МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ПЕРМСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

ЭЛЕКТРОТЕХНИЧЕСКИЙ ФАКУЛЬТЕТ

КАФЕДРА «ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И АВТОМАТИЗИРОВАННЫХ СИСТЕМ»

ОТЧЁТ ПО ГЕНЕРАТОРУ ПЕРЕСТАНОВОК БЕЗ ПОВТОРЕНИЙ

Дисциплина: «Основы алгоритмизации и программирования»

Выполнил:
Студент группы ИВТ-21-2б
Безух Владимир Сергеевич

Проверил:
Доцент кафедры ИТАС
Полякова Ольга Андреевна

Пермь,
Октябрь 2021

Постановка задачи

Реализовать перебор всех перестановок без повторений заданной пользователем последовательности символов. Найти лексикографически-разрядно максимальную строку символов среди всех перестановок. Если все символы цифры, то такая строка — максимальное число.

Анализ задачи

Наиболее часто задача решается в лексикографическом порядке, поэтому перед началом работы генератора перестановок без повторений следует отсортировать последовательность символов по возрастанию (“1234”, “abcd”).

Сложность алгоритма можно ориентировочно оценить через формулы комбинаторики. Количество перестановок равно $\frac{N!}{r_1! \cdot r_2! \cdot \ldots \cdot r_k!}$(1), где N — количество символов последовательности, а $\{r_1, r_2, \ldots, r_k\}$— множество количества повторений каждого символа в последовательности. Не трудно заметить, что, если все символы последовательности различны, формула (1) вырождается до формулы $N!$.

Пошагово рассмотрим алгоритм поиска следующей перестановки для текущей последовательности символов (далее — строки):

1. Рассмотрим строку справа налево, попарно проверяя рядом стоящие символы так, чтобы каждый элемент с большим номером (правый символ пары) лексикографически был не больше чем элемент с меньшим номером (левый символ пары). Если данное условие будет нарушено, необходимо остановиться и отметить индекс левого символа пары (2). Таким образом, мы ищем справа налево первую рядом стоящую пару символов, отсортированную в лексикографическом порядке по возрастанию.
2. Снова просмотреть пройденный путь справа налево пока не дойдем до первого символа, который лексикографически больше чем отмеченный на предыдущем шаге.
3. Поменять местами два полученных элемента.
4. Теперь в части строки, которая размещена справа от позиции (2), надо отсортировать все символы в порядке возрастания. Поскольку до этого они все уже были записаны в порядке убывания, достаточно эту подпоследовательность просто перевернуть.

Полученная последовательность будет рассматриваться в качестве исходной на следующей итерации алгоритма. Как только строка символов будет отсортирована по убыванию, мы получим последнюю перестановку и алгоритм завершится.

Не сложно заметить, что лексикографически-разрядно максимальная строка символов среди всех перестановок — это самая последняя перестановка. Если все символы цифры, то такая строка — максимальное число.

Демонстрация исполнения кода

В рамках демонстрации работы кода его участки расположены в удобной для восприятия последовательности. Расположение этих участков в исходном коде может отличаться в соответствии со всеми особенностями языка программирования C++.

```
#include <random>
#include <string>
#include <iostream>
#include <algorithm>

using namespace std;

int main()
{
	setlocale(LC_ALL, "Russian");

	firstTask();
}
```

Выше приведённый участок кода содержит директивы подключения необходимых в работе программы стандартных библиотек, объявление рабочего пространства имён, установку русской локализации обработки символов, основную функцию генератора перестановок без повторений.

```
void firstTask()
{
	size_t permutation_counter = 1; // счётчик количества перестановок без повторений

	string input_string = getString();

	sort(input_string.begin(), input_string.end());
	// сортируем символы в строке по возрастанию

	...
```

После получения строки от пользователя через функцию getString(), происходит сортировка строки по возрастанию через функцию sort().

```
string getString()
{
	cout << "Введите значение: ";

	string INPUT_STRING; getline(cin, INPUT_STRING);

	return INPUT_STRING;
}
```

