Министерство образования и науки Российской Федерации

Федеральное‌ ‌государственное‌ ‌бюджетное‌ ‌образовательное‌ ‌учреждение‌

высшего‌ ‌образования‌

**«Пермский национальный исследовательский**

**политехнический университет»**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

**О Т Ч Ё Т**

**по лабораторной работе №11.3**

Дисциплина: «Основы алгоритмизации и программирования»

Тема: Информационные и динамические структуры. Стеки

Вариант 18

Выполнил:

Студент группы РИС-20-1б

Тараканов Д. М.

Проверила:

Доцент кафедры ИТАС

Полякова О. А.

Пермь

2021 год

# **Цель работы**

1) Получить практические навыки работы со стеками;

2) Получить практические навыки работы со стеками через STL;

# **Постановка задачи**

Задача состоит в разработке программы, в которой создаётся стек и стек через STL, выполнить обработку стеков в соответствии со своим вариантом:

(18 вариант)

Стек

1. Сформировать стек с информационным полем типа double.
2. Удалить из стека все элементы с четными индексами (2, 4, 6 и т.д.).

Очередь через STL

1. Сформировать стек через STL с информационным полем типа double.
2. Удалить из стека все элементы с четными индексами (2, 4, 6 и т.д.).

# **Анализ задачи**

Стек

1. Определить какие предстоит выполнить действия:

* Создать структуру стек в виде стека

struct Stack {

char data;

Stack\* prev;

};

* Разработать функцию FillStack для заполнения стека

void\* FillStack(int N, Stack\*& top) {

Stack\* p = new Stack;

double el = -1;

cout << "Введите 1 элемент: ";

cin >> el;

p->data = el;

p->prev = nullptr;

top = p;

for (int i = 2; i <= N; i++) {

Stack\* k = new Stack;

cout << "Введите " << i << " элемент: ";

cin >> el;

k->data = el;

k->prev = top;

top = k;

}

return top;

};

* Разработать функцию DeleteElement для удаление элементов с четными индексами в стеке.

void DeleteElement(Stack\* top, int N)

{

Stack\* p = top;

double key = 0;

while (p->prev != nullptr) {

if (p->prev->data == key) {

Stack\* q = p->prev;

p->prev = p->prev->prev;

delete q;

}

else {

p = p->prev;

}

}

};

2. С какими типами данных действие надо сделать, в каком виде эти данные будут представлены:

* Для запоминания длины стека используем переменную size типа int.

int size = -1;

* Для данных в очереди используем переменную data типа double.

double data;

* Для хранения адреса предыдущего элемента используем указатель prev типа Stack.

Stack\* prev;

* Для хранения нового элемента используем переменную el типа double.

double el;

* Для обращения к стеку используем указатель stack на верхний элемент стека типа Stack.

Stack\* stack;

* Для функции FillStack используем следующие элементы.

1. Длину стека типа int.
2. Указатель на верхний элемент стека типа Stack

Сама функция имеет тип void так, как входе её работы функция не возвращает значений.

void\* FillStack(int N, Stack\*& top)

* Для функции PrintStack используем следующие элементы.

1. Указатель на верхний элемент стека типа Stack.

Сама функция имеет тип void так, как входе её работы функция не возвращает значений.

void PrintStack(Stack\* top)

* Для функции ZeroElement используем следующие элементы.

1. Указатель на верхний элемент стека типа Stack.
2. Длину стека типа int.

Сама функция имеет тип void так, как входе её работы функция не возвращает значений.

void ZeroElement(Stack\* top, int size)

* Для функции DeleteElement используем следующие элементы.

1. Указатель на верхний элемент стека типа Stack.
2. Длину стека типа int.

Сама функция имеет тип void так, как входе её работы функция не возвращает значений.

void DeleteElement(Stack\* top, int size)

3. Для решения задачи данные были представлены в виде множества элементов типа double.

4. Реализация операций ввода и вывода данных:

* Ввод данных с консоли реализован при помощи цикла for и оператора cin.

for (int i = 2; i <= N; i++) {

Stack\* k = new Stack;

cout << "Введите " << i << " элемент: ";

cin >> el;

k->data = el;

k->prev = top;

top = k;

}

* Вывод данных на консоль реализован при помощи цикла while и оператора cout.

while (p != nullptr) {

cout << p->data << " ";

p = p->next;

}

5. Поставленные задачи будут решены при помощи следующих действий:

* Программа запрашивает у пользователя длину стека.

while (size < 0) {

cin >> size;

if (size < 0) {

cout << "Введите размер стека: ";

}

}

* При помощи функции FillStack происходит заполнение стека с консоли.

void\* FillStack(int N, Stack\*& top)

