

Звіт

Лабораторна робота №4

“Дослідження структур даних стек та черга”

Студента группи ДA-12

Краковича Павла Дмитровича

Київ – 2021

1. 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 C++.

2. Варіант роботи - №11

3. Хід виконання роботи

* + №1 Структура даних(стек) Книга має наступні поля: назва, кількість сторінок,ціна. Виконати наступні завдання:

1) Створити стопку книг, розмір стопки. Написати функцію для сортування алгоритмом “бульбашкою”, відсортувати масив за зростанням

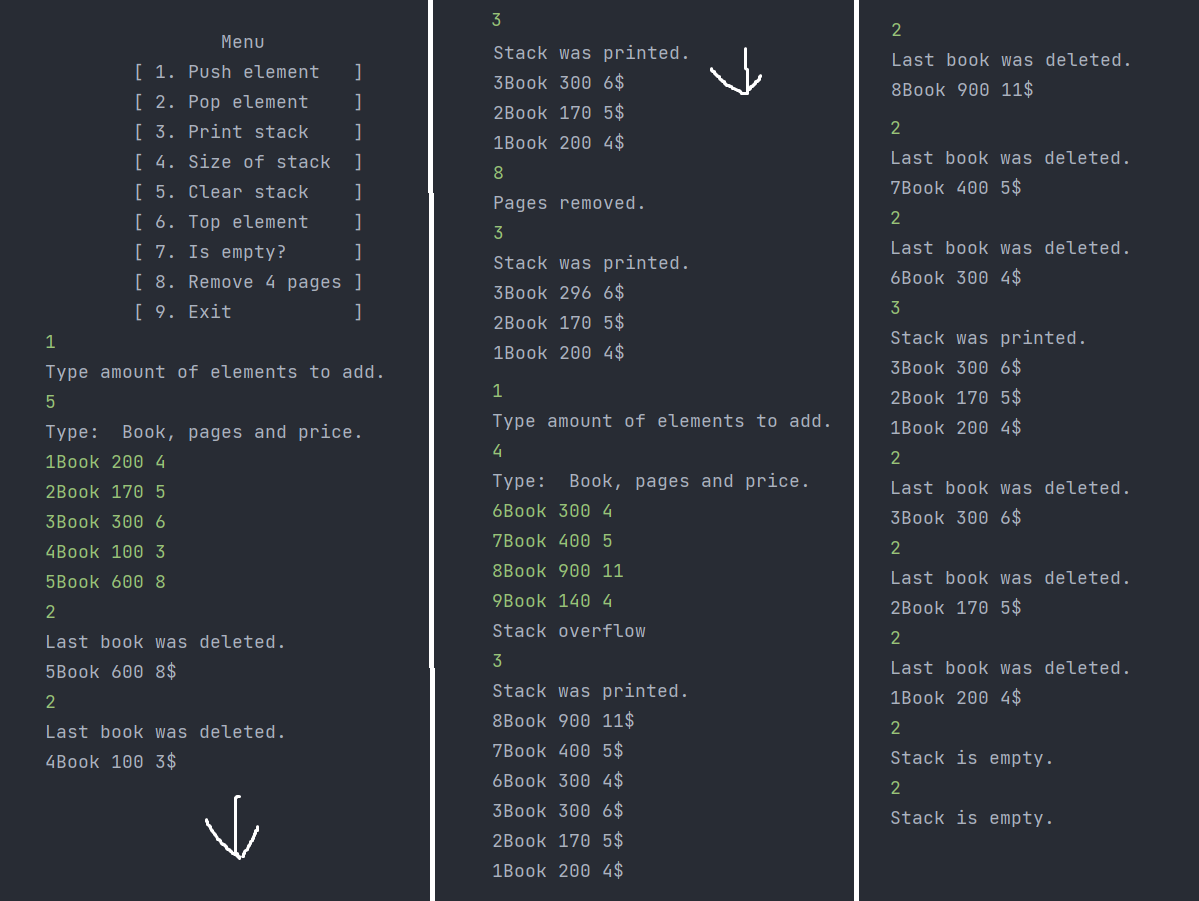
2) Додати в неї 5 різних книг.

