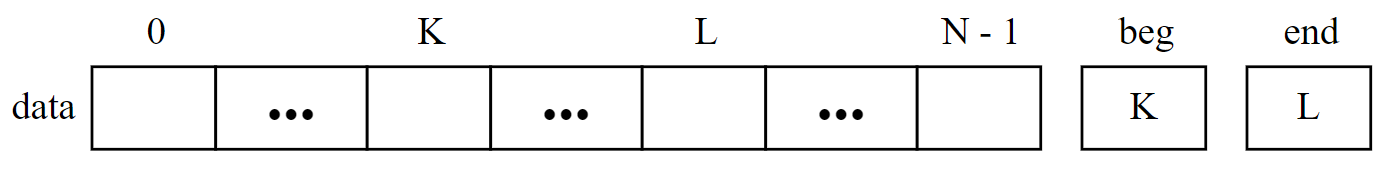
|  |  |  |
| --- | --- | --- |
|  | | |
| Федеральное государственное бюджетное  образовательное учреждение высшего образования FPMI_ngtu_neti_rgb_polya«Новосибирский государственный технический университет» | | |
|  | | |
| Кафедра теоретической и прикладной информатики | | |
| Лабораторная работа № 2 | | |
| по дисциплине «Основы программирования» | | |
| **Структуры данных: стеки и очереди** | | |
|  | | |
|  | Бригада | ФИО |
| №2 | Веселый Денис Андреевич |
|  | Ворончук Илья Игоревич |
| Группа | ПМИ-32 |
|  |  |
|  |  |
| Преподаватель | Дворецкая виктория константиновна |
|  |  |
| Новосибирск,2024 | | |

**1) Цель работы**

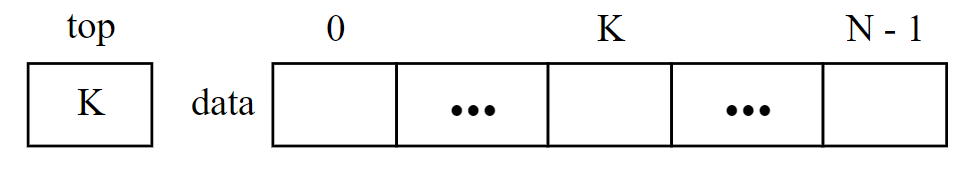
Сформировать практические навыки организации таких распространенных структур, как стеки и очереди, и их использования при решении задач

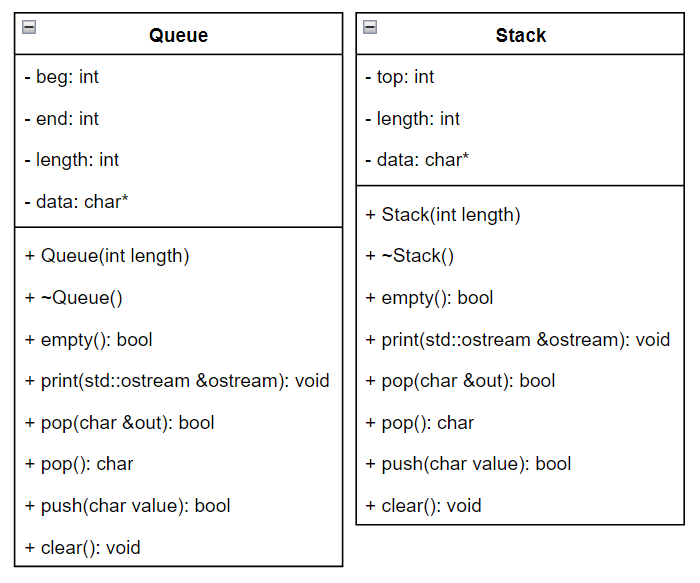
**2) Условие задачи**

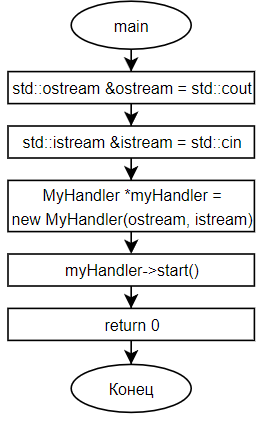
2.1) Очередь

Линейный массив с двумя указателями (на начало и на конец очереди). Если указатель на конец очереди достиг конца массива, а в начале очереди есть свободные позиции (очередь псевдополна), то все элементы очереди сдвинуть в ее начало.

2.2) Стек

Линейный массив с указателем на вершину стека.