Затем, пока для текущей строки существует следующая перестановка без повторений, продолжается цикл, последовательно выводящий результаты нахождения новых перестановок без повторений (т.е. непосредственно сами перестановки).

```
...

cout << "Перестановка №" << permutation_counter << " – " << input_string << "\n";
	// продолжаем цикл, пока существует следующая перестановка без повторений
	while (nextPermutation(input_string))
		cout << "Перестановка №" << ++permutation_counter
		     << " – " << input_string << "\n";
```

```
        cout << "Всего перестановок без повторений: " << permutation_counter << "\n"
             << "Лексикографически-разрядно максимальная"
             << " строка символов среди всех перестановок"
             << " (если все символы цифры, то это максимальное число) : "
             << input_string << "\n\n";
}
```

Работа функции nextPermutation() подробно закомментирована. Алгоритм работы описан выше в разделе «Общий анализ задачи».

```
bool nextPermutation(string &string)
{
        // на вход подаётся заранее отсортированная по возрастанию строка символов

        int i = string.length() - 2; // индекс предпоследнего элемента

        // пока есть символы и не найдена пара отсортированных по возрастанию элементов...
        while (i >= 0 && string[i] >= string[i + 1]) --i;
        // ...проверять рядом стоящие пары элементов справа налево

        if (i == -1) // если все пары элементов отсортированы по убыванию...
                return false; // ...значит больше нет новых перестановок без повторений

        int j = string.length() - 1; // индекс последнего элемента

        while (string[i] >= string[j]) --j;
        // ищем первый элемент j с конца, который больше элемента i

        swap(string[i], string[j]); // сортируем найденные элементы по убыванию

        int l = i + 1, r = string.length() - 1;
        // сортируем по возрастанию часть последовательности справа от i

        while (l < r) // поскольку все элементы справа от i отсортированы по убыванию...
                swap(string[l++], string[r--]);
        // ...достаточно перевернуть эту подпоследовательность

        return true; // новая перестановка успешно найдена
}
```

Как только перестановки без повторений заканчиваются, программа завершается.

