

**Министерство образования Республики Беларусь
Учреждение образования «Полоцкий государственный
университет»**

Кафедра технологий
программирования

Основы алгоритмизации и программирования
Отчет по лабораторной работе №8
Вариант 11

Выполнил

Ланцев Евгений Николаевич.
21-ИТ-1, ФИТ

Проверил

Пантелейко А.Ф.
Преподаватель-стажер

Полоцк
2022 г.

Лабораторная работа № 8

“Динамические структуры данных: очереди, стеки, деки”

Цель работы: изучить методы организации списочных структур в динамической памяти. Реализовать алгоритмы помещения и изъятия элементов из стека, дека или очереди.

Вариант 11

“Дек типа char[20]”

```
struct node {  
    std::string data; // указатель на адрес data в  
    node *next; // указатель на следующий  
    node *prev; // указатель на предыдущий  
  
    node(std::string c, node *n, node *p){  
        c.resize(20); data = c; next = n; prev = p;  
    }  
};
```

Рисунок 1 - Структура узла.

```
class deque {  
    node *root; // Указатель на корневой узел  
    node *end; // Указатель на конечный узел  
  
public:  
    deque(std::string c_r, std::string c_e){ //  
        root = new node(c_r, nullptr, nullptr);  
        end = new node(c_e, nullptr, root); // C  
        root->next = end;  
    }  
};
```

Рисунок 2 - Класс дека.

```

void insert_root_node(std::string data){ // Вставка по
    node *new_node = new node(data,root→next,root); /
    root→next→prev = new_node; root→next = new_node
}

void insert_end_node(std::string data){ // Вставка до
    node *new_node = new node(data,end,end→prev); //
    end→prev→next = new_node; end→prev = new_node;
}

```

Рисунок 3 - Методы добавления узла дека.

```

void remove_root_node(deque *d){
    if(is_deque_empty(d)){
        return;
    }
    node* pnter = d→root→next;
    d→root→next = pnter→next;
    pnter→next→prev = d→root;
    delete pnter;
}

void remove_end_node(deque *d){
    if(is_deque_empty(d)){
        return;
    }
    node* pnter = d→end→prev;
    d→end→prev = pnter→prev;
    pnter→prev→next = d→end;
    delete pnter;
}

```

Рисунок 4 - Методы удаления узла из дека.

```

void print_deque(){
    node * temp = root;
    while(temp){
        std::cout << temp→data << " "; temp = temp→next;
    }
}

```

Рисунок 5 - Метод вывода узлов из дека.

```

void done_deque(deque *d){
    while(!is_deque_empty(d)){
        remove_root_node(d);
    }
    delete d->root; delete d->end;
}

```

Рисунок 6 - Метод удаления всех узлов из дека.

```

Input a root node → 1
Input a end node → hello

Choose task :
1-Add left node
2-Add right node
3-Delete right node
4-Delete left node
5-Print all nodes
6-Clean DSD
Input a task number →1
Input a node →world
world
Choose task :
1-Add left node
2-Add right node
3-Delete right node
4-Delete left node
5-Print all nodes
6-Clean DSD
Input a task number →5
1 world hello
Choose task :
1-Add left node
2-Add right node
3-Delete right node

```

Рисунок 7 - Работа программы.

Вариант 11
“очередь FIFO”

```
struct node {  
    float data; // указатель  
    node *next; // указатель  
  
    node(float c, node *n){  
        data = c; next = n;  
    }  
};
```

Рисунок 1 - Структура узла.

```
class quene {  
    node *root; // Указатель на корнев  
  
public:  
    quene(float data ) { // конструктор  
        root = new node(data, nullptr);  
    }  
};
```

Рисунок 2 - Класс очереди.

```
void insert_node(float data) { // Вставка до  
    node* last = get_last(root);  
    node* new_node = new node(data, nullptr);  
    last->next = new_node; // Перестановка п  
}
```

Рисунок 3 - Методы добавления узла в очередь.

```
void remove_node(queene *d){
    node* new_root = root->next;
    delete root;
    root = new_root;
}
```

Рисунок 4 - Методы удаления узла из очереди.

```
void print_quene(){
    node * temp = root;
    while(temp){
        std::cout << temp->data << " "; temp = temp->next;
    }
}
```

Рисунок 5 - Метод вывода узлов из очереди.

```
void done_quene(queene* d){
    while(root){
        remove_node(d);
    }
    delete root;
}
```

Рисунок 6 - Метод удаления всех узлов из очереди.

```
Input a root node → 1

Choose task :
1-Add node
2-Remove node
3-Print all nodes
4-Clean DSD
Input a task number →1
Input a node →5

Choose task :
1-Add node
2-Remove node
3-Print all nodes
4-Clean DSD
Input a task number →3
1 5
Choose task :
1-Add node
2-Remove node
3-Print all nodes
4-Clean DSD
Input a task number →2
```

Рисунок 7 - Работа программы.

Вывод: Я изучил методы организации списочных структур в динамической памяти. Реализовать алгоритмы помещения и изъятия элементов из стека, дека или очереди