Пермский Национальный Исследовательский Политехнический Университет

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

Лабораторная работа № 11.3

по дисциплине

«Основы алгоритмизации и программирования»

Тема: «Динамические структуры данных. Стеки»

Вариант № 17

Выполнил:

студент группы РИС-20-2б

Пономарев Егор Витальевич

Проверила:

доцент кафедры ИТАС

Полякова О. А.

Пермь, 2021

**Цель работы**

Получить практические навыки работы с очередями.

**Постановка задачи**

* Сформировать стеки с помощью структуры и библиотеки STL.
* Распечатать полученную структуру.
* Выполнить обработку структуры в соответствии с заданием.
* Распечатать полученный результат.
* Удалить соответствующую структуру из памяти.

Задание варианта:

1. Тип информационного поля int.
2. Удалить из списка все элементы с четными информационными полями.

**Анализ задачи**

**Реализация через STL:**

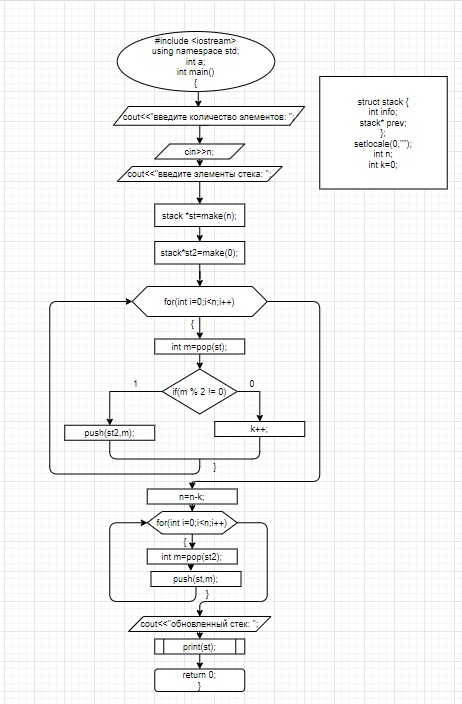
1. Так как нужно реализовать программу через STL – библиотеку, нужно подключить библиотеку <stack> .
2. Из поставленной задачи можно сделать вывод, что нужно выводить стек несколько раз, поэтому, чтобы не писать несколько раз одно и тоже, создадим функцию (void print(stack<char> st)) для вывода элементов стека. Введём переменную (int p), которая будет считать размер стека с помощью (size). С помощью цикла (for (int i = 0; i < p; i++)) должны выводиться элементы стека. Чтобы выводились все элементы, надо после вывода вершины стека удалить элемент из стека.
3. Записи в стеке содержат ключевое поле типа int (строка символов).
4. Сформировать стек:
5. Так как нужно записывать элементы в стек, надо его создать (stack<char> st;). После чего, так как не сказано сколько элементов содержит стек, пользователь должен ввести (int n-количество элементов в стеке). Затем с помощью цикла (for (int i = 0; i < n; i++)) введем элемент стека в переменную (int a) и добавим его в стек. После чего с помощью ранее написанной функции (print) выведем элементы стека на экран.
6. Удалить элементы с четным значением:
7. Чтобы выполнить поставленную задачу создадим второй стек, в который будут переноситься нужные элементы главного стека (stack<int> st2;).. После чего с помощью цикла (for (int i = 0; i < n; i++)) и оператора выбора (if (st.top() % 2 != 0)) перенесём все нужные элементы во второй стек (st2.push(st.top());). Для того, чтобы определить количество элементов после удаления воспользуемся (size) (n = st2.size();). Затем с помощью цикла (for (int i = 0; i < n; i++) ) перенесём элементы из второго стека в исходный.

