## ЗВІТ З ЛАБОРАТОРНОЇ РОБОТИ №2

# Моделювання роботи стеку та черги.

## з курсу «Алгоритми і структури даних»

## студента групи ПА-19-2

## Філіпенка Дмитра Миколайовича

## кафедра комп’ютерних технологій, ДНУ

## 2019/2020 н.р.

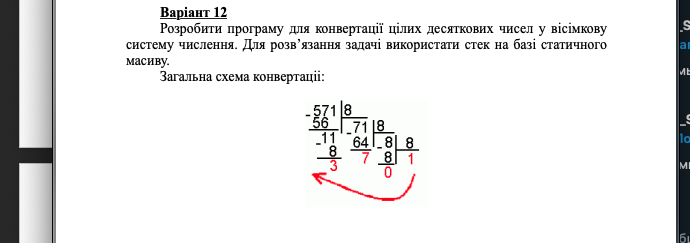
1. Постановка задачі.

**Завдання**. Розробити програму, в якій

* реалізувати абстрактний тип даних, необхідний для виконання індивідуального завдання згідно з варіантом (стек або чергу);
* реалізувати основні операції для розробленої структури;
* реалізувати індивідуальне завдання.

Обов’язкові вимоги:

* після завантаження програма пропонує користувачу меню для вибору дії з такими пунктами:
  + Додати елемент до структури даних.
  + Видалити елемент із структури даних.
  + Вивести елементи структури на екран.
  + Очистити структуру.
  + Вивести кількість елементів структури, індекс першого та останнього елементу.
  + *Пункти щодо індивідуальних завдань згідно з варіантом*;
  + Завершення роботи.
* Організувати роботу програми таким чином, щоб перелічені вище дії можна було б виконувати неодноразово за один сеанс роботи програми.
* При видаленні елементів передбачити обробку помилкових ситуацій, коли будуть виконуватися спроби видалення елемента із порожньої структури.
* Структуру даних реалізувати на базі статичного масиву із *n* елементів.
* Операції над структурою даних реалізувати у вигляді окремих функцій.
* Індивідуальні завдання за варіантом – окремі функції.
* Для виконання операцій над елементами структури використовувати розроблені операції-функції, які відповідають порядку роботи даної структури (наприклад, неможна видаляти елемент з середини стеку або черги).



2. Опис розв’язку. За необхідністю навести блок-схему алгоритму.

Для цього завдання я реалізував 3 змінні та 3 функції

Size – макс розмір масиву,

Data – сам масив,

topS – індекс останнього (верхнього елемента),

DelStack() – очистка стеку,

Push() – вибір самого верхнього елемента,

Pop() – додання нового елемента

Для кожного завдання реалізована функція   
AddSD, ShowSD, DeleteSd, ClearSD, InfoSD, Task1

В усі ці функції передаємо аргументи, які необхідні для завдання де потім виконуємо операції з його основними функціями POP PUSH

Для того щоб змінювати значення змінних прямо в функція наприклад змінну topS,

Потрібно передавати її по посиланню на цю зміну

При невдалих push, pop є відповідні помилки OVERFLOW, UNDERFLOW.

3.Вихідний текст програми (з необхідними коментарями).

//

// main.cpp

// algoritms-lb2

//

// Created by Дима Филипенко on 08.03.2020.

// Copyright © 2020 Дима Филипенко. All rights reserved.

//

#include <iostream>

**using** **namespace** std;

**bool** cinFail = **false**;

**void** wait\_key(){

**do**

{

cout << '\n' << "Press a key to continue...";

} **while** (cin.get() != '\n');

getchar();

// clrscr();

// ClearScreen();

}

**bool** Push(**int** x,**int** & topS, **int** size,**int** \* data){

**if** (topS == size){

**return** **false**;

}

topS++;

data[topS-1]=x;

**return** **true**;

}

**int** Pop(**int** &topS, **int** size,**int** \* data){

**if** (topS <= 0){

**return** **false**;

}

topS--;

//cout<<this->topS<<" new top s"<<endl;

**return** data[topS];

}

**bool** AddSD(**int** &topS, **int** size,**int** \* data){

**int** x;

cout<<"PUSH x=";

**if**(!(cin>>x)){

cout << "Некоректний ввід";

cin.clear();

cin.ignore(numeric\_limits<streamsize>::max(), '\n');

**return** **false**;

}**else**{

cout<<endl;