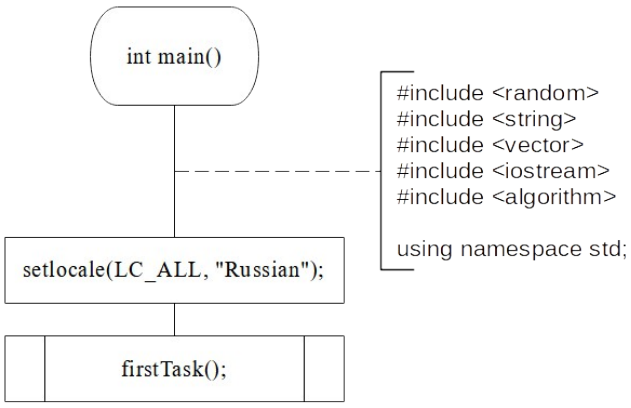
Описание переменных

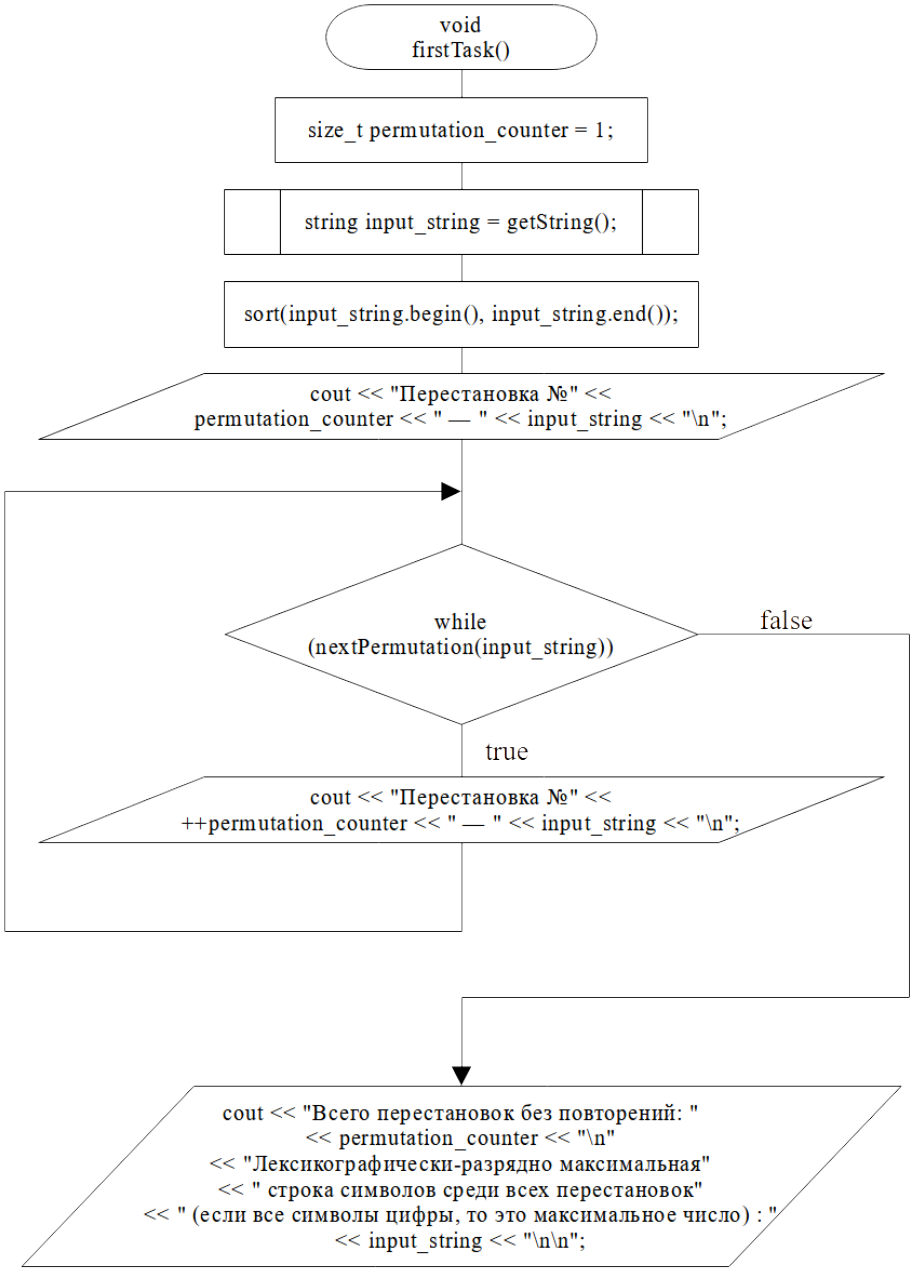
size_t permutation_counter = 1; — счётчик количества перестановок без повторений.

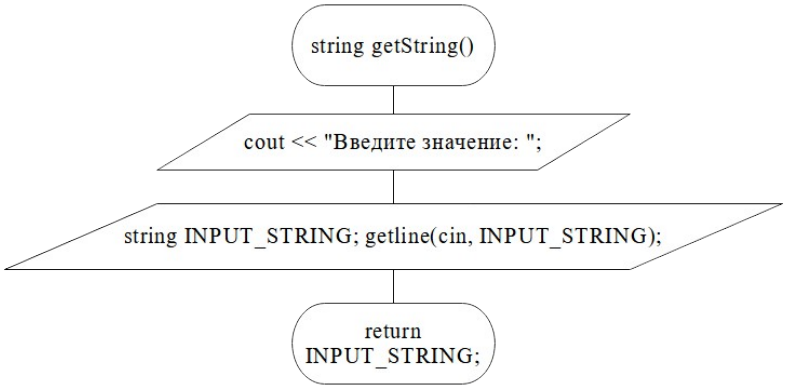
string input_string = getString(); — введённая пользователем строка символов.

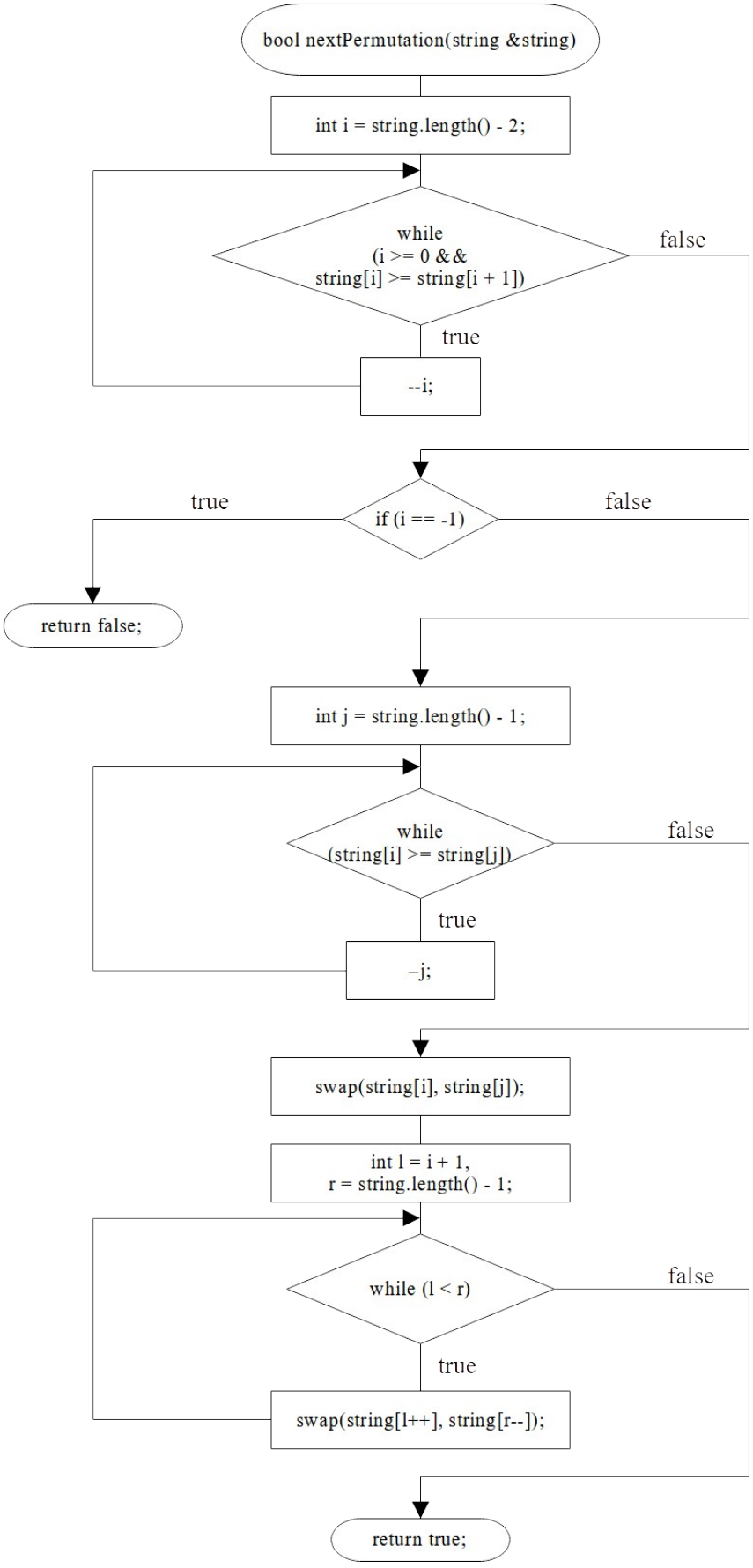
int i, j, l, r; — различные переменные индексов, необходимые для работы алгоритма поиска следующей перестановки.

Блок-схема


int main()
#include <random>
#include <string>
#include <vector>
#include <iostream>
#include <algorithm>
using namespace std;
setlocale(LC_ALL, "Russian");
firstTask();


void
firstTask()
size_t permutation_counter = 1;
string input_string = getString();
sort(input_string.begin(), input_string.end());
cout << "Перестановка №" <<
permutation_counter << " — " << input_string << "\n";
while
(nextPermutation(input_string))
false
true
cout << "Перестановка №" <<
++permutation_counter << " — " << input_string << "\n";
cout << "Всего перестановок без повторений: "
<< permutation_counter << "\n"
<< "Лексикографически-разрядно максимальная"
<< " строка символов среди всех перестановок"
<< " (если все символы цифры, то это максимальное число) : "
<< input_string << "\n\n";


string getString()
cout << "Введите значение: ";
string INPUT_STRING; getline(cin, INPUT_STRING);
return
INPUT_STRING;

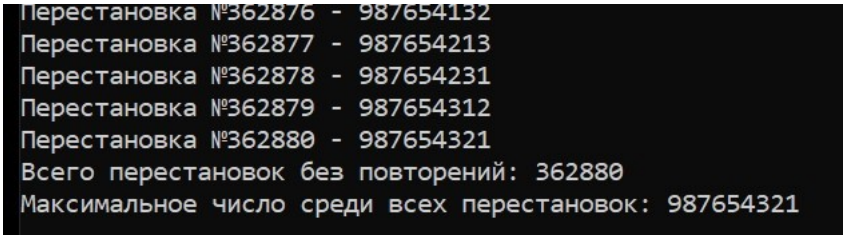

bool nextPermutation(string &string)
int i = string.length() - 2;
while
(i >= 0 &&
string[i] >= string[i + 1])
false
true
--i;
if (i == -1)
true
false
return false;
int j = string.length() - 1;
while
(string[i] >= string[j])
false
true
–j;
swap(string[i], string[j]);
int l = i + 1,
r = string.length() - 1;
while (l < r)
false
true
swap(string[l++], string[r--]);
return true;

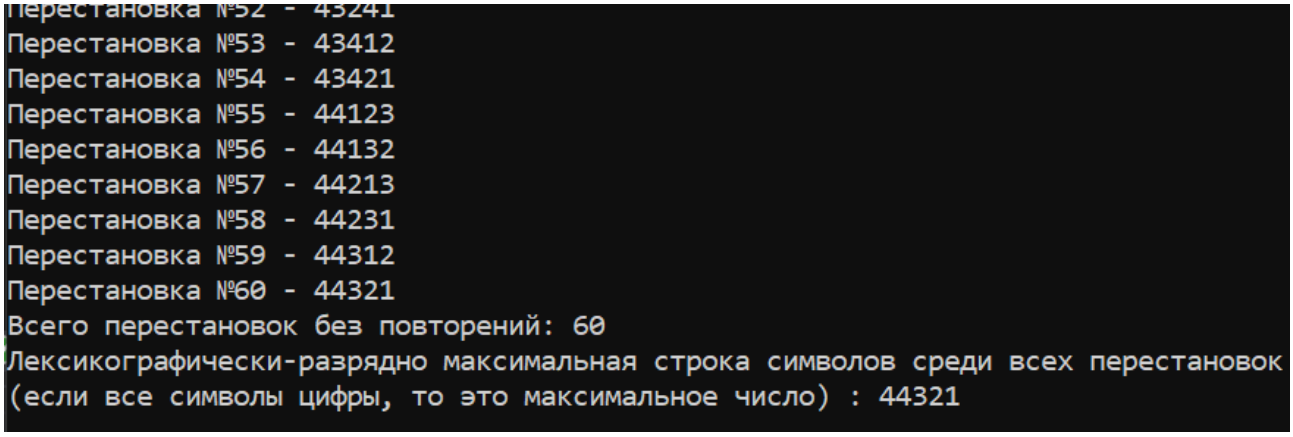
Исходный код

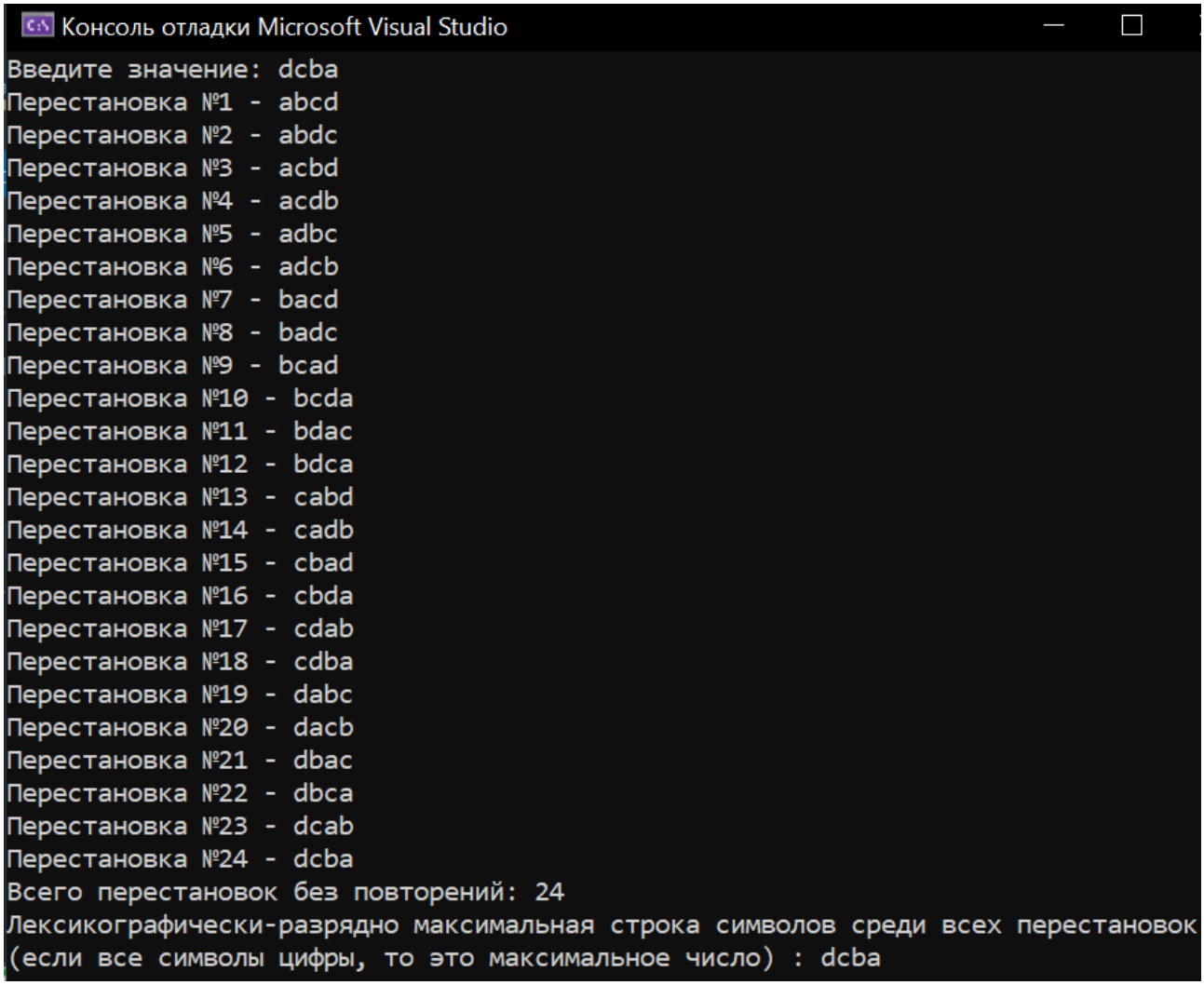
```
/*
* This code is licensed under the Creative Commons
* Attribution - NonCommercial - NoDerivatives 4.0 International License.
* To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-nd/4.0/
* or send a letter to Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.
*
* https://bezukh.wixsite.com/blog
* https://github.com/BezukhVladimir
*
* © Developed by Bezukh Vladimir. All right reserved.
*/

/*
* Developed by Bezukh Vladimir
* October 2021
* ИВТ-21-26
*
* Вывести все перестановки строки символов без повторений.
* Найти лексикографически-разрядно максимальную строку символов среди всех перестановок.
* Если все символы цифры, то такая строка – максимальное число.
*/

#include <random>
#include <string>
#include <iostream>
#include <algorithm>

using namespace std;

string getString()
{
	cout << "Введите значение: ";

	string INPUT_STRING; getline(cin, INPUT_STRING);

	return INPUT_STRING;
}
```

```cpp
bool nextPermutation(string &string)
{
	// на вход подаётся заранее отсортированная по возрастанию строка символов

	int i = string.length() - 2; // индекс предпоследнего элемента

	// пока есть символы и не найдена пара отсортированных по возрастанию элементов...
	while (i >= 0 && string[i] >= string[i + 1]) --i;
	// ...проверять рядом стоящие пары элементов справа налево

	if (i == -1) // если все пары элементов отсортированы по убыванию...
		return false; // ...значит больше нет новых перестановок без повторений

	int j = string.length() - 1; // индекс последнего элемента

	while (string[i] >= string[j]) --j;
	// ищем первый элемент j с конца, который больше элемента i

	swap(string[i], string[j]); // сортируем найденные элементы по убыванию

	int l = i + 1, r = string.length() - 1;
	// сортируем по возрастанию часть последовательности справа от i

	while (l < r) // поскольку все элементы справа от i отсортированы по убыванию...
		swap(string[l++], string[r--]);
	// ...достаточно перевернуть эту подпоследовательность

	return true; // новая перестановка успешно найдена
}


void firstTask()
{
	size_t permutation_counter = 1; // счётчик количества перестановок без повторений

	string input_string = getString();

	sort(input_string.begin(), input_string.end());
	// сортируем символы в строке по возрастанию

	cout << "Перестановка №" << permutation_counter << " – " << input_string << "\n";

	// продолжаем цикл, пока существует следующая перестановка без повторений
	while (nextPermutation(input_string))
		cout << "Перестановка №" << ++permutation_counter
		     << " – " << input_string << "\n";

	cout << "Всего перестановок без повторений: " << permutation_counter << "\n"
	     << "Лексикографически-разрядно максимальная"
	     << " строка символов среди всех перестановок"
	     << " (если все символы цифры, то это максимальное число) : "
	     << input_string << "\n\n";
}

int main()
{
	setlocale(LC_ALL, "Russian");

	firstTask();
}
```

Скриншоты консольного интерфейса программы







Анализ результатов

Заявленный генератор перестановок без повторений работает с любой длиной последовательности символов (с учётом стремительного роста сложности исполнения), с любыми символами (без учёта особенностей отображения локализаций), которые можно лексикографически сравнить между собой, и с любым сочетанием этих символов.

За счёт однозначной интерпретации вывода результатов алгоритма, лексикографически-разрядно максимальная строка символов среди всех перестановок — это самая последняя перестановка. Если все символы этой строки цифры, то такая последовательность цифр — максимальное число.