Балтийский государственный технический университет  
«ВОЕНМЕХ» им. Д. Ф. Устинова

Кафедра О7 «Информационные системы и программная инженерия»

**Практическая работа №1**по дисциплине «Структуры и организация данных»  
на тему «Линейные структуры данных»  
часть 2 «Стеки, очереди, деки»  
  
Вариант 4

Выполнил:  
Студент Антонов Н.М.  
Группа И924Б  
  
Преподаватель:  
Гладевич А. А.

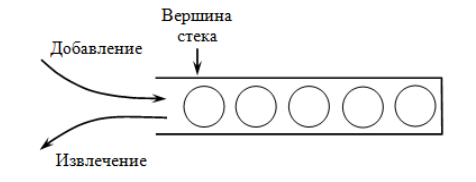
Санкт-Петербург  
2023 г.

Структура данных – стек.

Уровень сложности – ***Средний.***

Написать на языке С++ программу, решающую поставленную в вариативной части задачу, используя указанную там же структуру данных. Выполнение задания осуществляется в два этапа. На первом этапе требуется реализовать необходимую структуру данных с помощью одной структуры хранения (векторной или связной), реализацию оформить в виде класса. Методы класса должны выполнять только основные и 2 дополнительные операции, допустимые над элементами описываемой СД (перечислены ниже). Никаких других операций в классе быть не должно. На втором этапе требуется использовать объекты созданного класса при решении поставленной задачи.

Схематичное изображение структуры данных:

**

Схематичное изображение структуры хранения, использованной в программе:

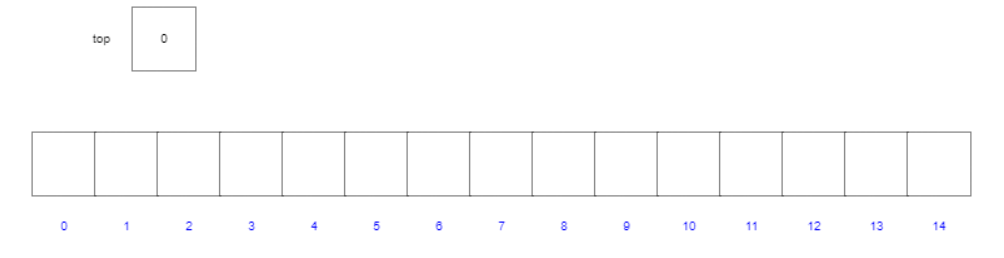
**

Рисунок 1 – Пустой стек

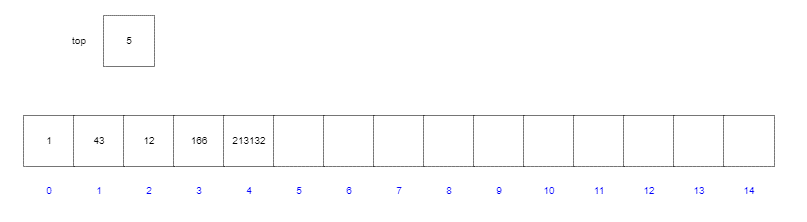


Рисунок 2 – Частично заполненный стек

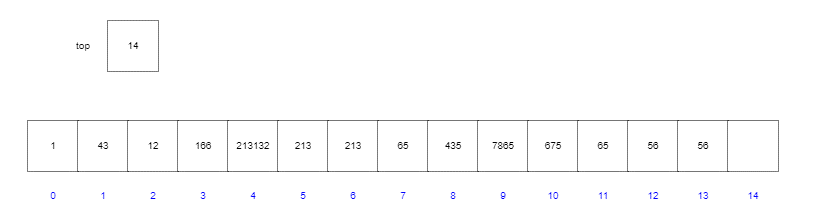


Рисунок 3 – Заполненный стек

Текст программы:

#include <iostream>

#include <ctime>

#include <cstdlib>

#include <iomanip>

using namespace std;

const int MAX = 100;

//структура оборудование

struct obor {

int count = 0;

double price = 0;

obor(int c, double p) {

count = c;

price = p;

}

obor() {

int count = 0;

double price = 0;

}

};

// стэк

class stack {

obor data[MAX];

int ind;

public:

stack() {

ind = -1;

}

// Конструктор копирования

stack(const stack& other) {

ind = other.ind;

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

data[i] = other.data[i];

}

}

// Перегрузка оператора присваивания

stack& operator=(const stack& other) {

if (this != &other) {

ind = other.ind;

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

data[i] = other.data[i];

}

}

return \*this;

}

~stack() {

cout << "Destructor was called" << endl;

}

void push(obor value) {

if (ind == MAX - 1) {

cout << "Stack overflow" << endl;

return;

}

ind++;

data[ind] = value;

}

obor pop() {

obor top = data[ind];

ind--;

return top;

}

obor top() {

return data[ind];

}

bool empty() {

return ind == -1;

}

bool full() {

return ind == MAX - 1;

}

};

int random(int min, int max) {

return min + rand() % (max - min + 1);

}

int main() {

srand(time(nullptr));

char pref, menu;

int count, countr = 0, counts = 0, summy, county, razn;

double summs = 0, summr = 0, price, temp, sellsr = 0, sells = 0;

stack s1, s2;

obor o;

for (int i = 0; i < 15; i++) {

temp = rand() % 201 + 100;

price = (temp / 100) + 0.03;

count = (rand() % 12) + 1;

obor o(count, price);

s1.push(o);

s2 = s1;

}

do {

system("cls");

s2 = s1;

cout << "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" << endl;

cout << "| Count | Price + 30% | Original price |" << endl;

