

**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технологический университет «СТАНКИН»**

**(ФГБОУ ВО «МГТУ «СТАНКИН»)**

|  |  |
| --- | --- |
| **Институт**  **информационных**  **технологий** | **Кафедра**  **информационных технологий и вычислительных систем** |

**КУРСОВАЯ РАБОТА**  
ПО ДИСЦИПЛИНЕ

«СТРУКТУРЫ И АЛГОРИТМЫ ОБРАБОТКИ ДАННЫХ»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТА | *2* | КУРСА | *Бакалавриата* | ГРУППЫ | *ИДБ-19-03* |
|  | | |  |  | |

|  |
| --- |
| **СМАРАНДИ ОЛЕГА** |
|  |

ТЕМА РАБОТЫ

|  |
| --- |
| Очередь векторов целого типа |

|  |  |
| --- | --- |
| Направление: | 09.03.01 Информатика и вычислительная техника |
| Профиль подготовки: | Программное обеспечение средств вычислительной техники и автоматизированных систем |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | Отчет сдан «\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_г. | | | | |  |  |  |  | | Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | | |  | | | | | Преподаватель | Лакунина О.Н., ст. преподаватель |  |  | |  |  |  | *(подпись)* | |

**Оглавление**

**Задание на курсовую работу .................................................................................................... 3**

**Описание структур данных ..................................................................................................... 4**

**Конечная схема реализуемой структуры данных ............................................................... 8**

**Описание структур на языке C .............................................................................................. 10**

**Схема вызова функций ............................................................................................................ 11**

**Список функций и их назначение ........................................................................................... 12**

**Исходный код программы с комментариями ...................................................................... 15**

**Задание на курсовую работу**

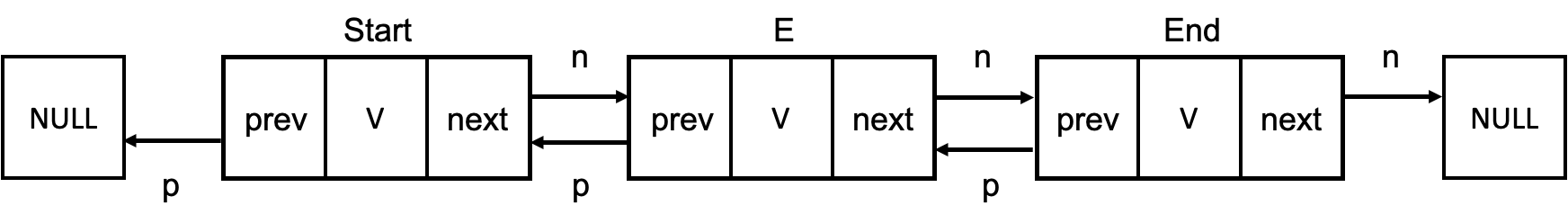
Написать программу, реализующую логическую структуру данных — очередь векторов целого типа . Программа должна работать в диалоговом режиме. Каждая операция должна быть реализована в отдельной функции.

Очередь должна быть реализована на базе структуры хранения двусвязный  список.

Написать отчет по выполненной работе.

**Описание структур данных**

**Двусвязный список** – это структура данных, состоящая из элементов одного типа, связанных между собой последовательно посредством указателей. Каждый элемент списка имеет указатель на следующий и на предыдущий элементы. Указатель на следующий элемент списка у последнего элемента и указатель на предыдущий элемент и первого элемента списка указывают на NULL. Также должны быть указатели на начало списка и на конец.



|  |  |
| --- | --- |
| v | Переменная, хранящая значение элемента списка |
| prev | Переменная, хранящая указатель на предыдущий элемент списка |
| next | Переменная, хранящая указатель на следующий элемент списка |
| n | Указатель на следующий элемент списка |
| p | Указатель на предыдущий элемент списка |
| Start | Первый элемент списка |
| End | Последний списка |
| E | Очередной элемент списка |

***Вектор***– конечное упорядоченное множество простых данных, или скаляров, одного и того же типа, называемых элементами вектора. Элементы вектора находятся друг с другом в отношении непосредственного следования. Строгая упорядоченность элементов вектора позволяет пронумеровать их последовательными целыми числами, которые называются индексами или селекторами элементов. Каждому элементу вектора ставится в соответствие определенное значение индекса, которое дает возможность однозначно идентифицировать соответствующий элемент. Логическая структура вектора полностью определяется числом и типом элементов вектора. Физическая структура вектора представляется в машинной памяти последовательностью одинаковых по длине участков памяти, называемых полями или слотами, каждый из которых предназначен для хранения одного элемента вектора. Слот может иметь размер минимальной адресуемой ячейки памяти (например, размер в 1 байт) или соответствовать целой группе последовательных ячеек памяти. Если слот состоит из нескольких ячеек памяти, то его адресом обычно считают адрес самой левой ячейки.



|  |  |
| --- | --- |
| vector | Указатель на 0-ой элемент вектора |
| N | Количество элементов вектора |
| i | Индекс |

***Очередь***– такой последовательный список с переменной длиной, включение элементов в который происходит с одной стороны, а исключение элементов с другой стороны. Включение элементов производится с конца очереди. Исключение - с начала очереди.



|  |  |
| --- | --- |
| START | Указатель на начало очереди |
| END | Указатель на конец очереди |
| V | Значение |
| NEXT | Указатель на следующий элемент очереди |
| PREV | Указатель на предыдущий элемент очереди |