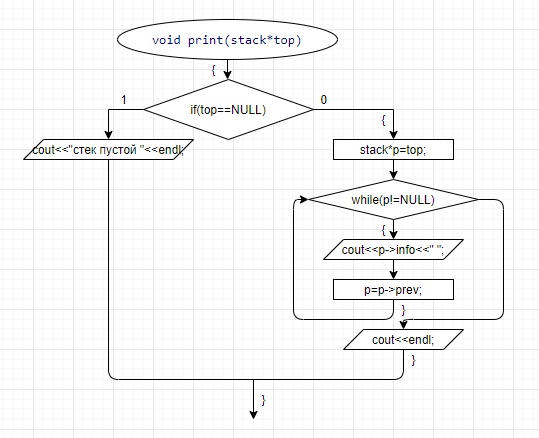
**Реализация через структуру:**

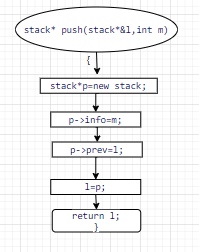
1. Так как нужно реализовать программу через структуры, нужно создать структуру, которая будет содержать поля (int data;) для данных и (Stack\* prev;) для адреса следующего элемента.
2. Создам функцию (Stack\* make\_stack (int n)) для создания стека и его заполнения. Выделим память под 1 элемент (p = new Stack;), пользователь записывает данные (cin >> a; p->data = (a);) и адрес на NULL (p->prev = NULL; ) в элемент. Затем ставим указатель на верхний элемент стека (top = p;).
3. После этого с помощью цикла (for (int i = 2; i <= n; i++)) введём оставшиеся элементы. В конце возвращаем указатель на верхний элемент.
4. Создадим функцию (void print\_stack(Stack\* top)) для вывода элементов стека. В начале проверим с помощью оператора выбора (if) указывает ли верхний элемент стека (top) на NULL. Если да, то выводим на экран сообщение “стек пуст”. Иначе с помощью (while) выводим элемент и переходим к следующему пока указатель не будет равен NULL.
5. Создадим функцию (int pop(Stack\*& top)) для возвращения первого элемента и его удаления. Для начала считаем количество элементов в стеке (k) с помощью (while). Потом с помощью (if) проверяем равно ли (k) единице. Если да, то обнуляем указатель и возвращаем элемент. Иначе сохраняем значение последнего элемента, делаем второй элемент первым, удаляем последний элемент и возвращаем первый элемент.
6. Создадим функцию (Stack\* push(Stack\*& top, int m)) для добавления элемента в стек. Сначала выделим память под новый элемент. Потом присваиваем значение, которое вводит пользователь, для нового элемента, делаем указатель на нижний элемент и делаем новый элемент первым элементом стека.
7. Записи в стеке содержат ключевое поле типа char(строка символов).
8. Сформировать стек:
9. Так как не сказано сколько элементов содержит стек, то пользователь должен ввести количество (n). Потом вызвать ранее написанные функцию для создания стека и его заполнения (Stack\* st = make\_stack(n);) и функцию для вывода стека (print\_stack(st);).
10. Удалить элемент с четным значением:
11. Чтобы выполнить поставленную задачу создадим второй стек, в который будут переноситься нужные элементы главного стека (Stack\* st2 = NULL). Потом, так как ключ не указан, пользователь должен ввести ключ (b) для удаления. После чего с помощью цикла (for (int i = 0; i < n; i++)) и оператора выбора (if) перенесём все нужные элементы во второй стек и посчитаем количество четных элементов (k). Для того, чтобы определить количество элементов после удаления вычтем из n-k. Затем с помощью цикла (for (int i = 0; i < n; i++) ) перенесём элементы из второго стека в исходный.

