

Алгоритми та структури даних

**Лабораторна робота №4**

**“Дослідження структур даних стек та черга”**

Робота : Лещинського Б.Д. группа КА-07

08.12.2020

Варіант 14

**Мета роботи** :

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++.

**Хід виконання роботи:**

**Завдання:**

Написати програми для вирішення двох задач свого варіанту: першої з використанням структури даних стек, другої з використанням черги. Алгоритм реалізації кожної програми:

1. Створити структуру (struct) для зберігання даних по задачі свого варіанту.

2. Створити структуру (struct) Стек / Черга, вона буде містити в собі масив структур з даними. ! реалізувати стек / чергу на основі статичного масиву (не однозв’язного списку), тобто стек / черга фіксованої довжини N (#define N) ! використовувати тільки структури (struct), заборонено використання глобальних змінних, ООП, класів

3. Реалізувати основні функції стеку / черги:  push() – додає елемент в стек / чергу  pop() – видаляє елемент зі стеку / черги  empty() – перевіряє стек / чергу на пустоту  size() – повертає розмір стеку / черги  \*тільки для стеку: top() – повертає верхній елемент стеку  \*тільки для черги: front() – повертає перший елемент черги У функціях додати захист від помилок, тобто перевірки на переповнення або зчитування з пустого стеку / черги, при помилці виводити попередження.

4. Реалізувати функцію print() для виведення стеку / черги з даними, що зберігаються, використовуючи тільки створені вище функції, без прямого доступу до масиву.

5. Ініціалізувати стек / чергу та виконати поставлені завдання по варіанту, після кожного завдання виводити розмір та вміст стеку / черги.

Варіант 14

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.

2. Структура даних Пісня має наступні поля: назва, тривалість у секундах, кількість лайків. Виконати наступні завдання:

1) Створити плейлист (чергу) пісень, розмір плейлисту 4.

2) Додати в нього 2 різні пісні.

3) Додати ще 4 пісні.

4)Взяти перші дві пісні та знову додати їх у плейлист вже в зворотьому порядку.

5) Взяти першу пісню, збільшити її кількість лайків на 100 та повернути у плейлист.

6) Забрати 5 пісень з плейлисту.

**Код програми(для структури стек)**:

#include<iostream>

#include<cstdlib>

#include<string>

#define N 7

using namespace std;

struct docChange{

int lineNum;

string text;

};

struct Stack{

docChange changeHistory[N];

int top = -1;

};

void push(Stack \*stack, docChange \*change){

if(stack->top+1 >= N){

cout<<"Access denied: stack overflow!\n";

return;

}

stack->top++;

stack->changeHistory[stack->top] = \*change;

}

bool empty(Stack \*stack){

return stack->top==-1;

}

void pop(Stack \*stack){

if(stack->top == -1){

cout<<"Access denied: stack underflow!\n";

return;

}

stack->top--;

}

int size(Stack \*stack){

return stack->top+1;

}

docChange top(Stack \*stack){

return stack->changeHistory[stack->top];

}

void print(Stack \*stack){

int n = size(stack);

docChange restore[n];

int i = 0;

docChange topEl;

while(empty(stack)==0){;

topEl = top(stack);

cout<<topEl.lineNum<<" "<<topEl.text<<"\n";

restore[n - 1 - i] = top(stack);

pop(stack);

i++;

}

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

push(stack,&restore[i]);

}

}

int main(){

docChange change;

Stack stack;

for(int i = 0; i<4; i++){

cout<<"Enter some changes into document(first line number then text changes): \n";

cin>> change.lineNum;

cin>> change.text;

push(&stack,&change);

}

cout<<"The current stack is: \n";

print(&stack);

cout<<endl;

for(int i = 0; i<5; i++){

cout<<"Enter some changes into document(first line number then text changes): \n";

cin>> change.lineNum;

cin>> change.text;

push(&stack,&change);

}

cout<<"The current stack is: \n";

print(&stack);

cout<<endl;

pop(&stack);

pop(&stack);

cout<<"The current stack is: \n";

print(&stack);

cout<<endl;

pop(&stack);

cout<<"The current stack is: \n";

print(&stack);

cout<<endl;

cout<<"Enter some changes into document(first line number then text changes): \n";

cin>> change.lineNum;

cin>> change.text;

push(&stack,&change);

cout<<"The current stack is: \n";

print(&stack);

cout<<endl;

pop(&stack);

pop(&stack);

pop(&stack);

pop(&stack);

pop(&stack);

pop(&stack);

cout<<"The current stack is: \n";

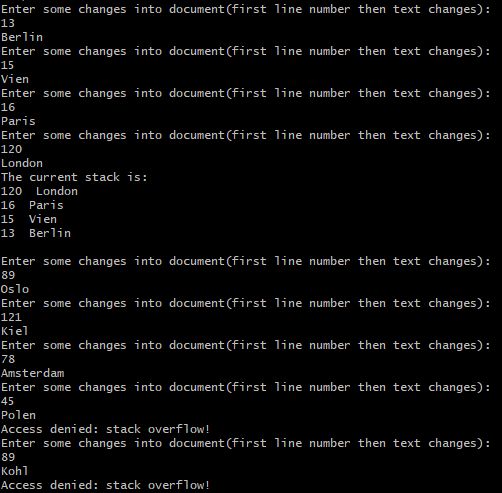
print(&stack);