{

Stack\* p = new Stack;

double el=-1;

top = nullptr;

cout << "\nВведите 1 элемент: ";

do {

cin >> el;

if (el==0) {

cout << "Введите 1 элемент: ";

}

} while (el == 0);

p->data = el;

p->prev = nullptr;

top = p;

for (int i = 2; i <= N; i++) {

Stack\* k = new Stack;

cout << "Введите " << i << " элемент: ";

do {

cin >> el;

if (el == 0)

{

cout << "Введите " << i << " элемент: ";

}

} while (el == 0);

k->data = el;

k->prev = top;

top = k;

}

return top;

};

* При помощи функции ZeroElement происходит обнуление значений элементов с четным индексом.

void ZeroElement(Stack\* top,int N) {

Stack\* p = top;

if (N % 2 != 0) {

while (p->prev != nullptr) {

p = p->prev;

p->data = 0;

if (p->prev != nullptr) {

p = p->prev;

}

}

}

else {

while (p->prev != nullptr) {

p->data = 0;

p = p->prev;

if (p->prev != nullptr) {

p = p->prev;

}

}

}

};

* При помощи функции DeleteElement происходит удаление элементов с нулевым значением. При помощи оператора while происходит проверка на то является ли значение элемента стека нулевым, а затем происходит его удаления из стека. Удаление элемента происходит путем изменения адресов элементов.

void DeleteElement(Stack\* top, int N)

{

Stack\* p = top;

double key = 0;

while (p->prev != nullptr) {

if (p->prev->data == key) {

Stack\* q = p->prev;

p->prev = p->prev->prev;

delete q;

}

else {

p = p->prev;

}

}

}

* При помощи функции PrintStack программа выводит элементы стека на консоль.

void PrintStack(Stack\* top)

{

Stack\* p = top;

while (p != nullptr) {

cout << p->data << " ";

p = p->prev;

}

cout << endl;

};

Стек через STL

1. Определить какие предстоит выполнить действия:

* Создать стек через STL.

stack <double> Stack;

* Создать стек в котором буду хранится элементы стека без элементов с четными индексами.

stack <double> StackP;

2. С какими типами данных действие надо сделать, в каком виде эти данные будут представлены:

* Для запоминания длины стека используется переменная size типа int.

int size=-1;

* Для инициализации стека используется переменная Stack.

stack <double> Stack;

* Для инициализации стека в котором будут хранится элементы стека без элементов с четными индексами.

stack <double> StackP;

* Для функции PrintStack используем следующие элементы.

1. Стек

Сама функция имеет тип void так, как входе её работы функция не возвращает значений.

void PrintStack(stack <double> Stack)

3. Для решения задачи данные были представлены в виде множества элементов типа double.

4. Реализация операций ввода и вывода данных:

* Ввод данных с консоли реализован при помощи цикла for и оператора cin.

for (int i = 0; i <= N; i++) {

cout << "Введите " << i + 1 << " элемент: ";

cin >> el;

Stack.push(el);

}

* Вывод данных на консоль реализован при помощи цикла for и оператора cout.

for (int i = 1; i <= N; i++) {

cout << Stack.top() << " ";

Stack.pop();

}

5. Поставленные задачи будут решены при помощи следующих действий:

* Программа запрашивает у пользователя размер стека.

while (size < 0) {

if (size < 0) {

cout << "Введите размер стека: ";

cin >> size;

}

}

* При помощи оператора for в стек добавляются элементы, введенные с консоли.

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

cout << "Введите " << i + 1 << " элемент: ";

cin >> el;

Stack.push(el);

}

* При помощи оператора for программа передает элементы с четными индексами из Stack в StackP при помощи методов top, push и удаляет элементы из Stack при помощи метода pop. Программа копирует элементы из StackP в Stack.

int N = Stack.size();

for (int i = 1; i <= N; i++) {

if ((i)% 2 != 0) {

StackP.push(Stack.top());

}

Stack.pop();

}

N = StackP.size();

for (int i = 1; i <= N; i++) {

Stack.push(StackP.top());

StackP.pop();

}

* При помощи функции PrintStack программа выводит элементы стека на консоль.

void PrintStack(stack <double> Stack) {

int N = Stack.size();

for (int i = 1; i <= N; i++) {

cout << Stack.top() << " ";

Stack.pop();