**Блок-схема**

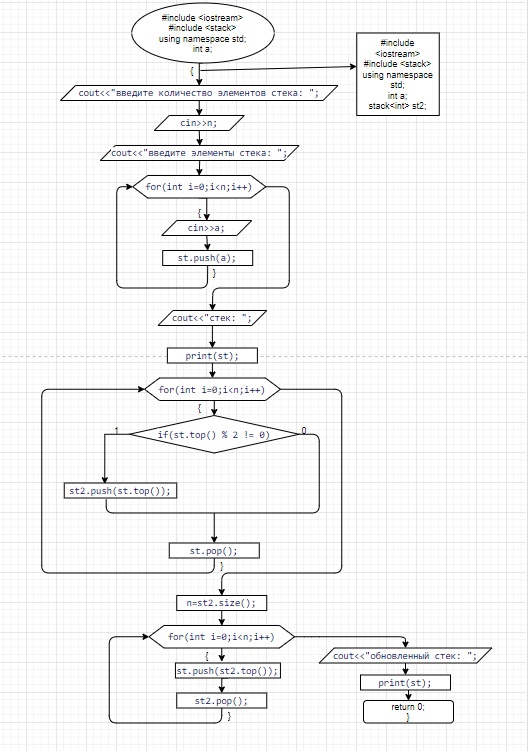
**Структура:**

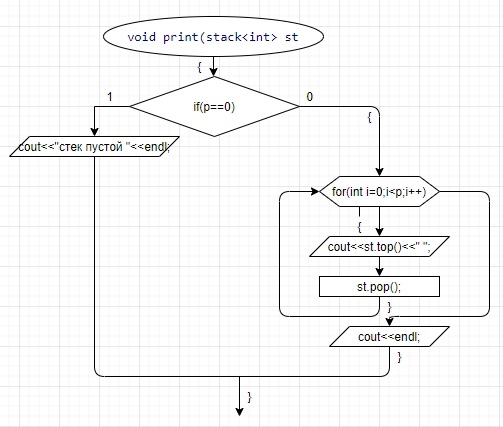






**STL:**





**Код С++**

**Структура:**

#include <iostream>

using namespace std;

int a;

struct Stack {

int data;

Stack\* prev;

};

Stack\* make\_stack(int n) {

Stack\* top, \* p;

top = NULL;

p = new Stack;

cin >> a;

p->data = a;

p->prev = NULL;

top = p;

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

Stack\* h = new Stack;

cin >> a;

h->data = a;

h->prev = top;

top = h;

}

return top;

}

void print(Stack\* top) {

if (top == NULL)

cout << "Стек пуст.\n";

else {

Stack\* p = top;

while (p != NULL) {

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

p = p->prev;

}

cout << endl;

}

}

int pop(Stack\*& top) {

Stack\* p = top;

int k = 0;

while (p != NULL) {

k++;

p = p->prev;

}

p = top;

if (k == 1) {

int t = top->data;

delete p;

top = NULL;

return t;

}

else {

Stack\* t = p->prev;

int u = p->data;

top = t;

delete p;

return u;

}

}

Stack\* push(Stack\*& top, int m) {

Stack\* p = new Stack;

p->data = m;

p->prev = top;

top = p;

return top;

}

int main()

{

setlocale(LC\_ALL, "rus");

cout << "Введите количество элементов: ";

int n, e = 0;

cin >> n;

cout << "Введите элементы стека:\n";

Stack\* st = make\_stack(n);

cout << "Стек:\n";

print(st);

Stack\* st2 = NULL;

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

int m = pop(st);

if (m % 2 != 0)

push(st2, m);

else

e = e + 1;

}

n = n - e;

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

int m = pop(st2);

push(st, m);

}

cout << "Обновленный стек:\n";

print(st);

return 0;

}

**STL:**

#include <iostream>

#include <stack>

using namespace std;

int a;

void print(stack<int> st)

{

int p = st.size();

if (p == 0)

{

cout << "стек пустой";

}

else {

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

{

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

st.pop();

}

}

cout << endl;

}

int main()

{

setlocale(0, "rus");

stack<int> st;

int n;

cout << "введите количество элементов стека: ";

cin >> n;

cout << "введите элементы стека: ";

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

{

cin >> a;

st.push(a);

}

cout << "стек: ";

print(st);

stack<int> st2;

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

{

if (st.top() % 2 != 0)

{

st2.push(st.top());

}

st.pop();

}

n = st2.size();

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

{

st.push(st2.top());

st2.pop();

}

cout << "обновленный стек: ";

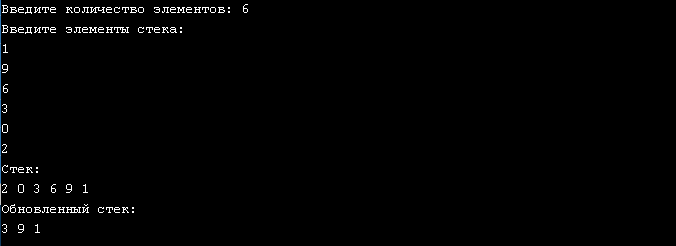
print(st);

return 0;

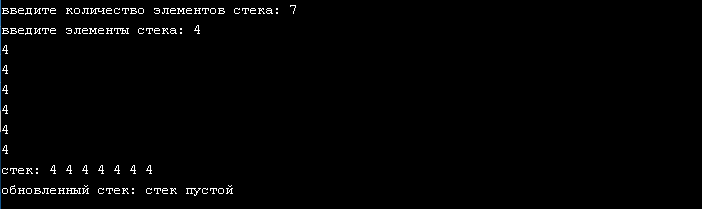
}

**Результаты выполнения**

**Структура:**

****

**STL:**

****