**Реализуемые функции для очереди**

1. Начать работу,
2. Сделать очередь пустой
3. Проверка, пуста или не пуста
4. Показать значение элемента в начале очереди
5. Удалить начало очереди
6. Взять элемент из начала очереди
7. Изменить значение элемента в начале очереди
8. Добавить элемент в конец очереди
9. Распечатать очередь
10. Закончить работу с очередью

11. Закончить работу с программой

**Конечная схема реализуемой структуры данных**



|  |  |
| --- | --- |
| START | Указатель на начало очереди |
| END | Указатель на конец очереди |
| NEXT | Указатель на следующий элемент очереди |
| PREV | Указатель на предыдущий элемент очереди |
| V | Значение |
| vector | Указатель на 0-ой элемент вектора |
| i | Индекс |
| N | Количество элементов вектора |

**Описание структур на языке С++**

**Схема вызова функций**

**Список функций и их назначение**

**Исходный код программы с комментариями**

**main.cpp**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include "Line.h"

int main()

{

Line my\_line;

my\_line.menu();

}

**Line.h**

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

class Line // Структура описывающая очередь

{

class Node // Элемент структуры

{

public:

int data[20] = { 0 }; // Статичный вектор

int size; // Его длина

Node\* next; // Указатель на следующий элемент

Node\* prev; // Указатель на пред элемент

Node() :next(0), size(0), prev(0)

{}

};

Node\* first; // Указатель на следующий элемент

int max\_size; // Дно очереди

void New(int\* mass, int size) // Содание первого элемента

{

first = (Node\*)malloc(sizeof(Node));

\*first = Node();

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

first->data[i] = mass[i];

first->size = size;

}

public:

Line() : first(0), max\_size(15)

{}

~Line() // Деструктор

{

Node\* ptr = NULL, \* pdel;

ptr = first;

while (ptr)

{

pdel = ptr;

ptr = ptr->next;

free(pdel);

}

first = 0;

}

void append(int\* mas, int size) // Функция добавляющая в конец очереди

{

Node\* ptr = first;

if (!first)

{

New(mas, size); // Если нет элемент вызывает метод для создания очереди

return;

}

while (ptr->next)

ptr = ptr->next;

ptr->next = (Node\*)malloc(sizeof(Node)); // Выделяем память под элемент

ptr->next->prev = ptr; // Сохраняем связь

ptr = ptr->next;

ptr->next = 0; // Обнуляем указатель на след элемент

for (int i = 0; i < size; i++) // Заносим данные в дату

ptr->data[i] = mas[i];

ptr->size = size;

}

void del\_first() // удаляем начало очереди

{

if (!first)

return;

Node\* ptr = first;

if (first->next)

{

first = first->next;

first->prev = 0;

free(ptr);

}

else

{

free(ptr);

first = 0;

}

}

bool eof()

{

if (first)

return 1;

return 0;

}

void prmenu()

{

puts(" \

\r1)Сделать очередь пустой \n\

\r2)Проверка на пустую очередь \n\

\r3)Показать первый вектор \n\

\r4)Удалить первый вектор \n\

\r5)Взять первый вектор \n\

\r6)Изменить первый вектор \n\

\r7)Добавить вектор в конец \n\

\r8)Вывести очередь \n\

\r0)Завершить работу \n\

");

}

void print\_line() // Выводим всю очередь

{

Node\* ptr = first;

while (ptr)

{

for (int i = 0; i < ptr->size; i++)

printf("%d\t", ptr->data[i]);

puts("");

ptr = ptr->next;

}

}

void menu()

{

int choice, size, \* pointer\_arr;

int arr[20];

do

{

puts("1)Начать работу\n\

\r2)Закончить работу");

scanf("%d", &choice);

getchar();

if (choice == 1)

do

{

prmenu(); // Вызов текста меню

scanf("%d", &choice);

getchar();

switch (choice)

{

case 0:

this->~Line(); // Удаление очереди

break;

case 1:

this->~Line(); // Удаление очереди

break;

case 2:

if (!first) // Проверка

puts("Очередь пуста");

else

puts("В очереди есть элементы");

break;

case 3:

if (first)

for (int i = 0; i < first->size; i++)

printf("%d\t", first->data[i]);

else

puts("Очередь пуста");

puts("");

break;

case 4:

if (first)

{

del\_first(); // Удаляем первый элемент

print\_line(); // Выводим очередь

putchar('\n');

}

else

puts("Очередь пуста");

break;

case 5:

if (first)

{

for (int i = 0; i < first->size; i++)

arr[i] = first->data[i];

size = first->size;

del\_first();

print\_line();

putchar('\n');

printf("Взятый элемент:\n");

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

printf("%d\t", arr[i]);

puts("\n");

}

else

puts("Очередь пуста");

break;

case 6:

do {

puts("Введите длину вектора");

scanf("%d", &size);

if (size > 20 || size <= 0)

puts("Не корректный размер");

} while (size > 20 || size <= 0);

puts("Введите вектора");

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

scanf("%d", &first->data[i]);

first->size = size;

print\_line();

putchar('\n\n');

break;

case 7:

do {

puts("Введите длину вектора");

scanf("%d", &size);

if (size > 20 || size <= 0)

puts("Не корректный размер");

} while (size > 20 || size <= 0);

puts("Введите вектор");

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

scanf("%d", &arr[i]);

append(arr, size);

print\_line();

putchar('\n\n');

break;

case 8:

if (first)

{

print\_line(); //Вывод очереди

putchar('\n');

}

else

puts("Очередь пуста");

break;

default:

break;

}

} while (choice);

} while (choice != 2);

}

};