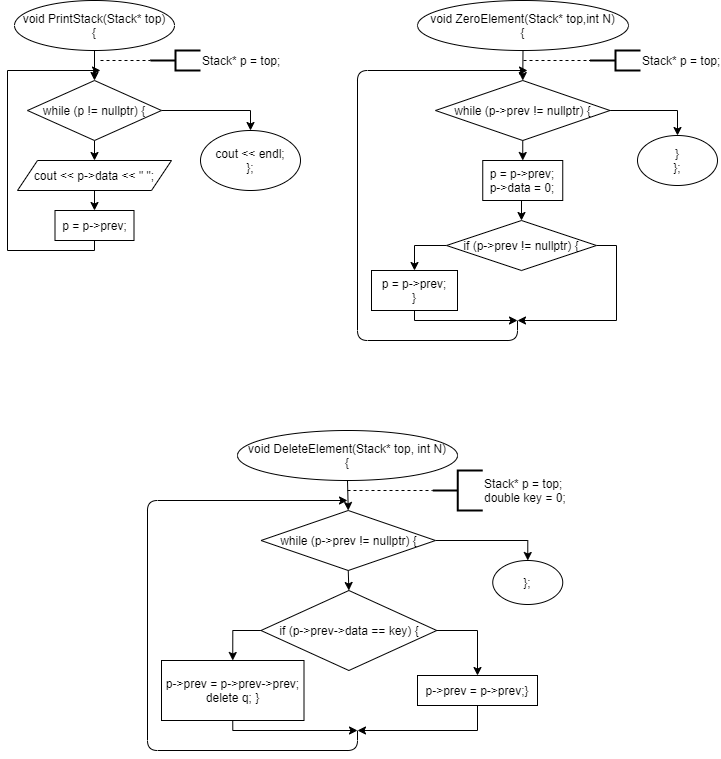
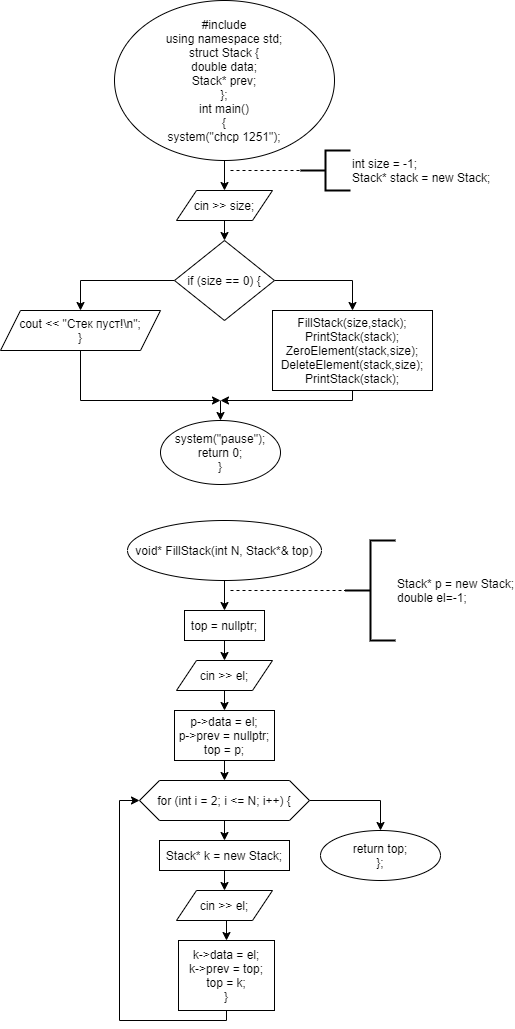
}

cout << endl;

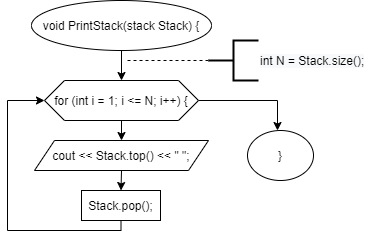
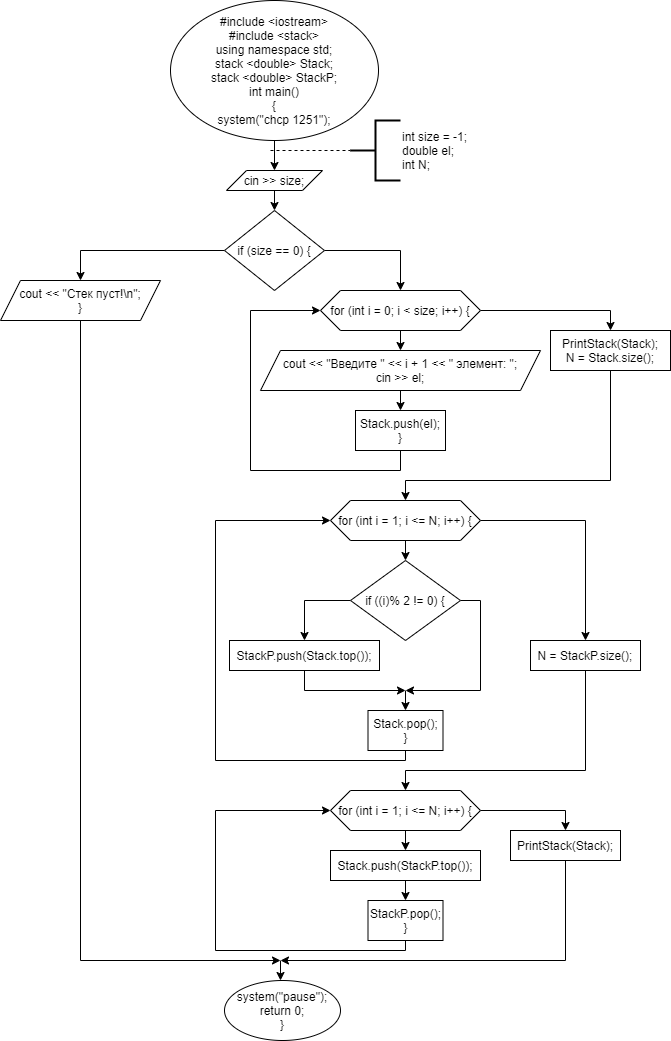
};

