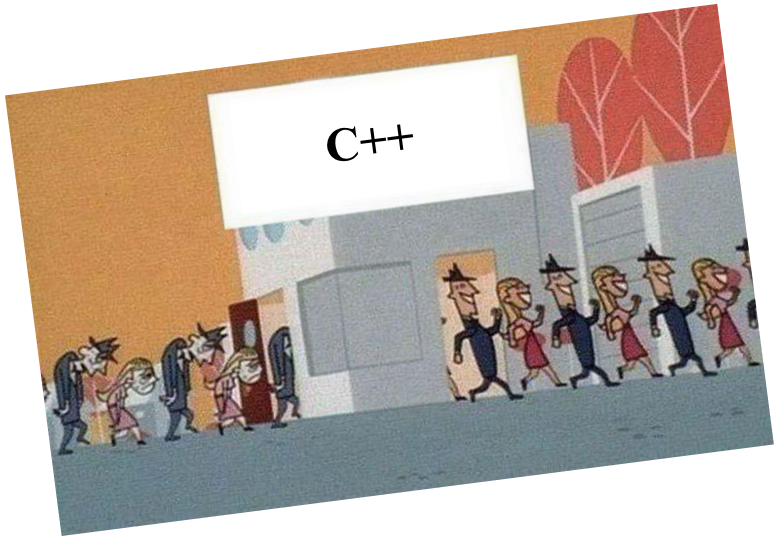
Структуры данных: queue, stack, deque

10 класс

Очередь queue (FIFO)

```
(Глобальная область)
#include <iostream>
#include <queue>
using namespace std;

int main()
{
    queue<int> queue;  // пустая очередь
}
```

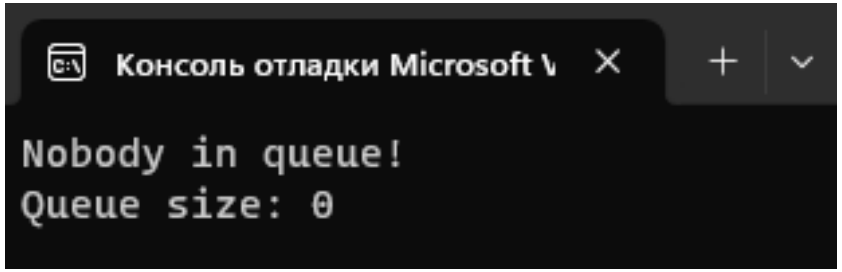


First In – First Out

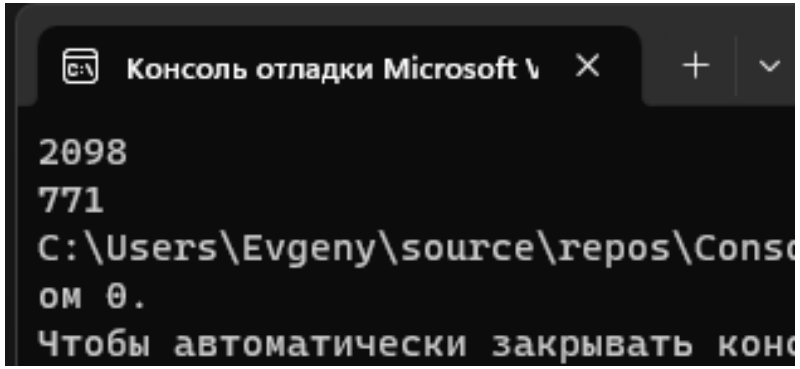
Первый вошел – первый вышел

Методы очередей

```cpp
int main()
{
    queue<int> queue;  // пустая очередь
    if (queue.empty())
    {
        cout << "Nobody in queue!" << endl;
        cout << "Queue size: " << queue.size() << endl;
    }
}
```




```cpp
int main()
{
    queue<int> queue;  // пустая очередь
    queue.push(2098);
    queue.push(771);
    cout << queue.front() << endl;
    cout << queue.back();
    queue.pop();
}
```



Сложность операций

Common Data Structure Operations

Data Structure	Time Complexity								Space Complexity
	Average				Worst				Worst
	Access	Search	Insertion	Deletion	Access	Search	Insertion	Deletion	
Array	O(1)	O(n)	O(n)	O(n)	O(1)	O(n)	O(n)	O(n)	O(n)
Stack	O(n)	O(n)	O(1)	O(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Queue	O(n)	O(n)	O(1)	O(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Singly-Linked List	O(n)	O(n)	O(1)	O(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Doubly-Linked List	O(n)	O(n)	O(1)	O(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Skip List	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)	O(n)	O(n)	O(n)	O(n log(n))
Hash Table	N/A	O(1)	O(1)	O(1)	N/A	O(n)	O(n)	O(n)	O(n)
Binary Search Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)
Cartesian Tree	N/A	O(log(n))	O(log(n))	O(log(n))	N/A	O(n)	O(n)	O(n)	O(n)
B-Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Red-Black Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Splay Tree	N/A	O(log(n))	O(log(n))	O(log(n))	N/A	O(log(n))	O(log(n))	O(log(n))	O(n)
AVL Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
KD Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)

