Задание 1

Руководство «Предприятия» оказалось в целом удовлетворено вашей работой над программой для определения допустимой скобочной структуры и приняло решения доверить вам решить ещё одну задачу, в которой необходимо использование линейных структур данных.

«Предприятие» является настолько крупным предприятием, что в дни приёма по личным вопросам к руководителям структурных подразделений предприятия выстраиваются длинные очереди из десятков посетителей. Иногда люди пытаются «просто спросить». Некоторые заявляют о том, что человек, занимавший очередь перед ними, на самом деле никакой очереди не занимал. Иногда дело доходит до драк.

Руководство «Предприятия» поставило задачу написать программу, реализующую электронную очередь, призванную упорядочить поток посетителей.

Отдел постановки задач департамента разработки программного обеспечения сформулировал для вас как для исполнителя следующие требования в техническом задании.

**Часть 1.**

1) Программа должна иметь имя 2022\_SAS02\_01\_Фамилия, где фамилия – фамилия автора программы на латинице.

2) Программа должна использовать функции, разработанные непосредственно исполнителем.

3) Структура программы должна быть следующей: главная функция main выводит меню, определяет, какую операцию хочет выполнить пользователь, и в зависимости от этой операции будет вызывать нужную функцию.

4) Описывая любую функцию, следует исходить из того, что функция при вызове принимает фактические параметры, в процессе работы выполняет некоторые вычисления и преобразования с использованием этих параметров и в конце работы возвращает результат при помощи оператора **return**. **Внутри функции не должен происходить ввод и вывод информации** (если только эта функция специально не предназначена для ввода и вывода). Функция принимает входные параметры из вне, и возвращает результат работы.

5) Для работы с очередью должны использоваться следующие функции:

5.1) int isEmpty(int front, int rear), возвращает 1, если очередь пуста, иначе возвращает 0;

5.2) int isFull(int front, int rear), возвращает 1, если очередь заполнена, иначе возвращает 0;

5.3) void enqueue(int \*queue, int front, int& rear, int m), ставит элемент m в очередь. Должна быть вызвана, только если isFull возвращает 0, таким образом проводить проверку на заполненность очереди внутри функции enqueue нет необходимости.

5.4) int dequeue(int \*queue, int &front), выводит элемент из очереди. Должна быть вызвана, только если isEmpty возвращает 0, таким образом, проводить проверку на пустоту очереди внутри функции dequeue нет необходимости. Функция возвращает значение выведенного элемента.

5.5) void view(int \*queue, int front, int rear) выводит содержимое очереди. Хитрость данной функции заключается в том, что очередь может располагаться в массиве по разному: либо очередь ещё не закольцевалась (все элементы располагаются между front и rear), либо очередь уже закольцевалась (тогда элементы располагаются от front до SIZE (размера очереди) и далее от 0 до rear).

**Часть 2.**

Руководство предприятия сообщает, что очередность посещения руководства определяется не только порядком постановки в очередь, но и приоритетом, который определяется лицом, выдавшим талон. Люди с приоритетом, имеющим меньшее значение (1 приоритет) должны попасть на приём раньше людей, имеющими большее значение приоритета (2 приоритет, 3 приоритет). При этом люди с одинаковым приоритетом должны вызываться на приём в порядке обыкновенной очередности.

1) Программа должна иметь имя 2022\_SAS02\_02\_Фамилия, где фамилия – фамилия автора программы на латинице.

2) Программа должна использовать функции, разработанные непосредственно исполнителем.

3) Структура программы должна быть такая же, как и структура программы 2022\_SAS02\_01\_Фамилия.

4) Описывая любую функцию, следует исходить из того, что функция при вызове принимает фактические параметры, в процессе работы выполняет некоторые вычисления и преобразования с использованием этих параметров и в конце работы возвращает результат при помощи оператора **return**. **Внутри функции не должен происходить ввод и вывод информации** (если только эта функция специально не предназначена для ввода и вывода). Функция принимает входные параметры из вне, и возвращает результат работы.