3) Забрати 2 верхні книги.

4) Додати ще 4 нові книги у стопку.

5) Взяти верхню книгу, зменшити кількість сторінок на 10

6) Забрати 8 книги зі стопки.



#include <iostream>  
#include <string>  
using namespace std;  
#define STACK\_SIZE 6  
  
struct Book {  
 string name;  
 int pages;  
 int price;  
};  
  
struct Stack {  
 Book arr[STACK\_SIZE];  
 Book empty = {};  
 Book temp;  
 int top = -1;  
  
 void push(Book \*info) {  
 if (top == STACK\_SIZE - 1) {  
 cout << "Stack overflow" << endl;  
 } else {  
 top++;  
 arr[top] = \*info;  
 }  
 }  
  
 bool isEmpty() {  
 return top == -1;  
 }  
  
 Book pop() {  
 if (isEmpty()) {  
 return empty;  
 } else {  
 temp = arr[top];  
 arr[top] = empty;  
 top--;  
 return temp;  
 }  
 }  
  
 Book topValue() {  
 if (isEmpty()) {  
 return empty;  
 } else {  
 temp = arr[top];  
 return temp;  
 }  
 }  
 int size() {  
 return top+1;  
 }  
 void clear() {  
 if (!isEmpty()) {  
 top = -1;  
 }  
 }  
  
 void deletePages() {  
 arr[top].pages = arr[top].pages - 4;  
 }  
};  
  
  
void display(struct Stack shelf) {  
 int counter=0;  
 while (!shelf.isEmpty()){  
 Book temp = shelf.pop();  
 cout << temp.name << " " << temp.pages << " " << temp.price <<"$" << endl;  
 counter++;  
 }  
 shelf.top = counter-1;  
}  
  
  
int main() {  
  
 Stack shelf;  
 int menu;  
 cout << "\t\tMenu\n" << "\t[ 1. Push element ]\n" << "\t[ 2. Pop element ]\n" << "\t[ 3. Print stack ]\n" << "\t[ 4. Size of stack ]\n" << "\t[ 5. Clear stack ]\n" << "\t[ 6. Top element ]\n" << "\t[ 7. Is empty? ]\n"<< "\t[ 8. Remove 4 pages ]\n"<< "\t[ 9. Exit ]\n";  
 do {  
 cin >> menu;  
 if (menu==1) {  
 string name;  
 int pages,price,repeats=0;  
 cout << "Type amount of elements to add.\n";  
 cin >> repeats;  
 cout << "Type: Book, pages and price.\n";  
 for (int i=0; i<repeats; i++) {  
 cin >> name >> pages >> price;  
 Book info = {name, pages, price};  
 shelf.push(&info);  
 }  
  
 }  
 if (menu==2) {  
 Book temp = shelf.pop();  
 if (temp.name==shelf.empty.name) {  
 cout << "Stack is empty.\n";  
 }  
 else {  
 cout << "Last book was deleted.\n";  
 cout << temp.name << " " << temp.pages << " " << temp.price <<"$" << endl;  
 }  
 }  
 if (menu==3) {  
 if (shelf.size()<1) {  
 cout << "Stack is empty.\n";  
 }  
 else {  
 int counter=0;  
 cout << "Stack was printed.\n";  
 display(shelf);  
 }  
  
