МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ

Кафедра системного проектування

**ЗВІТ**про виконання лабораторної роботи №3

«Дослідження структур даних стек та черга»

з дисципліни «Алгоритми та структури даних»

Виконав:

cтудент І курсу

групи ДА-02

Гринчишин Д.Б.

Перевірили:

..

Київ – 2020

**Мета**. Ознайомитись і дослідити принципи FIFO, LIFO та лінійні структури даних: стек та чергу. Реалізувати їх на основі статичних масивів, набути навичок роботи зі структурами та вказівниками на мові програмування С/C++.

**Варіант завдань - 2**

**Завдання 1.**

Структура даних Подія редагування документу має наступні поля: текст, номер рядка. Виконати наступні завдання:

1) Створити стек історії редагування документу, розмір історії 7.

2) Додати в неї 4 різні події.

3) Додати в неї ще 5 подій.

4) Зробити “undo” останніх двох подій.

5) Зробити “undo” останньої події та додати її ще раз вже з іншим текстом.

6) Зробити “undo” ще 6 разів.

**Код:**

#include <iostream>

#include <string>

#define N 7

using namespace std;

struct change

{

string str;

int row;

};

struct stack

{

change\* changes[N];

};

int size(stack\* hist)

{

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

{

if (!hist->changes[i]) return i;

}

return N;

}

void push(stack \*hist, change \*a)

{

int l = size(hist);

if (l == N) { cout << "push: error, stack is full" << endl; }

else { hist->changes[l] = a; }

}

bool empty(stack\* hist) { return size(hist) == 0; }

change \*top(stack \*hist)

{

if (!empty(hist)) return hist->changes[size(hist) - 1];

cout << "top: error, stack is empty" << endl;

return 0;

}

void pop(stack \*hist)

{

if (!empty(hist)) hist->changes[size(hist) - 1] = 0;

else cout << "pop: error, stack is empty" << endl;

}

void print(stack \*hist)

{

if (!empty(hist))

{

change\* save[N];

int l = size(hist);

for (int i = N - 1; i >= 0; i--)

{

if (i >= l) save[i] = 0;

else

{

save[i] = top(hist);

cout << "change #" << i << " text: " << save[i]->str << " row: " << save[i]->row << endl;

pop(hist);

}

}

for (int i = 0; i < N; i++) push(hist, save[i]);

}

else cout << "print: error, stack is empty" << endl;

}

int main()

{

change a = { "change 1", rand() % 20 };

change b = { "change 2", rand() % 20 };

change c = { "change 3", rand() % 20 };

change d = { "change 4", rand() % 20 };

change e = { "change 5", rand() % 20 };

change f = { "change 6", rand() % 20 };

change g = { "change 7", rand() % 20 };

//add 4

stack hist = {&a, &b, &c, &d};

//add 5

push(&hist, &e);

push(&hist, &f);

push(&hist, &g);

push(&hist, &a);

push(&hist, &b);

//undo

pop(&hist);

pop(&hist);

//undo and add

change\* last = top(&hist);

pop(&hist);

last->str = "new text";

push(&hist, last);

//undo 6 times

for (int i = 0; i < 6; i++) pop(&hist);

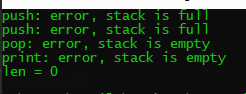
//print

print(&hist);

cout << "len = " << size(&hist) << endl;

return 1;}

**Результат виконання програми:**



**Завдання 2.**

Структура даних **Робітник** має наступні поля: посада, стаж роботи, заробітна плата. Виконати наступні завдання:

1) Створити чергу робітників на підвищення зарплати, розмір черги 8.

2) Додати в неї 5 різних робітників.

3) Взяти першого робітника, підвищити йому зарплату на 200 та знову додати в чергу.

4) Додати ще 4 робітника.

5) Забрати 3 робітників з черги.

6) Забрати ще 7 робітників з черги.

**Код:**

#include <iostream>

#include <string>

#define N 8

using namespace std;

struct worker

{

string position;

int experience;

int salary;

};

struct queue

{

worker\* list[N];

};

int size(queue\* workers)

{

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

{

if (!workers->list[i]) return i;

}

return N;

}

void push(queue\* workers, worker\* a)

{

int l = size(workers);

if (l < N)

{

for (int i = N - 1; i > 0; i--)

{

workers->list[i] = workers->list[i - 1];

}

workers->list[0] = a;

}

else cout << "push : error, queue is full" << endl;

}

bool empty(queue\* workers) { return size(workers) == 0; }

worker\* front(queue\* workers)

{

if (!empty(workers)) return workers->list[size(workers) - 1];

cout << "front: error, queue is empty" << endl;

return 0;

}

void pop(queue\* workers)

{

if (!empty(workers)) workers->list[size(workers) - 1] = 0;

else cout << "pop: error, queue is empty" << endl;

}

void print(queue\* workers)

{

if (!empty(workers))

{

worker\* save[N];

int l = size(workers);

for (int i = N - 1; i >= 0; i--)

{

if (i < l)

{

save[i] = front(workers);

cout << "worker #" << l - i << ", position: " << save[i]->position << ", experience: " << save[i]->experience << ", salary: " << save[i]->salary << endl;

pop(workers);

}

}

for (int i = l - 1; i >= 0; i--)

{

push(workers, save[i]);

}

}

else cout << "print: error, queue is empty" << endl;

}

int main()

{

worker a = { "worker 1", rand() % 20, rand() % 20 + 100 };

worker b = { "worker 2", rand() % 20, rand() % 20 + 100 };

worker c = { "worker 3", rand() % 20, rand() % 20 + 100 };

worker d = { "worker 4", rand() % 20, rand() % 20 + 100 };

worker e = { "worker 5", rand() % 20, rand() % 20 + 100 };

worker f = { "worker 6", rand() % 20, rand() % 20 + 100 };

worker g = { "worker 7", rand() % 20, rand() % 20 + 100 };

//5 workers

queue workers = { &a, &b, &c, &d, &e };

//get increase 200 and add

worker\* temp = front(&workers);

pop(&workers);

temp->salary += 200;

push(&workers, temp);

//4 more

push(&workers, &g);

push(&workers, &f);

push(&workers, &e);

push(&workers, &d);

//dequeue 3

for(int i =0;i<3;i++) pop(&workers);

//dequeue 7 more

for (int i = 0; i < 7; i++) pop(&workers);

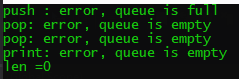
//print

print(&workers);

cout << "len =" << size(&workers) << endl;

return 1;

}

**Результат виконання програми:  
**

**Висновок:**

Я здобув конкретні навички та знання в написанні коду на мові програмування C++, а саме набув навичок в реалізації таких та використання таких структур даних, як стек та черга. Також я отримав навички використання вказівників та передавання їх у ункцію в якості аргумента. Упродовж виконання цієї лабораторної роботи було реалізовано такі класичні функції для роботи з стеком, як size, top, empty, push, pop та print та функції для роботи з чергою - size, front, empty, push, pop та print.