**if**(!Push(x,topS,size,data)){

cout<<"STACK OVERFLOW";

}**else**{

cout<<"Елмент додано успішно";

}

cout<<endl;

**return** **true**;

}

}

**void** ShowSD(**int** &topS,**int** \* data){

cout<<endl;

**if** (topS>0){

**for** (**int** i=topS-1; i>=0; i--) {

cout<<" "<<data[i];

}

}**else**{

cout<<"Stack is empty";

}

cout<<endl;

}

**void** DeleteSd(**int** &topS, **int** size,**int** \* data){

**int** pop = Pop(topS,size,data);

**if**(!pop){

cout<<endl;

cout<<"STACK UNDERFLOW";

cout<<endl;

}**else**{

cout<<endl;

cout<<"Елемент "<<pop<<" видалено успішно ";

cout<<endl;

}

}

**void** InfoSD(**int** topS){

cout<<"Кількість елементів - "<<topS<<endl;

**if** (topS>0){

cout<<"Індекс першого - "<<0<<endl;

cout<<"Індекс останнього - "<<topS-1<<endl;

cout<<endl;

}

}

**void** StackDel(**int** &topS,**int** size,**int** \* data)

{

**while** (topS>0) {

Pop(topS,size, data);

}

// this->topS=0;

// this->data = new int[this->size];

}

**bool** Task1(**int** &topS,**int** size,**int** \* data){

StackDel( topS, size, data);

**int** n,remainder;

cin>>n;

**if**(cin.fail()){

cinFail=**true**;

**return** **false**;

}

**do** {

remainder=n%8;

n=n/8;

**if**(!Push(remainder,topS,size,data)){

cout<<"STACK OVERFLOW";

}

}**while** (n!=0);

**return** **true**;

}

**void** ClearSD(**int** topS,**int** size,**int** \* data){

StackDel( topS, size, data);

cout<<endl;

cout<<"Stack очищено";

cout<<endl;

}

**int** main(**int** argc, **const** **char** \* argv[]) {

**int** size;

**int** \*data;

**int** topS=0;

**int** n;

**do**{

cout<<"введіть розмір стеку"<<endl;

cin>>n;

**if**(cin.fail())

{

**break**;

}**else**{

size=n;

data= **new** **int** [size];

}

}**while**(n<=0);

**int** k;

cout<<"1. Додати елемент до стеку \n";

cout<<"2. Видалити елемент \n";

cout<<"3. Вивести елементи стеку на екран \n";

cout<<"4. Очистити структуру \n";

cout<<"5. Вивести кількість елементів структури, індекс першого та останнього елементу … \n";

cout<<"6. Конвертація цілих десяткових чисел у вісімкову систему числення.\n";

cout<<"7. Вихід \n";

**do**

{

cout<<endl;

cout<<"Очікую # з меню..."<<endl;

cout<<endl;

cin>>k;

**if**(cin.fail() || cinFail == **true** ){

**break**;

}

**switch** (k)

{

**case** 1: //додавання

**if**(AddSD(topS, size, data)){

wait\_key();

**break**;

}**else**{

k=7;

**break**;

}

**case** 2: DeleteSd( topS,size,data); wait\_key(); **break**; // видалення

**case** 3: ShowSD(topS, data); wait\_key(); **break**; // показати

**case** 4: ClearSD(topS,size,data); wait\_key(); **break**; // очистити структуру

**case** 5: InfoSD(topS); wait\_key(); **break**; // кількість перший та останій елемент

**case** 6: // задача варіант 12

**if**(Task1(topS,size,data)){

wait\_key();

**break**;

}**else**{

k=7;

}

**case** 7: **break**; // задача варіант 12

**default**: cout<<"Некоректний ввід"<<endl; **break**;

}

}

**while** (k!=7);

}

4. Опис інтерфейсу (керівництво користувача).

Для початку треба ввести розмір стеку

При запуску програми ви бачите меню з відповідними функціями .