Стек stack (LIFO)

```cpp
#include <iostream>
#include <stack>
using namespace std;

int main()
{
    stack<string> stack;
    // добавляем три элемента
    stack.push("Tom");
    stack.push("Bob");
    stack.push("Sam");

    cout << "stack size: " << stack.size() << endl; // stack size: 3

    while (!stack.empty())
    {
        cout << stack.top() << endl;
        stack.pop();
    }
}
```

Last In – First Out

Последний вошел – первый вышел

Сложность операций

Common Data Structure Operations

Data Structure	Time Complexity								Space Complexity
	Average				Worst				Worst
	Access	Search	Insertion	Deletion	Access	Search	Insertion	Deletion	
Array	O(1)	O(n)	O(n)	O(n)	O(1)	O(n)	O(n)	O(n)	O(n)
Stack	O(n)	O(n)	O(1)	O(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Queue	O(n)	O(n)	O(1)	O(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Singly-Linked List	O(n)	O(n)	O(1)	O(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Doubly-Linked List	O(n)	O(n)	O(1)	O(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Skip List	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)	O(n)	O(n)	O(n)	O(n log(n))
Hash Table	N/A	O(1)	O(1)	O(1)	N/A	O(n)	O(n)	O(n)	O(n)
Binary Search Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)
Cartesian Tree	N/A	O(log(n))	O(log(n))	O(log(n))	N/A	O(n)	O(n)	O(n)	O(n)
B-Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Red-Black Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Splay Tree	N/A	O(log(n))	O(log(n))	O(log(n))	N/A	O(log(n))	O(log(n))	O(log(n))	O(n)
AVL Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
KD Tree	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)

Двусторонняя очередь дек deque

```cpp
(Глобальная область)
#include <iostream>
#include <deque>
using namespace std;

int main()
{
    deque<int> numbers{ 1, 2, 3, 4, 5 };

    int first = numbers.front();     // 1
    int last = numbers.back();       // 5
    int second = numbers[1];         // 2
    int third = numbers.at(2);       // 3
    cout << first << second << third << last << endl; // 1235
}
```