 }  
 if (menu==4) {  
 cout << "Size of Stack is: ";  
 cout << shelf.size() << endl;  
 }  
 if (menu==5) {  
 if (shelf.top==-1) {  
 cout << "Stack is empty.\n";  
 } else {  
 cout << "Stack was cleared.\n";  
 shelf.clear();  
 }  
 }  
 if (menu==6) {  
 if (shelf.top==-1) {  
 cout << "Stack is empty.\n";  
 }  
 else {  
 cout << "Top value is:\n";  
 cout << shelf.topValue().name << " " << shelf.topValue().pages << " " << shelf.topValue().price <<"$"<< endl;  
 }  
 }  
 if (menu==7) {  
 if (shelf.isEmpty()) {  
 cout << "Yes.\n";  
 }  
 else {  
 cout << "No.\n";  
 }  
 }  
 if (menu==8) {  
 if (shelf.arr[shelf.top].pages < 4 || shelf.size()==0) {  
 cout << "I can not remove pages.\n";  
 }  
 else {  
 cout << "Pages removed.\n";  
 shelf.deletePages();  
 }  
 }  
  
 }  
 while (menu!=9);  
 return 1;  
}

№2 Структура даних Товар має наступні поля: назва, ціна, строк придатності(днів). Виконати наступні завдання:

1) Створити стрічку (чергу) товарів на касі магазину, розмір стрічки 5.

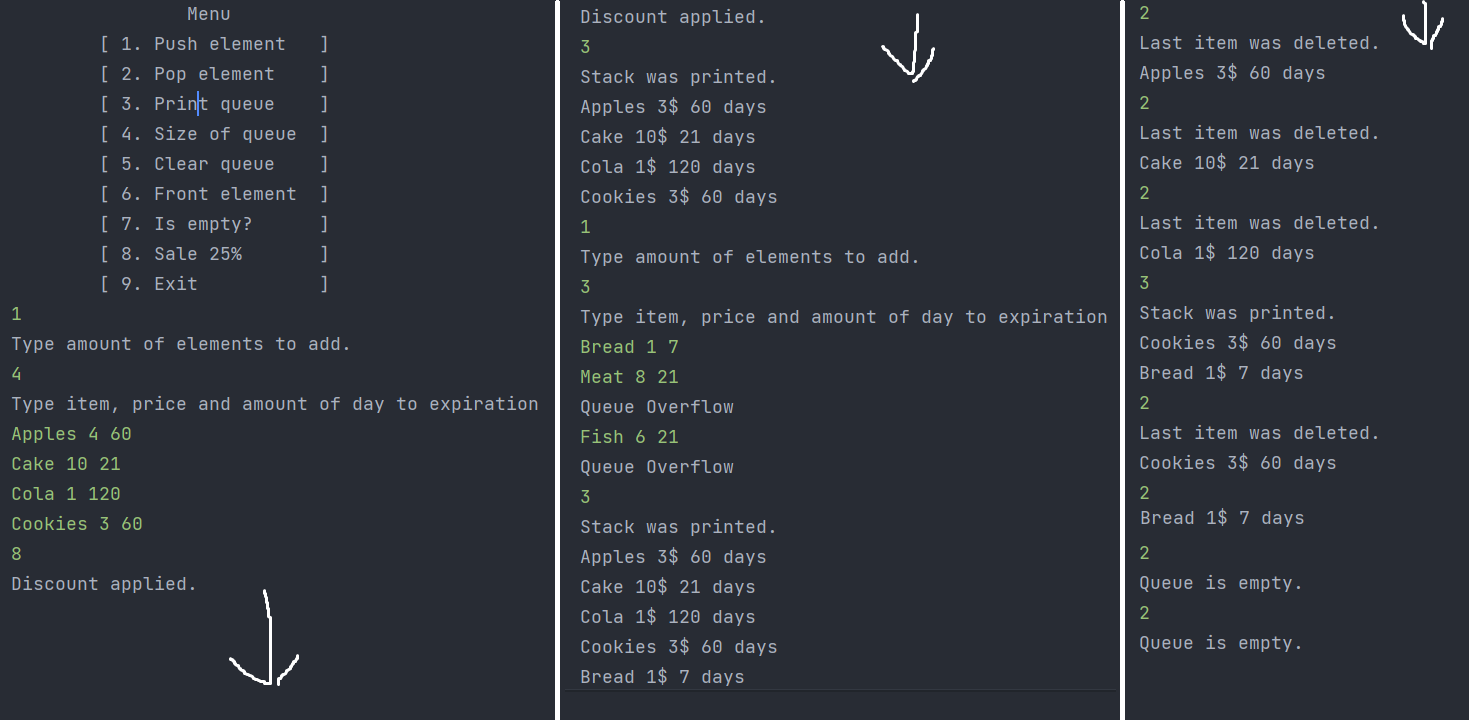
2) Покласти на неї 4 різні товари.