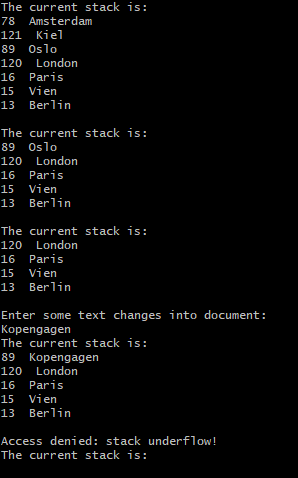
cout<<endl;

return 0;

}

**Вивід програми**:





**Код програми(для структури черга)**:

#include<iostream>

#include<cstdlib>

#include<string>

#define N 4

using namespace std;

struct Song{

string title;

int secDuration;

int likesNum;

};

struct Queue{

Song playlist[N];

int head = 0;

int tail = 0;

int capacity = 0;

};

bool empty(Queue \*queue){

return queue->capacity==0;

}

int size(Queue \*queue){

return queue->capacity;

}

void pop(Queue \*queue){

if(empty(queue)){

cout<<"Access denied: queue underflow"<<endl;

}

else{

queue->head = (queue->head + 1)%N;

queue->capacity--;

}

}

void push(Queue \*queue, Song \*song){

if(size(queue) != N ){

queue->playlist[queue->tail] = \*song;

queue->tail = (queue->tail + 1)%N;

queue->capacity++;

}

else{

cout<<"Access denied: queue overflow"<<endl;

}

}

Song front(Queue \*queue){

return queue->playlist[queue->head];

}

void print(Queue \*queue){

int n = size(queue);

Song restore[n];

Song topEl;

int i = 0;

while(empty(queue)!=1){

topEl = front(queue);

cout<<topEl.title<<" "<<topEl.secDuration<<" "<<topEl.likesNum<<endl;

restore[i] = front(queue);

pop(queue);

i++;

}

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

push(queue,&restore[i]);

}

}

int main(){

Queue playlist;

Song song;

Song shuffle1;

Song shuffle2;

for(int i=0; i<2; i++){

cout<<"Add some song into your playlist(title, duration, likes respectively):\n";

cin>>song.title;

cin>>song.secDuration;

cin>>song.likesNum;

push(&playlist,&song);

}

cout<<"The current playlist is: \n";

print(&playlist);

cout<<endl;

for(int i = 0; i<4; i++){

cout<<"Add some song into your playlist(title, duration, likes respectively):\n";

cin>>song.title;

cin>>song.secDuration;

cin>>song.likesNum;

push(&playlist,&song);

}

cout<<"The current playlist is: \n";

print(&playlist);

cout<<endl;

shuffle1 = front(&playlist);

pop(&playlist);

shuffle2 = front(&playlist);

pop(&playlist);

push(&playlist,&shuffle2);

push(&playlist,&shuffle1);

cout<<"The current playlist is: \n";

print(&playlist);

cout<<endl;

shuffle1 = front(&playlist);

shuffle1.likesNum += 100;

pop(&playlist);

push(&playlist,&shuffle1);;

cout<<"The current playlist is: \n";

print(&playlist);

cout<<endl;

pop(&playlist);

pop(&playlist);

pop(&playlist);

pop(&playlist);

pop(&playlist);

cout<<"The current playlist is: \n";

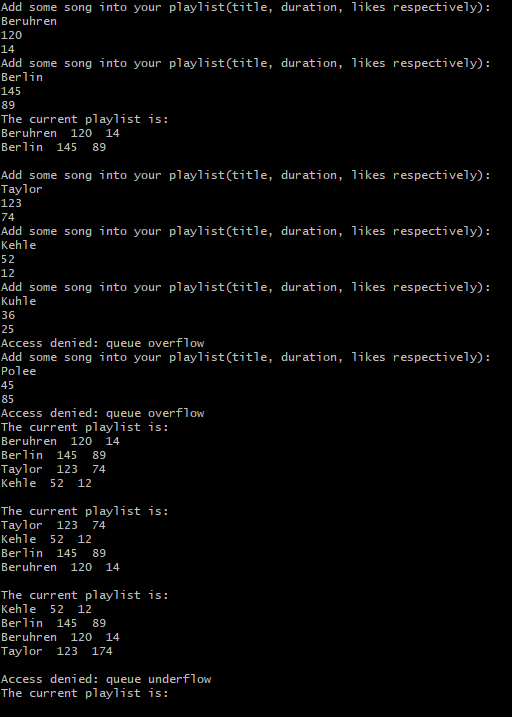
print(&playlist);

cout<<endl;

return 0;

}

**Вивід програми**:



**Висновок:**

Я ознайомився і дослідив принципи FIFO, LIFO та лінійні структури даних: стек та чергу. Реалізував їх на основі статичних масивів, набувши навичок роботи зі структурами та вказівниками на мові програмування C++.