

**Міністерство освіти і науки України**

**Національний технічний університет України**

**«Київський політехнічний інститут»**

# Лабораторна робота № 4

**Розробка класу рядка**

Виконала студентка групи:КВ-11

ПІБ: Шевчук Я.О.

Перевірив:\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Київ 2022**

**Завдання:**

Розробити шаблон класу узагальненого рядка, що може містити символи

довільного типу (тобто, String<T>). Передбачити можливість зберігання

рядка довільної довжини. В загальному випадку передбачається, що рядок

може включати нульові символи (тобто, значення T()).

Необхідно реалізувати наступні операції:

* Конструювання:

o за замовченням (порожній рядок);

o копіювання;

o переміщення;

o ініціалізація шляхом повторення символу заданої кількості

разів;

o ініціалізація вказівником на масив символів (при цьому,

розмір рядка не передається, а кінцем рядка вважається перше

включення нульового символу (значення T());

o ініціалізація вказівниками на початок (перший елемент) та

кінець (наступний після останнього елементу) масиву

символів (при цьому, допускається порожній масив

(початок == кінець, а також масив може містити нульові

елементи, але необхідно перевіряти, що початок

розташовано не пізніше кінця));

o перетворення із рядка з іншим типом символів (перетворення

символів може бути реалізоване за допомогою конструкції

static\_cast).

* Присвоєння копіювання та переміщення.
* Отримання довжини рядка.
* Перевірка, чи є рядок порожнім.
* Очищення рядка (робить його порожнім).
* Доступ до символу за індексом (як для читання, так і для запису).
* Отримання «підрядка»
* Конкатенація рядків та додавання одного символу
* Повторення рядка задану кількість разів.
* Порівняння рядків (на (не)рівніть: ==, !=, >, <)
* Трансформації рядка за допомогою функціонального об’єкту
* Функцію створення рядка відповідного типу із масиву символів
* Для рядка звичайних символів (char) додати також операції вводу

з клавіатури та виводу на екран.

Код програми:

#include <iostream>

#include <vector>

#include <string.h>

**using** **namespace** std;

**template** <**class** T>

**class** String

{

**private**:

vector<T> str;

**public**:

String()

{

vector<T> s;

str = s;

}

String(**int** n)

{

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

str.push\_back(1);

}

String(**int** n, T value)

{

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

str.push\_back(value);

}

String(String& stri)

{

**for** (**int** i = 0; i < stri.Length(); i++)

{

str.push\_back(stri[i]);

}

}

String(String&& stri)

{

**for** (**int** i = 0; i < stri.Length(); i++)

{

str.push\_back((T)stri[i]);

}

stri.Clear();

}

**template**<**typename** T2>

String(**const** T2\* array, **int** n)

{

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

{

str.push\_back((T)array[i]);

}

}

**template** <**typename** T2>

String(String<T2>& stri)

{

**for** (**int** i = 0; i < stri.Length(); i++)

{

str.push\_back((T)stri[i]);

}

}

vector<T> GetVect() **const**

{

**return** str;

}

string GetStr() **const**

{

string s;

**for** (**int** i = 0; i < str.size(); i++)

s += str[i];

**return** s;

}

**int** Length()

{

**return** str.size();

}

**void** Clear()

{

str.clear();

}

T& **operator**[](**int** i)

{

**if** (i >= 0)

**return** str[i];

**else** **return** str[str.size() + i];

}

**bool** IsClear()

{

**if** (str.empty())

**return** **true**;

**else** **return** **false**;

}

~String()

{

str.clear();

}

String SubString(**int** beg, **int** len)

{

**if** (beg + len > str.size())

**return** \***this**;

String<T> result;

**for** (**int** i = 0; i < len; i++)

{

result.str.push\_back(str[i + beg]);

}

**return** result;

}

**template** <**typename** T2>

String **operator** = (String<T2> s)

{

String<T> result;

**for** (**int** i = 0; i < s.Length(); i++)

{

result.str.push\_back((T)s[i]);

}

}

**void** **operator**+(T elem)

{

str.push\_back(elem);

}

**template** <**typename** T2>

**void** **operator**+(T2 st)

{

**for** (**int** i = 0; i < st.Length(); i++)

{

str.push\_back((T)st[i]);

}

}

**void** **operator**\*(**int** n)

{

String<T> tmp(\***this**);

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

{

\***this** + tmp;

}

}

**friend** ostream& **operator**<<(ostream& os, String& st)

{

**for** (**int** i = 0; i < st.Length(); i++)

cout << st[i];

cout << endl;