3) Взяти перший товар, застосувати до нього купон на знижку 25% та повернути на стрічку.

4) Покласти ще 3 товари.

5) Забрати 3 товари зі стрічки.

6) Забрати ще 4 товари зі стрічки.



#include <iostream>  
#include <string>  
using namespace std;  
#define QUEUE\_SIZE 5  
  
struct Item {  
 string name;  
 int price;  
 int expiration;  
};  
  
struct Queue {  
 Item arr[QUEUE\_SIZE];  
 Item empty = {};  
 int front = 0,rear = 0;  
 void push(Item \*info) {  
 if (rear == QUEUE\_SIZE)  
 cout << "Queue Overflow" << endl;  
 else {  
 rear++;  
 arr[rear] = \*info;  
 }  
 }  
  
 bool isEmpty() const {  
 return front == rear;  
 }  
  
 Item pop() {  
 if (isEmpty()) {  
 return empty;  
 } else {  
 front++;  
 Item temp = arr[front];  
 return temp;  
 }  
 }  
 Item frontValue() {  
 if (isEmpty()) {  
 return empty;  
 } else {  
 Item temp = arr[front+1];  
 return temp;  
 }  
 }  
 int size() const {  
 return rear - front;  
 }  
 void clear() {  
 if (!isEmpty()) {  
 front = 0;  
 rear = 0;  
 }  
 }  
  
 void sale() {  
 arr[front+1].price = arr[front+1].price / 4 \* 3;  
 }  
};  
  
void display(struct Queue items) {  
 int counter=0;  
 while (items.front < items.rear) {  
 Item print = items.pop();  
 counter++;  
 cout << print.name << " " << print.price << "$ " << print.expiration << " days" << endl;  
 }  
 items.front -= counter;  
}  
  
  
int main() {  
  
 Queue items;  
  
 int menu;  
 cout << "\t\tMenu\n" << "\t[ 1. Push element ]\n" << "\t[ 2. Pop element ]\n" << "\t[ 3. Print queue ]\n" << "\t[ 4. Size of queue ]\n" << "\t[ 5. Clear queue ]\n" << "\t[ 6. Front element ]\n" << "\t[ 7. Is empty? ]\n"<< "\t[ 8. Sale 25% ]\n"<< "\t[ 9. Exit ]\n";  
 do {  
 cin >> menu;  
 if (menu==1) {  
 string name;  
 int price,expiration,repeats=0;  
 cout << "Type amount of elements to add.\n";  
 cin >> repeats;  
 cout << "Type item, price and amount of day to expiration\n";  
 for (int i=0; i<repeats; i++) {  
 cin >> name >> price >> expiration;  
 Item info = {name, price, expiration};  
 items.push(&info);  
 }  
 }  
 if (menu==2) {  
 Item temp = items.pop();  
 if (temp.name == items.empty.name) {  
 cout << "Queue is empty.\n";  
 }  
 else {  
 cout << "Last item was deleted.\n";  
 cout << temp.name << " " << temp.price << "$ " << temp.expiration << " days" <<endl;  
 }  
 }  
 if (menu==3) {  
 if (items.isEmpty()) {  
 cout << "Queue is empty.\n";  
 }  
 else {  
 cout << "Stack was printed.\n";  
 display(items);  
 }  
  