5. Для упрощения положим, что приоритетом элемента в очереди является его значение. То есть в очереди, содержащей 10 2 3 45 6, элементом с наименьшим значением приоритета будет элемент 2. Главное отличие при реализации приоритетной очереди от обычной очереди в том, что не будет реализовываться циклическая организация очереди. Начало очереди всегда будет в элементе массива номер 0. Таким образом, необходимость в указателе на начало очереди front отпадает. Нужен только указатель на конец – rear: он будет содержать номер последнего элемента в очереди. Например, если очередь содержит 0 15 3 6 4 7, rear содержит значение 5.

5) Для работы с очередью должны использоваться следующие функции:

5.1) int isEmpty(int \*priorityQueue, int rear), возвращает 1, если очередь пуста, иначе возвращает 0. Для работы с приоритетной очередью положим, что очередь считается пустой, если указатель на конец очереди rear равен -1 (как в случае со стеком);

5.2) int isFull(int \*priorityQueue, int rear), возвращает 1, если очередь заполнена, иначе возвращает 0. Нужно анализировать только значение указателя на конец очереди. Очередь будет считаться заполненной, если значение указателя на последний элемент массива – номер последней ячейки массива;

5.3) void enqueue(int \*priorityQueue, int& rear, int m), ставит элемент m в очередь. Должна быть вызвана, только если isFull возвращает 0, таким образом проводить проверку на заполненность очереди внутри функции enqueue нет необходимости. Следовательно, эта функция просто увеличивает rear на 1 и записывает элемент в очередь;

5.4) void view(int \*priorityQueue, int front, int rear) выводит содержимое очереди;

5.5) int deleteMin(int \*priorityQueue, int &rear), выводит элемент из очереди. Должна быть вызвана, только если isEmpty возвращает 0, таким образом, проводить проверку на пустоту очереди внутри функции deleteMin нет необходимости. Функция возвращает значение выведенного элемента. Это единственная функция, над которой придётся потрудиться. Эта функция должна удалять из очереди элемент с наименьшим приоритетом. Алгоритм работы этой функции может быть таким:

найти первый элемент с наименьшим приоритетом

сохранить найденный элемент

сдвинуть все элементы очереди к началу массива

уменьшить указатель rear

вернуть значение удаленного элемента

Рассмотрим очередь (голубым обозначено, куда указывает rear):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 5 | 45 | 3 | 12 | 17 |

Наименьший приоритет у элемента 3. Следовательно, нужно сдвинуть влево элементы 12 и 17:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 5 | 45 | 12 | 12 | 17 |

Сдвинули элемент 12.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 5 | 45 | 12 | 17 | 17 |

Сдвинули элемент 17.

Оценка выполнения каждого задания (2022\_SAS02\_01 и 2022\_SAS02\_02) производится следующим образом:

Оценка 5 (отлично): все пункты задания выполнены полностью, программа является работоспособной, исполнитель обоснованно отвечает на контрольные вопросы.

Оценка 4 (хорошо): все пункты задания выполнены полностью, программа является работоспособной, исполнитель хорошо отвечает на контрольные вопросы, но не всегда может обосновать их.

Оценка 3 (удовлетворительно): пункты задания выполнены частично, программа является частично работоспособной, исполнитель затрудняется в ответах на контрольные вопросы.

Оценка 2 (неудовлетворительно): пункты задания выполнены частично, программа не является работоспособной, исполнитель затрудняется в ответах на контрольные вопросы.

Соответствие оценок баллам балльно-рейтинговой системы при оценивании выполнения практической работы:

Оценка 5 (отлично) – 2 балла;

Оценка 4 (хорошо) – 1,5 балла;

Оценка 3 (удовлетворительно) – 1 балл;

Оценка 2 (неудовлетворительно) – 0 баллов.