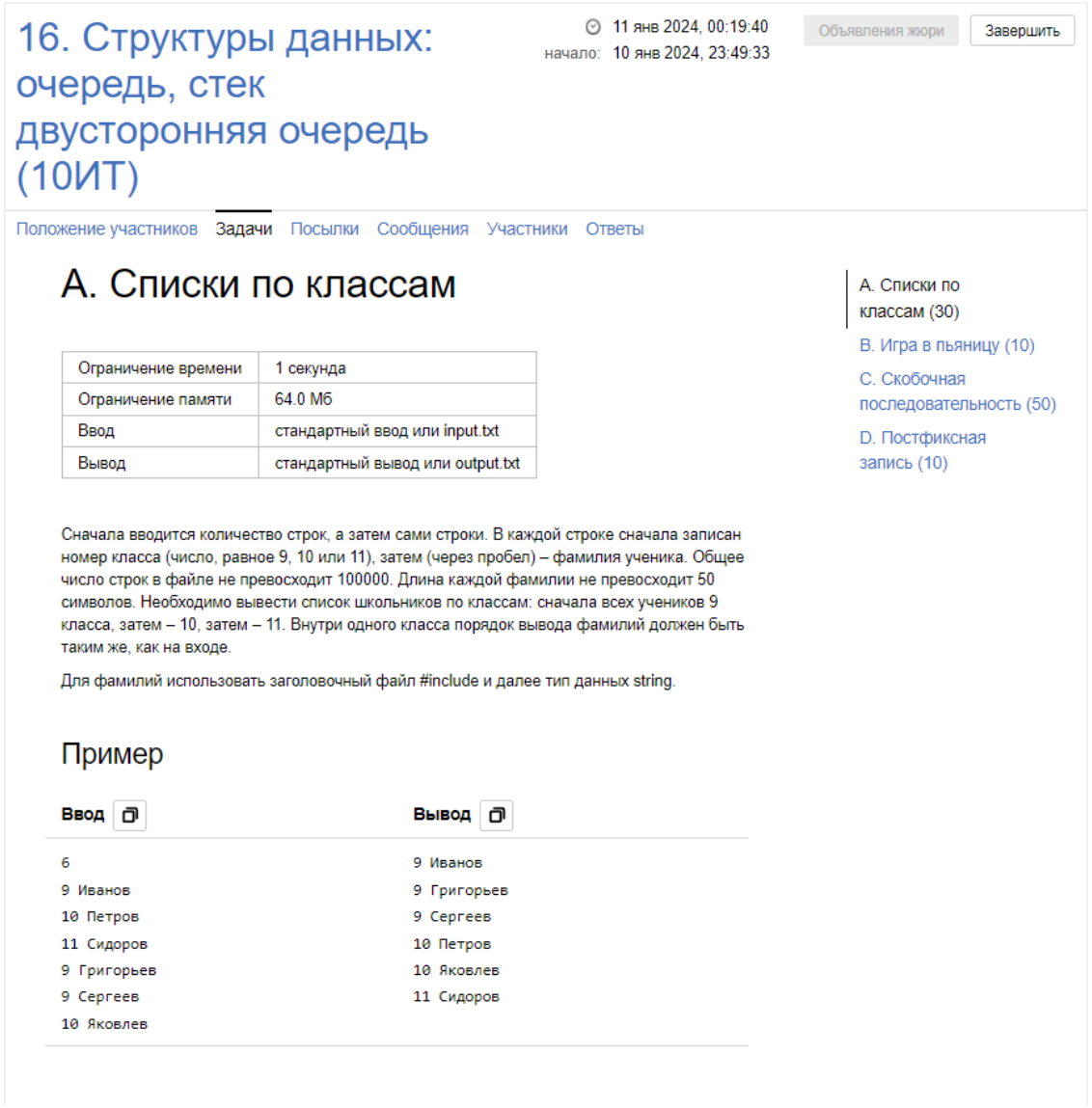
Методы двусторонних очередей

push_back(val): добавляет значение val в конец очереди
push_front(val): добавляет значение val в начало очереди
insert(pos, val): вставляет элемент val на позицию, на которую указывает итератор pos, аналогично функции emplace. Возвращает итератор на добавленный элемент
insert(pos, n, val): вставляет n элементов val начиная с позиции, на которую указывает итератор pos. Возвращает итератор на первый добавленный элемент. Если $n = 0$, то возвращается итератор pos.
insert(pos, begin, end): вставляет начиная с позиции, на которую указывает итератор pos, элементы из другого контейнера из диапазона между итераторами begin и end. Возвращает итератор на первый добавленный элемент. Если между итераторами begin и end нет элементов, то возвращается итератор pos.
insert(pos, values): вставляет список значений values начиная с позиции, на которую указывает итератор pos. Возвращает итератор на первый добавленный элемент. Если values не содержит элементов, то возвращается итератор pos.

Методы двусторонних очередей

clear(p): удаляет все элементы
pop_back(): удаляет последний элемент
pop_front(): удаляет первый элемент
erase(p): удаляет элемент, на который указывает итератор p. Возвращает итератор на элемент, следующий после удаленного, или на конец контейнера, если удален последний элемент
erase(begin, end): удаляет элементы из диапазона, на начало и конец которого указывают итераторы begin и end. Возвращает итератор на элемент, следующий после последнего удаленного, или на конец контейнера, если удален последний элемент

Ссылка на контест



16. Структуры данных: очередь, стек двусторонняя очередь (10ИТ)

11 янв 2024, 00:19:40
начало: 10 янв 2024, 23:49:33

Объявления жюри | Завершить

Положение участников | Задачи | Посылки | Сообщения | Участники | Ответы

A. Списки по классам

Ограничение времени	1 секунда
Ограничение памяти	64.0 Мб
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Сначала вводится количество строк, а затем сами строки. В каждой строке сначала записан номер класса (число, равное 9, 10 или 11), затем (через пробел) – фамилия ученика. Общее число строк в файле не превосходит 100000. Длина каждой фамилии не превосходит 50 символов. Необходимо вывести список школьников по классам: сначала всех учеников 9 класса, затем – 10, затем – 11. Внутри одного класса порядок вывода фамилий должен быть таким же, как на входе.

Для фамилий использовать заголовочный файл #include и далее тип данных string.

Пример

Ввод	Вывод
6 9 Иванов 10 Петров 11 Сидоров 9 Григорьев 9 Сергеев 10 Яковлев	9 Иванов 9 Григорьев 9 Сергеев 10 Петров 10 Яковлев 11 Сидоров

A. Списки по классам (30)
B. Игра в пьяницу (10)
C. Скобочная последовательность (50)
D. Постфиксная запись (10)

contest.yandex.ru/contest/58209