 }  
  
 if (menu==4) {  
 if (items.isEmpty()) {  
 cout << "Queue is empty.\n";  
 } else {  
 cout << "Size of queue is:\n";  
 cout << items.size() << endl;  
 }  
 }  
 if (menu==5) {  
 cout << "Queue cleared\n";  
 items.clear();  
 }  
 if (menu==6) {  
 if (items.isEmpty()) {  
 cout << "Queue is empty.\n";  
 } else {  
 cout << "Front value is:\n";  
 cout << items.frontValue().name << " " << items.frontValue().price << "$ " << items.frontValue().expiration << " days" << endl;  
 }  
 }  
 if (menu==7) {  
 if (items.isEmpty()) {  
 cout << "Yes.\n";  
 } else {  
 cout << "No.\n";  
 }  
 }  
 if (menu==8) {  
 items.sale();  
 cout << "Discount applied.\n";  
 }  
 if (menu==0) {  
 //For Debug  
 cout << "rear:" << items.rear << "front:" << items.front<< endl;  
 }  
 }  
 while (menu!=9);  
  
 return 0;  
}

№3 Реалізувати алгоритм на основі власноруч написаного стеку з попередніх завдань, який перевіряє чи є коректною послідовність відкриваючих та закриваючих дужок різного типу.



#include <iostream>  
using namespace std;  
  
#define STACK\_SIZE 255  
  
struct Stack {  
  
 char arr[STACK\_SIZE];  
 int top = -1;  
  
 void push(char b) {  
 if (top == STACK\_SIZE - 1) {  
 return;  
 } else {  
 top++;  
 arr[top] = b;  
 }  
 }  
  
 void pop() {  
 if (isEmpty()) {  
 return;  
 } else {  
 char b = arr[top];  
 arr[top] = char();  
 top--;  
 }  
 }  
  
 bool isEmpty() const {  
 return top == -1;  
 }  
  
 char getTop() {  
 if (isEmpty()) {  
 return arr[0];  
 } else {  
 return arr[top];  
 }  
 }  
};  
  
char getInverseBrace(char brace) {  
 switch (brace) {  
 case '{':  
 return '}';  
 case '(':  
 return ')';  
 case '[':  
 return ']';  
 default:  
 return ' ';  
 }  
}  
  
string braces(string braces,int length) {  
  
 Stack st;  
 for (int i = length - 1; i >= 0; i--) {  
 if (st.getTop() == getInverseBrace(braces[i])) {  
 st.pop();  
 } else {  
 st.push(braces[i]);  
 }  
 }  
  
 if (st.isEmpty()) {  
 return "yes";  
 } else {  
 return "no";  
 }  
}  
  
int main() {  
 string str;  
 cin >> str;  
 int length = str.length();  
 if (length%2!=0) {  
 cout << "no";  
 return 0;  
 }  
 else {  
 cout << braces(str, length);  
 }  
 return 0;  
}

Висновки

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

Я навчився як правильно передавати вказівники у функції та як використовувати вказівники на структури, написав програми для вирішення двох задач свого варіанту.

LIFO – Першим із стеку видаляється елемент, який був поміщений туди останнім, тобто в стеку реалізується стратегія «останнім увійшов – першим вийшов» (last-in, first-out – LIFO). Прикладом стека в реальному житті може бути стос тарілок: коли ми хочемо витягнути тарілку, ми повинні зняти всі тарілки вище. Повернемося до опису операцій стеку:

FIFO – Першим із черги видаляється елемент, який був поміщений туди першим, тобто в черзі реалізується принцип «першим увійшов – першим вийшов» (англ. first-in, first-out – FIFO). У черзі є голова (англ. head) та хвіст (англ. tail). Коли елемент ставиться у чергу, він займає місце у її хвості. З черги завжди виводиться елемент, що знаходиться у її голові.  
Асимптотична складність у кожного методу стеку/черги має бути O(1)