**3) Стандартная схема программы**

****

**4) Код программы**

**Queue.cpp**

#include <iostream>

#include "proto/Queue.h"

Queue::Queue(int length)

{

beg = 0;

end = -1;

this->length = length;

data = new char[length];

for (int i = 0; i < length; i++)

data[i] = '\0';

}

Queue::~Queue()

{

delete data;

data = nullptr;

}

// проверяет, пуста ли очередь

bool Queue::empty()

{

return end == -1;

}

// помещает элементы очереди в поток вывода

void Queue::print(std::ostream &ostream)

{

if (empty())

{

ostream << "Queue is empty";

return;

}

for (int i = 0; i < length; i++)

ostream << data[i] << " ";

}

// вытаскивает элемент из очереди и возвращает его

char Queue::pop()

{

char buffedElement = data[beg];

data[beg] = '\'';

if (beg == end)

{

beg = 0;

end = -1;

}

else

{

beg++;

}

return buffedElement;

}

// вытаскивает элемент из очереди и помещает в параметр out. Возвращает false, если очередь пуста, иначе – true

bool Queue::pop(char &out)

{

if (empty())

return 0;

out = pop();

return 1;

}

// помещает переданное значение в очередь. Возвращает false, если очередь полна, иначе – true

bool Queue::push(char value)

{

if (end == length - 1)

{

// в случае, если очередь псевдо-полна

if (beg != 0)

{

for (int i = beg; i < length; i++)

{

char buffer = data[i];

data[i] = '\0';

data[i - beg] = buffer;

}

end -= beg;

beg = 0;

}

else

return 0;

}

data[++end] = value;

return 1;

}

// очищает очередь

void Queue::clear()

{

for (int i = 0; i < length; i++)

data[i] = '\0';

}

**Stack.cpp**

#include <iostream>

#include "proto/Stack.h"

Stack::Stack(int length)

{

top = -1;

this->length = length;

data = new char[length];

for (int i = 0; i < length; i++)

data[i] = '\0';

}

Stack::~Stack()

{

delete data;

data = nullptr;

}

// проверяет, пуст ли стек

bool Stack::empty()

{

return top == -1;

}

// помещает элементы стека в поток вывода

void Stack::print(std::ostream &ostream)

{

if (empty())

{

ostream << "Stack is empty";

return;

}

for (int i = 0; i < length; i++)

{

char e = data[i];

ostream << e << " ";

}

}

// вытаскивает элемент из стека и возвращает его

char Stack::pop()

{

char b = data[top];

data[top--] = '\'';

return b;

}

// вытаскивает элемент из стека и помещает в параметр out. Возвращает false, если стек пуст, иначе – true

bool Stack::pop(char &out)

{

if (empty())

return 0;

out = pop();

return 1;

}

// помещает переданное значение в стек. Возвращает false, если стек полон, иначе – true

bool Stack::push(char value)

{

if (top == length - 1)

return 0;

data[++top] = value;

return 1;

}

// очищает стек

void Stack::clear()

{

for (int i = 0; i < length; i++)

data[i] = '\0';

}

**5) Тестирование программы**

|  |  |  |
| --- | --- | --- |
| № | Команды | Комментарий |
| 1 | sw q  5  psh q w e r t  pop  prt  psh y  prt  mt | 1. Инициализация очереди размерности 5. 2. Добавление элементов: ‘q’, ‘w’, ‘e’, ‘r’, ‘t’. 3. Получение первого (левого) элемента ‘q’ из очереди. 4. Вывод очереди в консоль: ' w e r t. 5. Добавление элемента ‘y’ в очередь. 6. Вывод очереди в консоль: w e r t y – так как очередь была псевдополна, то – сдвиг элементов в начало и добавление нового элемента в освободившуюся ячейку. 7. Проверка очереди на пустоту – ложь. |
| 2 | sw s  5  psh a b c d e  pop  pop  prt  psh m  prt | 1. Инициализация стека размерности 5. 2. Добавление элементов: ‘a’, ‘b’, ‘c’, ‘d’, ‘e’. 3. Удаление из стека и вывод в консоль двух элементов: ‘a’, ‘b’. 4. Вывод стека в консоль: a b c ' '. 5. Добавление элемента ‘m’ в стек. 6. Вывод стека в консоль: a b c m '. |
| 3 | sw q  2  psh m n  cl  mt | 1. Инициализация очереди размерности 2. 2. Добавление элементов ‘m’, ‘n’. 3. Очищение очереди. 4. Проверка очереди на пустоту – истина. |

**6) Результаты отладки**

Результаты работы программы показали, что программа работает успешно.