**return** os;

}

**friend** istream& **operator** >> (istream& is, String& st)

{

cout << "How much elements do you want to input?" << endl;

**int** n; cin >> n;

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

{

cout << i + 1 << ": ";

T elem;

cin >> elem;

st + elem;

}

**return** is;

}

**bool** **operator**==( String& st)

{

**if** (st.Length() == **this**->Length())

{

**for** (**int** i = 0; i < st.Length(); i++)

{

**if** (str[i] != st[i])

**return** **false**;

}

**return** **true**;

}

**return** **false**;

}

**bool** **operator**!=(String& st)

{

**if** (st.Length() != **this**->Length())

**return** **true**;

**if** (st.Length() <= **this**->Length())

{

**for** (**int** i = 0; i < st.Length(); i++)

{

**if** (str[i] == st[i])

**return** **false**;

**else** **return** **true**;

}

}

**if** (st.Length() >= **this**->Length())

{

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

{

**if** (str[i] == st[i])

**return** **false**;

**else** **return** **true**;

}

}

}

**bool** **operator**>(String& st)

{

**if** (\***this** == st)

**return** **false**;

**if** (st.Length() == **this**->Length())

{

**for** (**int** i = 0; i < st.Length(); i++)

{

**if** (str[i] < st[i])

**return** **false**;

}

**return** **true**;

}

**if** (st.Length() < **this**->Length())

**return** **true**;

**if** (st.Length() > **this**->Length())

**return** **false**;

}

**bool** **operator**<(String& st)

{

**if** (\***this** == st)

**return** **false**;

**if** (st.Length() == **this**->Length())

{

**for** (**int** i = 0; i < st.Length(); i++)

{

**if** (str[i] > st[i])

**return** **false**;

}

**return** **true**;

}

**if** (st.Length() < **this**->Length())

**return** **false**;

**if** (st.Length() > **this**->Length())

**return** **true**;

}

**int** SearchFirst(T elem)

{

**for** (**int** i = 0; i < str.size(); i++)

**if** (str[i] == elem)

**return** i;

**return** -1;

}

**int** SearchLast(T elem)

{

**for** (**int** i = str.size() - 1; i >= 0; i--)

**if** (str[i] == elem)

**return** i;

**return** -1;

}

};

**int** main()

{

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

String<**char**> s1(5);

cout << "(1)\n" << s1;

String<**char**> s2(5, 'X');

cout << "(2)\n" << s2;

String<**char**> s3 = s1;

cout << "(3)\n" << s3;

**int** arr[5] = { 1, 2, 3, 4, 5 };

String<**int**> s4(arr, 5);

cout << "(4)\n" << s4;

String<**int**> s6 = s3;

cout << "(5)\n" << s6;

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

string s = s3.GetStr();

cout << "(1)\n" << s << endl;

vector<**char**> v = s3.GetVect();

cout << "(2)\n";

**for** (**int** i = 0; i < v.size(); i++)

cout << v[i];

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

cout << "(1)\ns3 length = " << s3.Length() << endl;

cout << "(2)\ns3[3] = " << s3[3] << endl;

cout << "(3)" << endl;

cout << s3;

s3[3] = 'P';

cout << "s3[3] = 'P'. s3: \n" << s3 ;

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

String<**char**> s7 = s3.SubString(1, 3);

cout << "(1)\n";

cout << s7;

s7 + 'A';

s7 + 'B';

s7 + 'C';

cout << "(2)\n";

cout << s7;

s7 \* 3;

cout << "(3)\n";

cout << s7;

s7 + s3;

cout << "(4)\n";

cout << s7;

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

String<**char**> s8;

cout << "(1)\n";

cin >> s8;

cout << s8;

cout << "(2)\n";

**char** ar1[4] = {'x', 'b', 'c', 'd'};

**char** ar2[4] = { 'a', 'b', 'c', 'd' };

String<**char**> s9(ar1, 4);

cout << s9;

String<**char**> s10(ar2, 4);

cout << s10;

**if** (s9 > s10)

cout << "s9 > s10" << endl;

**if** (s9 < s10)

cout << "s9 < s10" << endl;

**if** (s9 == s10)

cout << "s9 == s10" << endl;

**if** (s9 != s10)

cout << "s9 != s10" << endl;

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

cout << s7;

cout << "(1)\n";

**int** n = s7.SearchFirst('P');

cout << n << endl;

cout << "(2)\n";

n = s7.SearchLast('P');

cout << n;

}

Результати тестування:

