

**МИНОБРНАУКИ РОССИИ**  
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**  
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**  
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**  
**Кафедра МО ЭВМ**

**ОТЧЕТ**  
**по лабораторной работе №4**  
**по дисциплине «Алгоритмы и структуры данных»**  
**Тема: Сортировки**

Студент гр. 9382

\_\_\_\_\_

Бочаров Г.С.

Преподаватель

\_\_\_\_\_

Фирсов М.А.

Санкт-Петербург

2020

## **Цель работы.**

Реализовать пасьянсную сортировку. Понять принцип работы данной сортировки и ее преимущества и недостатки.

## **Задание.**

Вариант 22:

22. Пасьянская сортировка.

## **Описание алгоритма.**

1) На этапе считывания из входных данных формируются упорядоченные подпоследовательности наибольшей длины.

2) Далее выбирается минимальное значение, являющееся началом какой-либо упорядоченной подпоследовательности. Значение заносится в результирующий массив и удаляется из подпоследовательности, началом которой является.

3) Пункт 2 повторяется  $n$  раз, где  $n$  — количество введенных элементов.

Для хранения массива подпоследовательностей был выбран массив односвязных списков.

При считывании каждого значения, оно заносится в массив списков следующим образом:

1) Если нашелся список начало которого больше считанного элемента, то элемент становится началом этого списка.

2) Если такого списка не нашлось, то создается новый список, началом которого является считанный элемент.

## **Функции и структуры данных.**

`bool isNumber(const std::string &str)` — Функция принимает на вход строку.

Функция возвращает `true`, если вся строка является числом, в противном случае

функция возвращает false.

bool compS(const std::string &a, const std::string &b) — Функция сравнивает 2 строки в лексикографическом порядке.

bool compI(const std::string &a, const std::string &b) — Функция сравнивает 2 строки по их числовому значению

bool compare(const std::string &a, const std::string &b) — Функция сравнивает одним из способов, в зависимости от выбора пользователя.

void pushInPiles(Piles &piles, const std::string &element) — Функция принимает на вход массив упорядоченных подпоследовательностей и строковое значение считанного элемента. Функция добавляет элемент в массив упорядоченных подпоследовательностей по алгоритму, описанному выше.

template<typename StreamT>

void readPiles(StreamT &in, Piles &piles) — Функция принимает на вход поток ввода и массив подпоследовательностей. Функция считывает в переданный массив элементы.

template<typename T>

void printPile(const T &pile) — Функция выводит подпоследовательность на экран.

template<typename T>

void printPiles(const T &piles) — Функция выводит массив подпоследовательностей на экран.

std::string popMin(Piles &piles, int &i) — Функция принимает на вход массив подпоследовательностей. Функция находит элемент являющийся минимальным

из начал подпоследовательностей, удаляет его из списка и возвращает его строковое значение.

`std::vector<std::string> sort(Piles &piles)` — Функция принимает на вход массив подпоследовательностей и возвращает отсортированный массив строк.

`void launch()` - Функция запрашивает у пользователя формат ввода и вызывает функцию считывания входных данных. Далее вызывает функцию сортировки и вывода результата.

### **Оценка алгоритма.**

При раскладывании элементов по стопкам (упорядоченные подпоследовательности) для поиска самой левой подходящей стопки используется бинарный поиск. Соответственно, поиск самой левой стопки занимает  $O(\log p)$ , где  $p$  — количество стопок (стеков). Таким образом, временная сложность раскладывания по стопкам не превышает  $O(n \log n)$ .

При получении минимальной вершины (начала стопки) из имеющихся мы пользуемся тем, что изначально эти вершины упорядочены и рассматривать для добавления вершину списка с номером  $n$  нам нужно только в случае, если мы уже добавили в результирующий массив вершину списка с номером  $n-1$ . В данной реализации временная сложность поиска минимальных вершин не превышает  $O(n^2)$ .

Таким образом в худшем случае алгоритм работает за  $O(n^2)$ ,

В лучшем за  $O(n)$ . Например, когда данные изначально отсортированы. Основное преимущество алгоритма в том, что на каждом этапе сортировки мы работаем с ограниченным набором данных, а конкретнее — только с вершинами списков.

### **Тестирование.**

Результаты тестирования представлены в табл. 1.

(Скрины см. в приложении Б)

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	1 23 43 2 323 0 43 3 3434 434 996 1	<p>Вывод стопок</p> <p>Стопка1 : 0 2 23</p> <p>Стопка2 : 1 3 43 43</p> <p>Стопка3 : 323</p> <p>Стопка4 : 434 3434</p> <p>Стопка5 : 996</p> <p>&lt;min&gt; = 0</p> <p>Вывод стопок</p> <p>Стопка1 : 2 23</p> <p>Стопка2 : 1 3 43 43</p> <p>Стопка3 : 323</p> <p>Стопка4 : 434 3434</p> <p>Стопка5 : 996</p> <p>&lt;min&gt; = 1</p> <p>Вывод стопок</p> <p>Стопка1 : 2 23</p> <p>Стопка2 : 3 43 43</p> <p>Стопка3 : 323</p> <p>Стопка4 : 434 3434</p> <p>Стопка5 : 996</p> <p>&lt;min&gt; = 2</p> <p>Вывод стопок</p> <p>Стопка1 : 23</p> <p>Стопка2 : 3 43 43</p> <p>Стопка3 : 323</p>	<p>Вывод массива</p> <p>стопок на каждом из этапов сортировки.</p> <p>Вывод результата сортировки числовой последовательности.</p>

		<p>Стопка4 : 434 3434</p> <p>Стопка5 : 996</p> <p>&lt;min&gt; = 3</p> <p>Вывод стопок</p> <p>Стопка1 : 23</p> <p>Стопка2 : 43 43</p> <p>Стопка3 : 323</p> <p>Стопка4 : 434 3434</p> <p>Стопка5 : 996</p> <p>&lt;min&gt; = 23</p> <p>Вывод стопок</p> <p>Стопка1 : 43 43</p> <p>Стопка2 : 323</p> <p>Стопка3 : 434 3434</p> <p>Стопка4 : 996</p> <p>&lt;min&gt; = 43</p> <p>Вывод стопок</p> <p>Стопка1 : 43</p> <p>Стопка2 : 323</p> <p>Стопка3 : 434 3434</p> <p>Стопка4 : 996</p> <p>&lt;min&gt; = 43</p> <p>Вывод стопок</p> <p>Стопка1 : 323</p> <p>Стопка2 : 434 3434</p> <p>Стопка3 : 996</p>	
--	--	---	--

		<p>&lt;min&gt; = 323</p> <p>Вывод стопок</p> <p>Стопка1 : 434 3434</p> <p>Стопка2 : 996</p> <p>&lt;min&gt; = 434</p> <p>Вывод стопок</p> <p>Стопка1 : 3434</p> <p>Стопка2 : 996</p> <p>&lt;min&gt; = 996</p> <p>Вывод стопок</p> <p>Стопка1 : 3434</p> <p>&lt;min&gt; = 3434</p> <p>Вывод стопок</p> <p>result--&gt;&gt; 0 1 2 3 23 43 43 323</p> <p>434 996 3434</p>	
2.	<p>0</p> <p>1002 2321 211 10 22 121</p> <p>203 56 3A4 21 3</p>	<p>Вывод стопок</p> <p>Стопка1 : 10 1002</p> <p>Стопка2 : 121 211 2321</p> <p>Стопка3 : 203 22</p> <p>Стопка4 : 21 3A4 56</p> <p>Стопка5 : 3</p> <p>&lt;min&gt; = 10</p> <p>Вывод стопок</p> <p>Стопка1 : 1002</p> <p>Стопка2 : 121 211 2321</p> <p>Стопка3 : 203 22</p> <p>Стопка4 : 21 3A4 56</p>	<p>Сортировка</p> <p>уже</p> <p>строковых</p> <p>значений в</p> <p>лексикографи</p> <p>ческом</p> <p>порядке.</p>

	<p>Стопка5 : 3</p> <p>&lt;min&gt; = 1002</p> <p>Вывод стопок</p> <p>Стопка1 : 121 211 2321</p> <p>Стопка2 : 203 22</p> <p>Стопка3 : 21 3A4 56</p> <p>Стопка4 : 3</p> <p>&lt;min&gt; = 121</p> <p>Вывод стопок</p> <p>Стопка1 : 211 2321</p> <p>Стопка2 : 203 22</p> <p>Стопка3 : 21 3A4 56</p> <p>Стопка4 : 3</p> <p>&lt;min&gt; = 203</p> <p>Вывод стопок</p> <p>Стопка1 : 211 2321</p> <p>Стопка2 : 22</p> <p>Стопка3 : 21 3A4 56</p> <p>Стопка4 : 3</p> <p>&lt;min&gt; = 21</p> <p>Вывод стопок</p> <p>Стопка1 : 211 2321</p> <p>Стопка2 : 22</p> <p>Стопка3 : 3A4 56</p> <p>Стопка4 : 3</p> <p>&lt;min&gt; = 211</p>	
--	--	--



		<p>Вывод стопок</p> <p>Стопка1 : 2321</p> <p>Стопка2 : 22</p> <p>Стопка3 : 3A4 56</p> <p>Стопка4 : 3</p> <p>&lt;min&gt; = 22</p> <p>Вывод стопок</p> <p>Стопка1 : 2321</p> <p>Стопка2 : 3A4 56</p> <p>Стопка3 : 3</p> <p>&lt;min&gt; = 2321</p> <p>Вывод стопок</p> <p>Стопка1 : 3A4 56</p> <p>Стопка2 : 3</p> <p>&lt;min&gt; = 3</p> <p>Вывод стопок</p> <p>Стопка1 : 3A4 56</p> <p>&lt;min&gt; = 3A4</p> <p>Вывод стопок</p> <p>Стопка1 : 56</p> <p>&lt;min&gt; = 56</p> <p>Вывод стопок</p> <p>result--&gt;&gt; 10 1002 121 203 21 211 22 2321 3 3A4 56</p>	
3.	0	result-->> I Somebody ain't	

	Somebody once told me the world is gonna roll me I ain't the sharpest tool in the shed	gonna in is me me once roll sharpest shed the the the told tool world	
4.	1 4 3 2 1 6 7 8 5 9 0 11	result-->> 0 1 2 3 4 5 6 7 8 9 11	
5.	1 100 12 5 4 3 2 1 9 8 7 6 34 55 23 77 87 65 43 27	result-->> 1 2 3 4 5 6 7 8 9 12 23 27 34 43 55 65 77 87 100	
6.	1 23 3DD3 F3S	Найдено нецелочисленное значение <3DD3>	Пользователь выбрал сортировку чисел, но не все данные имеют целочисленный тип.
7.	0 0	Вывод стопок Стопка1 : 0 <min> = 0 Вывод стопок  result-->> 0	
8.	1 2 3	Стопка1 : 2 Стопка2 : 3 <min> = 2 Вывод стопок Стопка1 : 3	

		$\langle \min \rangle = 3$ Вывод стопок result-->> 2 3	
9.	0 Введите имя файла : kek	Файл не найден!	Указано неверное название файла.

### **Выводы.**

В ходе работы был разработан алгоритм пасьянсовой сортировки для числовых и строковых значений. Оценено время работы реализованного алгоритма.

## ПРИЛОЖЕНИЕ А

### ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
#include <iostream>
```

```
#include <vector>
```

```
#include <forward_list>
```

```
#include <fstream>
```

```
#include <sstream>
```

```
bool sortFormat = false;
```

```
typedef std::vector<std::forward_list<std::string>> Piles;
```

```
typedef std::forward_list<std::string> Pile;
```

```
bool isNumber(const std::string &str) {
```

```
    char *ptr;
```

```
    strtol(str.c_str(), &ptr, 10);
```

```
    return *ptr == '\0';
```

```
}
```

```
bool compS(const std::string &a, const std::string &b) {
```

```
    return a >= b;
```

```
}
```

```
bool compI(const std::string &a, const std::string &b) {
```

```
    if (!isNumber(a))
```

```
        throw std::runtime_error("Найдено нецелочисленное значение < " + a + ">");
```

```
    if (!isNumber(b))
```

```
        throw std::runtime_error("Найдено нецелочисленное значение < " + b + ">");
```

```
    return strtol(a.c_str(), nullptr, 10) >= strtol(b.c_str(), nullptr, 10);
```

```
}
```

```
bool compare(const std::string &a, const std::string &b) {  
    return sortFormat == 1 ? compI(a, b) : compS(a, b);  
}
```

//Функция добавляет считанный элемент в массив списков

```
void pushInPiles(Piles &piles, const std::string &element) {  
    if (piles.empty()) {  
        Pile k;  
        k.push_front(element);  
        piles.push_back(k);  
        return;  
    }  
    int beg = 0, end = piles.size() - 1;  
    int temp1 = 0;  
    int temp2 = 0;  
    int mid;  
    while (beg <= end) {  
        mid = (end + beg) / 2;  
        if (compare(piles.at(mid).front(), element)) {  
            temp2 = mid;  
            end = mid - 1;  
        } else {  
            temp1 = mid;  
            beg = mid + 1;  
        }  
    }  
    if (compare(piles.at(temp1).front(), element)) {  
        piles.at(temp1).push_front(element);  
        return;  
    }  
}
```

```

    } else if (compare(piles.at(temp2).front(), element)) {
        piles.at(temp2).push_front(element);
        return;
    } else {
        Pile k;
        k.push_front(element);
        piles.push_back(k);
    }
}

```

//Функция считывает элементы из входного потока

```

template<typename StreamT>

```

```

void readPiles(StreamT &in, Piles &piles) {

```

```

    std::string element;

```

```

    std::string line;

```

```

    std::getline(in, line);

```

```

    if (line == "0")

```

```

        sortFormat = 0;

```

```

    else if (line == "1")

```

```

        sortFormat = 1;

```

```

    else throw std::runtime_error("Неверный тип сортировки");

```

```

    std::getline(in, line);

```

```

    std::istringstream str(line);

```

```

    while (getline(str, element, ' ')) {

```

```

        if (element != " ")

```

```

            pushInPiles(piles, element);

```

```

    }

```

```

}

```

//Функция выводит упорядоченный список на экран

```
template<typename T>
void printPile(const T &pile) {
    for (auto &i:pile) {
        std::cout << i << " ";
    }
}
```

//Функция выводит массив списков

```
template<typename T>
void printPiles(const T &piles) {
    int k = 1;
    std::cout << "Вывод стопок" << std::endl;
    for (auto &i:piles) {
        std::cout << "Стопка" << k << " : ";
        printPile(i);
        k++;
        std::cout << std::endl;
    }
}
```

//Функция извлекает минимальную вершину

```
std::string popMin(Piles &piles, int &k) {

    std::string res = piles.front().front();
    int num = 0;
    for (int i = 0; i <= k && i<piles.size(); i++) {
        if (compare(res, piles.at(i).front())) {
            res = piles.at(i).front();
            num = i;
        }
    }
}
```

```

    }
}
piles.at(num).pop_front();
k = num + 1;
if (piles.at(num).empty()) {
    piles.erase(piles.begin() + num);
}
std::cout << "<min> = " << res << std::endl;
return res;
}
//Функция сортирует считанные данные
std::vector<std::string> sort(Piles &piles) {
    std::vector<std::string> res;
    int k = 0;
    while (!piles.empty()) {
        res.push_back(popMin(piles, k));
        printPiles(piles);

        std::cout<<std::endl;
    }
    return res;
}
//Функция выводит массив
void printReasult(const std::vector<std::string>& result) {
    std::cout<<"result-->> ";
    for (auto &i : result)
        std::cout << i << " ";
}
//Функция считывает формат ввода и выполняет считывание входных данных
void launch() {
    Piles piles;

```



```

int readFormat;

std::cout << "0 - считать из файла, 1 - считать с консоли" << std::endl;
std::cin >> readFormat;
std::cin.ignore();
switch (readFormat) {
    case 0: {
        std::cout << "Введите имя файла : ";
        std::ifstream in;
        std::string fileName;
        std::cin >> fileName;
        in.open(fileName);
        if (in) {
            readPiles(in, piles);
        } else
            throw std::runtime_error("Файл не найден!");
        in.close();
        break;
    }
    case 1: {
        std::cout
            << "Введите тип сортировки 1 - сортировка целых чисел, 0 -
сортировка строк в лексикографическом порядке"
            << std::endl;
        std::cout << "Затем введите последовательность значений через пробел.
Например: 1 2 3 4 5"
            << std::endl;
        readPiles(std::cin, piles);
        break;
    }
    default: {

```

```

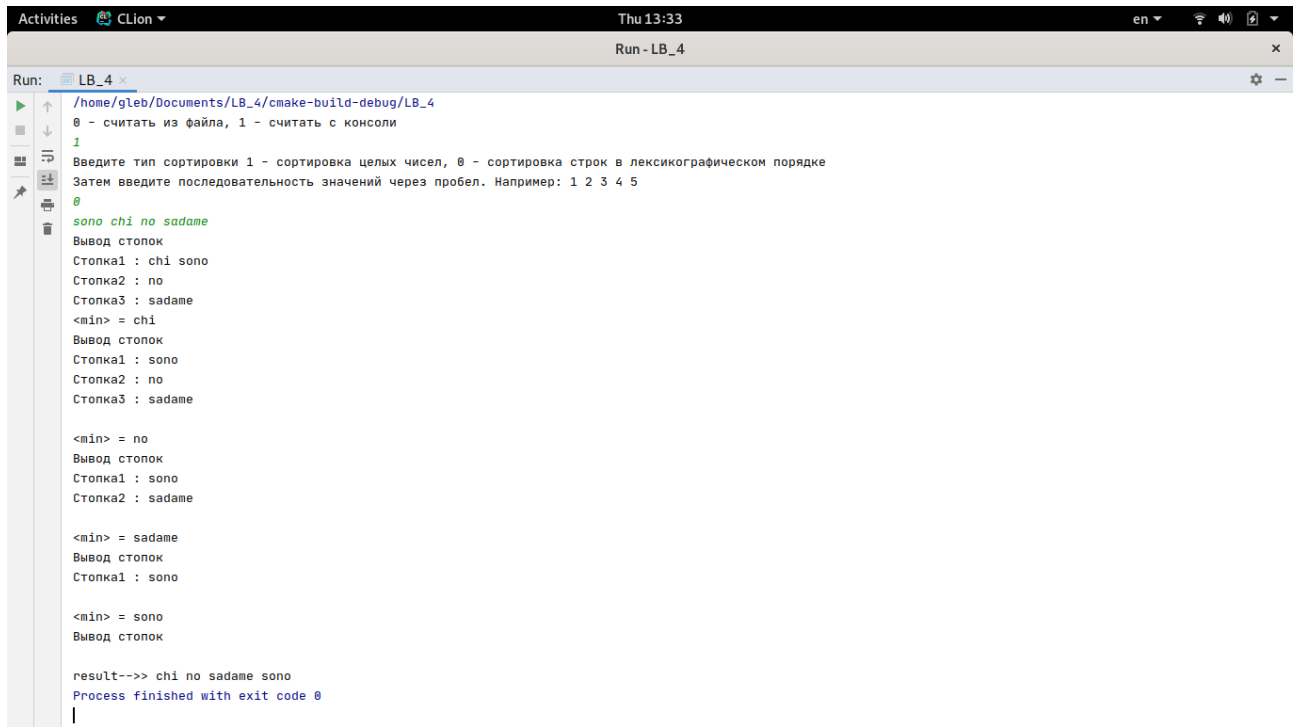
        throw std::runtime_error("Неверное действие");
    }
}

printPiles(piles);
printReasult(sort(piles));
}

int main() {
    try {
        launch();
    } catch (std::exception &e) {
        std::cerr << e.what() << std::endl;
    }
    return 0;
}

```

## ПРИЛОЖЕНИЕ Б



The screenshot shows the CLion Run window for the project 'LB\_4'. The window title is 'Run - LB\_4'. The output text is as follows:

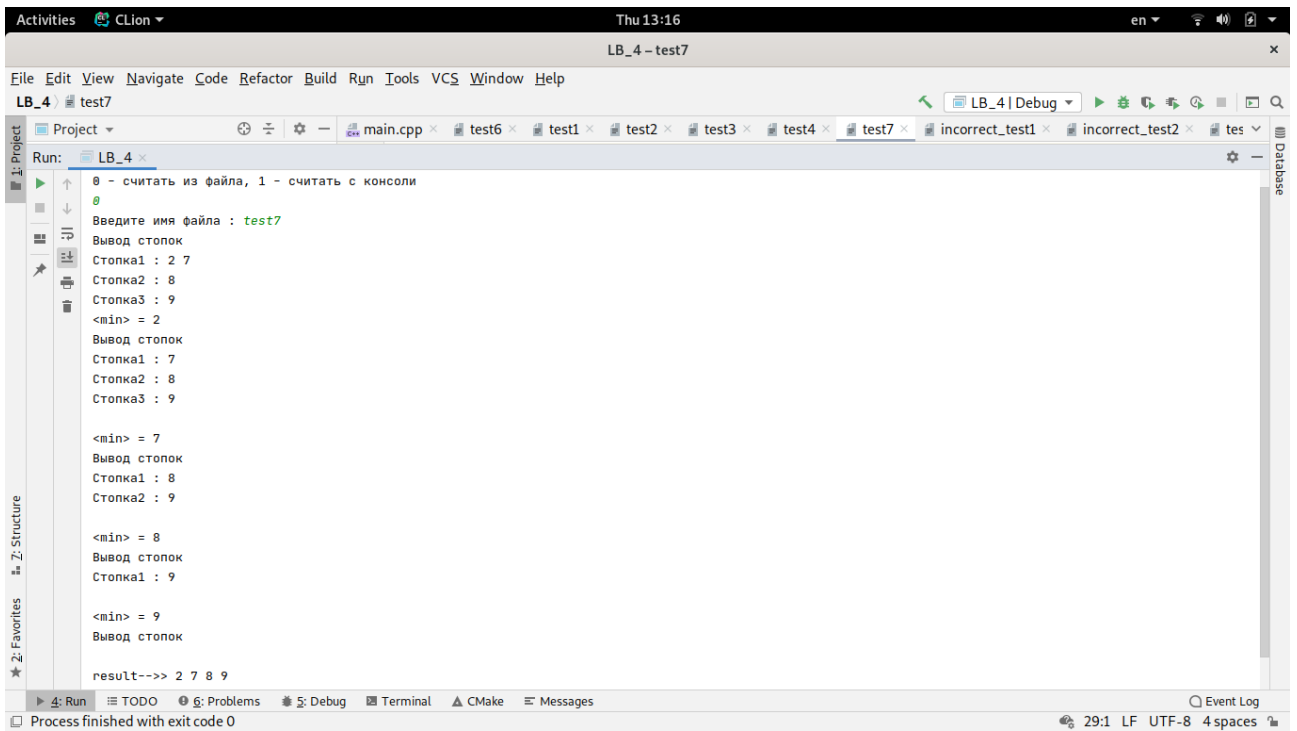
```
Run: LB_4
/home/gleb/Documents/LB_4/cmake-build-debug/LB_4
0 - считать из файла, 1 - считать с консоли
1
Введите тип сортировки 1 - сортировка целых чисел, 0 - сортировка строк в лексикографическом порядке
Затем введите последовательность значений через пробел. Например: 1 2 3 4 5
0
sono chi no sadame
Вывод стопок
Столпка1 : chi sono
Столпка2 : no
Столпка3 : sadame
<min> = chi
Вывод стопок
Столпка1 : sono
Столпка2 : no
Столпка3 : sadame

<min> = no
Вывод стопок
Столпка1 : sono
Столпка2 : sadame

<min> = sadame
Вывод стопок
Столпка1 : sono

<min> = sono
Вывод стопок

result--> chi no sadame sono
Process finished with exit code 0
```



The screenshot shows the CLion IDE interface. The top toolbar includes buttons for Run, Debug, and other development actions. The 'Run' window is open, showing the output for the 'LB\_4' project. The output text is as follows:

```
Run: LB_4
0 - считать из файла, 1 - считать с консоли
0
Введите имя файла : test7
Вывод стопок
Столпка1 : 2 7
Столпка2 : 8
Столпка3 : 9
<min> = 2
Вывод стопок
Столпка1 : 7
Столпка2 : 8
Столпка3 : 9

<min> = 7
Вывод стопок
Столпка1 : 8
Столпка2 : 9

<min> = 8
Вывод стопок
Столпка1 : 9

<min> = 9
Вывод стопок

result--> 2 7 8 9
Process finished with exit code 0
```

The bottom status bar shows the current file is 'test7', the encoding is 'UTF-8', and there are 4 spaces. The bottom right corner shows the time '29:1' and the text 'Event Log'.