**1. Додати елемент до стеку**

**2. Видалити елемент**

**3. Вивести елементи стеку на екран**

**4. Очистити структуру**

**5. Вивести кількість елементів структури, індекс першого та останнього елементу …**

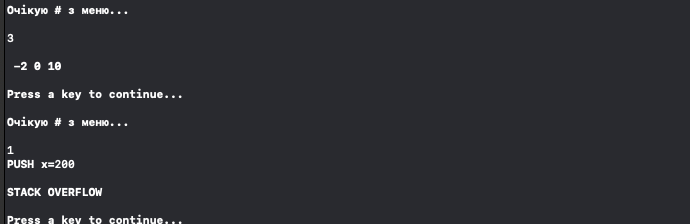
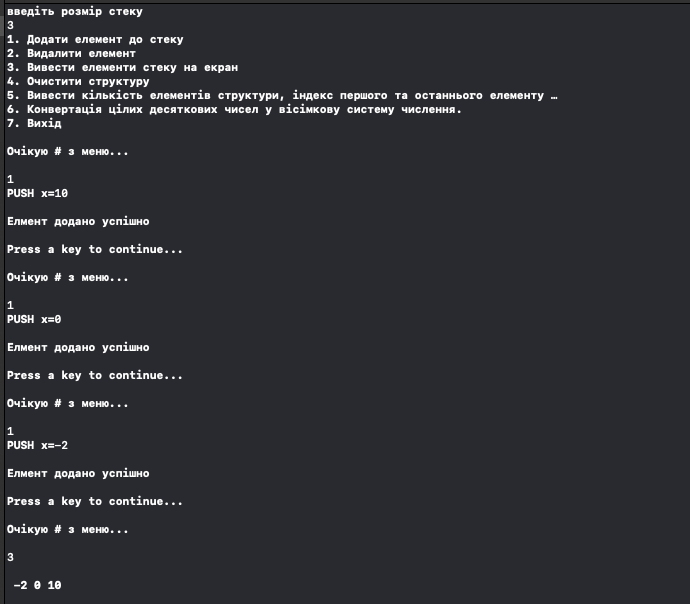
**6. Конвертація цілих десяткових чисел у вісімкову систему числення.**

