

}

String Add\_String(char\* str)

{

String tmp;

tmp.String\_memory(k + 1);

tmp.k = k + 1;

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

{

tmp.arr[i] = new char[strlen(arr[i]) + 1];

strcpy(tmp.arr[i], arr[i]);

}

tmp.arr[k] = new char[strlen(str) + 1];

strcpy(tmp.arr[k], str);

return tmp;

}

String operator+(const String& obj)

{

String tmp;

if (this->k < obj.k)

{

tmp.k = obj.k;

tmp.String\_memory(obj.k);

for (int i = 0; i < this->k; i++)

{

tmp.arr[i] = new char[strlen(this->arr[i]) + strlen(obj.arr[i]) + 2];

strcpy(tmp.arr[i], this->arr[i]);

strcat(tmp.arr[i], " ");

strcat(tmp.arr[i], obj.arr[i]);

}

for (int i = this->k; i < obj.k; i++)

{

tmp.arr[i] = new char[strlen(obj.arr[i]) + 1];

strcpy(tmp.arr[i], obj.arr[i]);

}

return tmp;

}

else if (this->k > obj.k)

{

tmp.k = this->k;

tmp.String\_memory(this->k);

for (int i = 0; i < obj.k; i++)

{

tmp.arr[i] = new char[strlen(this->arr[i]) + strlen(obj.arr[i]) + 2];

strcpy(tmp.arr[i], this->arr[i]);

strcat(tmp.arr[i], " ");

strcat(tmp.arr[i], obj.arr[i]);

}

for (int i = obj.k; i < this->k; i++)

{

tmp.arr[i] = new char[strlen(arr[i]) + 1];

strcpy(tmp.arr[i], this->arr[i]);

}

return tmp;

}

else

{

tmp.k = this->k;

tmp.String\_memory(this->k);

for (int i = 0; i < obj.k; i++)

{

tmp.arr[i] = new char[strlen(this->arr[i]) + strlen(obj.arr[i]) + 2];

strcpy(tmp.arr[i], this->arr[i]);

strcat(tmp.arr[i], " ");

strcat(tmp.arr[i], obj.arr[i]);

}

return tmp;

}

}

String operator\*(const String& obj)

{

String tmp;

tmp.k = this->k;

tmp.String\_memory(tmp.k);

for (int i = 0; i < this->k; i++)

{

tmp.arr[i] = new char[strlen(this->arr[i]) + 1];

strcpy(tmp.arr[i], this->arr[i]);

}

for (int i = 0; i < obj.k; i++)

{

if (!tmp.Is\_Ne\_Is(obj.arr[i]))

{

tmp = tmp.Add\_String(obj.arr[i]);

}

}

return tmp;

}

String& operator=(const String& obj)

{

Delete\_memory();

arr = new char\* [obj.k];

k = obj.k;

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

{

arr[i] = new char[strlen(obj.arr[i]) + 1];

strcpy(arr[i], obj.arr[i]);

}

return \*this;

}

String& operator++()

{

cout << "Введіть новий рядок:\n";

\*this = Add\_String(String\_for\_anything());

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

return \*this;

}

String operator++(int q)

{

String tmp;

tmp = \*this;

cout << "Введіть новий рядок:\n";

\*this = Add\_String(String\_for\_anything());

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

return tmp;

}

String operator--()

{

String tmp;

tmp.k = this->k - 1;

tmp.String\_memory(tmp.k);

for (int i = 0; i < tmp.k; i++)

{

tmp.arr[i] = new char[strlen(this->arr[i]) + 1];

strcpy(tmp.arr[i], this->arr[i]);

}

\*this = tmp;

return \*this;

}

String operator--(int q)

{

String tmp, tmp2;

tmp2 = \*this;

tmp.k = this->k - 1;

tmp.String\_memory(tmp.k);

for (int i = 0; i < tmp.k; i++)

{

tmp.arr[i] = new char[strlen(this->arr[i]) + 1];

strcpy(tmp.arr[i], this->arr[i]);

}

\*this = tmp;

return tmp2;

}

char\* operator[](int index)

{

if (index >= 0 && index < this->k)

return this->arr[index];

else

{

cout << "Ви вийшли за межі масиву!\n";

cout << "Буде повернутий нульовий елемент!\n";

return this->arr[0];

}

}

friend ostream& operator<<(ostream& out, const String& string);

friend istream& operator>>(istream& in, String& obj);

void operator()(const char\* str, int index)

{

if (index < this->k)

{

if (arr[index] != NULL)

delete[] arr[index];

arr[index] = new char[strlen(str) + 1];

strcpy(arr[index], str);

}

else

cout << "Ви вийшли за межі масиву!\n";

}

bool operator>(const String& obj)

{

bool b = true;

int min;

if (this->k < obj.k)

min = k;

else

min = obj.k;

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

{

if (strcmp(this->arr[i], obj.arr[i]) != 0)

b = false;

}

if (b)

{

if (this->k > obj.k)

return true;

else

return false;

}

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

{

if (strcmp(this->arr[i], obj.arr[i]) < 1)

return false;

}

if (this->k == obj.k)

return true;

else

return false;

}

bool operator<(const String& obj)

{

bool b = true;

int min;

if (this->k < obj.k)

min = k;

else

min = obj.k;

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

{

if (strcmp(this->arr[i], obj.arr[i]) != 0)

b = false;

}

if (b)

{

if (this->k < obj.k)

return true;

else

return false;

}

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

{

if (strcmp(this->arr[i], obj.arr[i]) > -1)

return false;

}

if (this->k == obj.k)

return true;

else

return false;

}

bool operator<=(const String& obj)

{

int min;

if (this->k < obj.k)

min = k;

else

min = obj.k;

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

{

if (strcmp(this->arr[i], obj.arr[i]) > 0)

return false;

}

if (this->k <= obj.k)

return true;

else

return false;

}

bool operator>=(const String& obj)

{

int min;

if (this->k < obj.k)

min = k;

else

min = obj.k;

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

{

if (strcmp(this->arr[i], obj.arr[i]) < 0)

return false;

}

if (this->k >= obj.k)

return true;

else

return false;

}

bool operator==(const String& obj)

{

if (this->k != obj.k)

return false;

for (int i = 0; i < this->k; i++)

{

if (strcmp(this->arr[i], obj.arr[i]) != 0)

return false;

}

return true;

}

bool operator!=(const String& obj)

{

if (this->k != obj.k)

return true;

for (int i = 0; i < this->k; i++)

{

if (strcmp(this->arr[i], obj.arr[i]) != 0)

return true;

}

return false;

}

};

ostream& operator<<(ostream& out, const String& string)

{

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

for (int i = 0; i < string.k; i++)

{

out << string.arr[i] << endl;

}

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

return out;

}

istream& operator>>(istream& in, String& obj)

{

obj.Delete\_memory();

cout << "Введіть скільки рядків хочете написати ->";

in >> obj.k;

obj.String\_memory(obj.k);

for (int i = 0; i < obj.k; i++)

{

cout << "Введіть " << i + 1 << " рядок:\n";

char\* tmp = obj.String\_for\_anything();

obj.arr[i] = new char[strlen(tmp) + 1];

strcpy(obj.arr[i], tmp);

delete[] tmp;

}

return in;

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

String a, b;

cin >> a;

cout << a;

/\*cin >> b;

if (a != b)

cout << "YES\n";

else

cout << "NO\n";\*/

a("12345",2);

cout << a;

/\*b = a++;

a.Print\_String();

b.Print\_String();\*/

/\*b = a--;

a.Print\_String();

b.Print\_String();\*/

//cout << a[2] << endl;

return 0;

}