# **Блок схема**

Стек



Стек через STL



# **Код программы на языке C++**

Стек

#include<iostream>

using namespace std;

struct Stack {

double data;

Stack\* prev;

};

void\* FillStack(int N, Stack\*& top)

{

Stack\* p = new Stack;

double el=-1;

top = nullptr;

cout << "\nВведите 1 элемент: ";

do {

cin >> el;

if (el==0) {

cout << "Введите 1 элемент: ";

}

} while (el == 0);

p->data = el;

p->prev = nullptr;

top = p;

for (int i = 2; i <= N; i++) {

Stack\* k = new Stack;

cout << "Введите " << i << " элемент: ";

do {

cin >> el;

if (el == 0)

{

cout << "Введите " << i << " элемент: ";

}

} while (el == 0);

k->data = el;

k->prev = top;

top = k;

}

return top;

};

void PrintStack(Stack\* top)

{

Stack\* p = top;

while (p != nullptr) {

if (p->data != 0) {

cout << p->data << " ";

p = p->prev;

}

else {

p = p->prev;

}

}

cout << endl;

};

void ZeroElement(Stack\* top,int N) {

Stack\* p = top;

if (N % 2 != 0) {

while (p->prev != nullptr) {

p = p->prev;

p->data = 0;

if (p->prev != nullptr) {

p = p->prev;

}

}

}

else {

while (p->prev != nullptr) {

p->data = 0;

p = p->prev;

if (p->prev != nullptr) {

p = p->prev;

}

}

}

};

void DeleteElement(Stack\* top, int N)

{

Stack\* p = top;

double key = 0;

while (p->prev != nullptr) {

if (p->prev->data == key) {

Stack\* q = p->prev;

p->prev = p->prev->prev;

delete q;

}

else {

p = p->prev;

}

}

};

int main()

{

system("chcp 1251");

int size = -1;

Stack\* stack = new Stack;

cout << "Введите размер стека: ";

while (size < 0) {

cin >> size;

if (size < 0) {

cout << "Введите размер стека: ";

}

}

if (size == 0) {

cout << "\nСтек пуст!\n";

}

else {

FillStack(size,stack);

cout << "\nСтек: \n";

PrintStack(stack);

ZeroElement(stack,size);

DeleteElement(stack,size);

cout << "\nСтек после удаления: \n";

PrintStack(stack);

}

system("pause");

return 0;

}

Стек через STL

#include <iostream>

#include <stack>

using namespace std;

stack <double> Stack;

stack <double> StackP;

void PrintStack(stack <double> Stack) {

int N = Stack.size();

for (int i = 1; i <= N; i++) {

cout << Stack.top() << " ";

Stack.pop();

}

cout << endl;

};

int main()

{

system("chcp 1251");

int size = -1;

double el;

while (size < 0) {

if (size < 0) {

cout << "Введите размер стека: ";

}

cin >> size;

}

cout << endl;

if (size == 0) {

cout << "Стек пуст!\n";

}

else {

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

cout << "Введите " << i + 1 << " элемент: ";

cin >> el;

Stack.push(el);

}

cout << "\nСтек:\n";

PrintStack(Stack);

int N = Stack.size();

for (int i = 1; i <= N; i++) {

if ((i)% 2 != 0) {

StackP.push(Stack.top());

}

Stack.pop();

}

N = StackP.size();

for (int i = 1; i <= N; i++) {

Stack.push(StackP.top());

StackP.pop();

}

cout << "\nСтек:\n";

PrintStack(Stack);

}

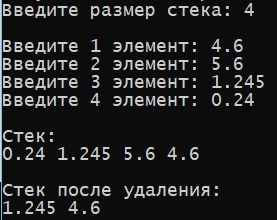
system("pause");

return 0;

}

# **Скриншоты тестов**

Стек



Стек через STL