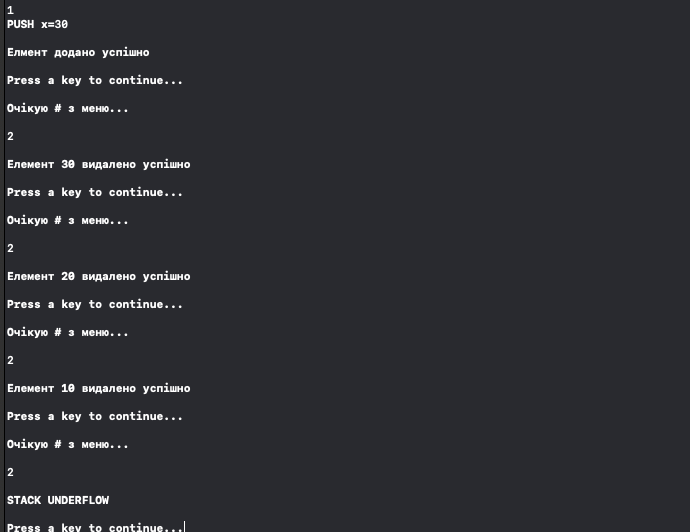
**7. Вихід**

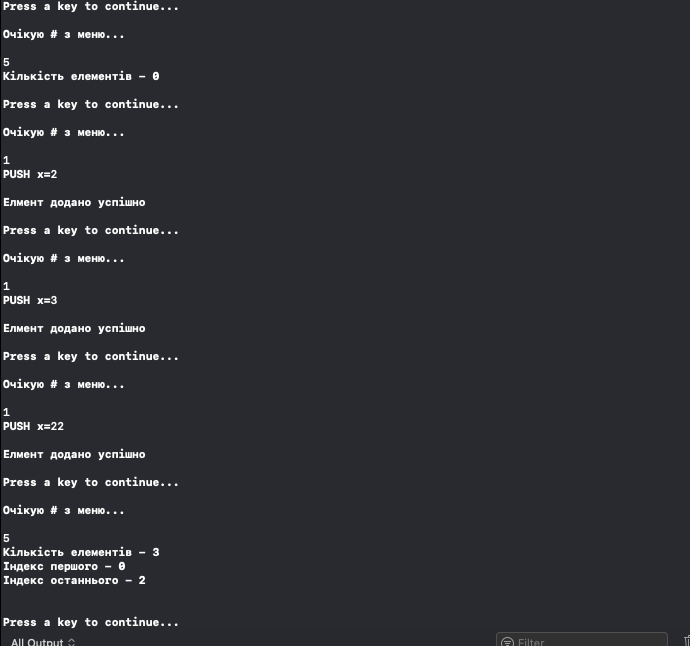
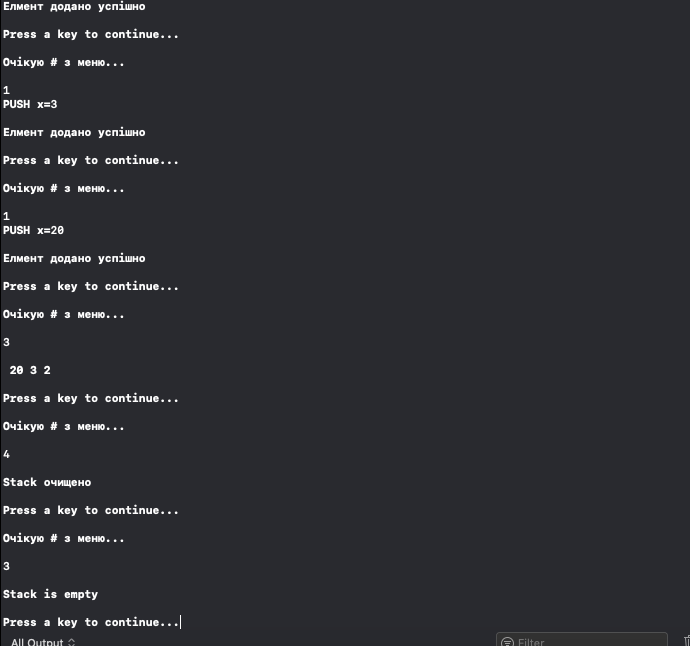
**Якщо натиснули 1 – ви повинні ввести число для додання**

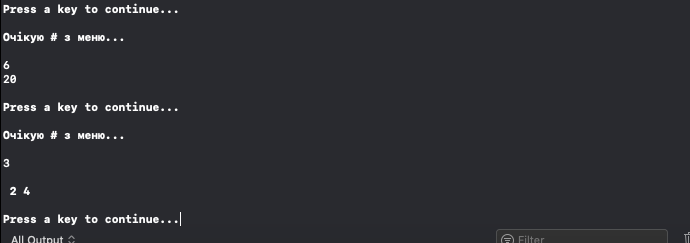
**Якщо 2 – видаляється останній верхній елемент**

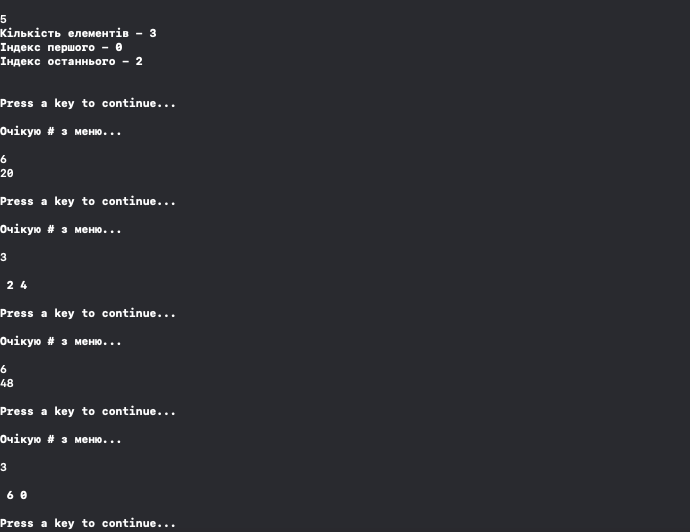
5. Опис тестових прикладів та скриншоти їх реалізації.











6. Аналіз результатів. Висновки.

**push** елемент додається в стек та розміщується в його верхівці. Розмір стека збільшується на одиницю. При перевищенні розміру стека граничної величини, відбувається переповнення

**pop** (отримує елемент з верхівки стека. При цьому він видаляється зі стека і його місце в верхівці стека займає наступний за ним відповідно до правила LIFO, а розмір стека зменшується на одиницю. При намаганні «виштовхнути» елемент з вже пустого стека, відбувається ситуація «незаповненість» стека

Кожна з цих операцій зі стеком виконується за фіксований час [O(1)](https://uk.wikipedia.org/wiki/%D0%9D%D0%BE%D1%82%D0%B0%D1%86%D1%96%D1%8F_%D0%9B%D0%B0%D0%BD%D0%B4%D0%B0%D1%83) і не залежить від розміру стеку