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

if (s2.empty() == 0) {

o = s2.pop();

double sum = o.count \* o.price;

cout << "|" << internal << setw(5) << o.count << " |" << internal << setw(9) << setprecision(4) << sum \* 1.3 << "$" << " |" << internal << setw(11) << setprecision(4) << sum << "$" << " |" << endl;

}

else break;

}

cout << "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" << endl << endl;

cout << "--------------------------------------" << endl;

cout << "Choose what to do..." << endl;

cout << "'R'. Add new pieces" << endl;

cout << "'S'. Sell pieces" << endl;

cout << "0. Exit" << endl;

cout << "--------------------------------------" << endl;

cin >> menu;

switch (tolower(menu)) {

case 'r': {

system("cls");

cout << "It will be operation of buying" << endl;

cout << "Enter count of pieces" << endl;

do {

cin >> count;

if (count < 1 || count > 76) cout << "Enter value from 1 to 76!!!" << endl;

} while (count < 1 || count > 76);

if (s1.full()) {

cout << "Can't add pieces cause of stack overflow" << endl;

goto h;

}

else {

cout << "Enter price for 1 piece" << endl;

do {

cin >> price;

if (price < 1 || price > 100) cout << "Enter value from 1 to 100!!!" << endl;

} while (price < 1 || price > 100);

if (s1.full()) {

cout << "Can't add pieces cause of stack overflow, added only " << countr << " pieces" << endl;

break;

}

else {

obor o(count, price);

s1.push(o);

countr += o.count;

}

}

s2 = s1;

cout << "You succesfully added pieces" << endl;

h:

system("pause");

break;

}

case 's': {

system("cls");

s2 = s1;

cout << "Type number of selled pieces" << endl;

do {

cin >> counts;

summy = counts;

if (counts < 1 || counts > 100) cout << "Enter value from 1 to 100" << endl;

} while (counts < 1 || counts > 100);

for (int i = 0; counts > 0; i++) {

if (s1.empty() == 0) {

o = s1.top();

if (o.count - counts > 0) {

s1.pop();

razn = o.count - counts;

countr += counts;

counts -= o.count - razn;

summr += o.price \* o.count;

obor i(razn, o.price \* razn);

s1.push(i);

razn = 0;

}

else {

s1.pop();

countr += o.count;

counts -= o.count;

summr += o.price \* o.count;

}

}

else break;

}

if (counts == 0) cout << "Succesfully sold " << summy << " pieces on summ of " << summr \* 1.3 << "$" << endl;

else if (counts > 0) {

cout << "Didn't enough pieces to sold all, need more " << summy - countr << " pieces" << endl;

goto g;

}

s2 = s1;

cout << "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" << endl;

cout << "| Count | Price + 30% | Original price |" << endl;

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

if (s2.empty() == 0) {

o = s2.pop();

cout << "|" << internal << setw(5) << o.count << " |" << internal << setw(9) << setprecision(4) << (o.price \* 1.3) << "$" << " |" << internal << setw(11) << setprecision(4) << o.price << "$" << " |" << endl;

}

else break;

}

cout << "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~" << endl << endl;

countr = 0;

counts = 0;

g:

system("pause");

break;

}

case '0': {

system("cls");

cout << "Exiting program..." << endl;

break;

}

default: {

cout << "Invalid choice try again" << endl;

break;

}

}

} while (menu != '0');

return 0;

}

Результаты работы программы:

При запуске выводится стек и меню программы с выбором действий, что показано на рисунке 4.

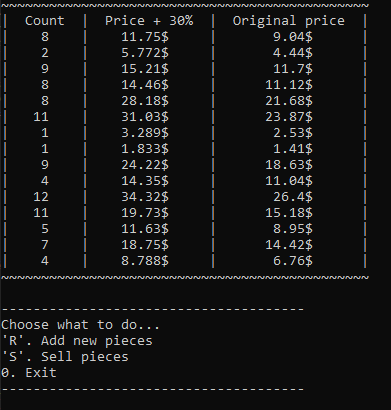


Рисунок 4 – Запуск программы

При нажатии «R» или «r» можно добавить элемент в стек, что показано на рисунке 5.

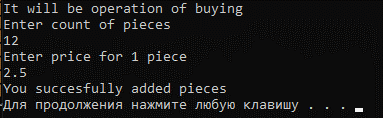


Рисунок 5 – Добавление элемента в стек

При введении некорректного значения выводится ошибка, что показано на рисунке 6.

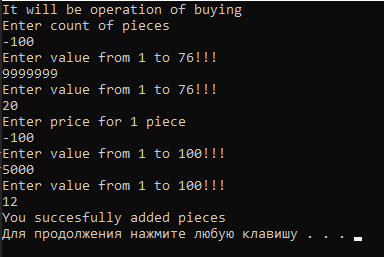


Рисунок 6 – Ввод некорректного значения

При нажатии «S» или «s» в главном меню можно продать оборудование на введенную с клавиатуры сумму, после чего выведется сообщения либо об успешной продаже требуемого количества оборудования, либо о продаже только той части, которой было возможно, рисунок 7.

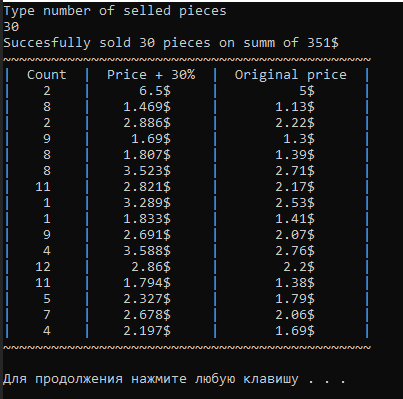


Рисунок 7 – Функция продажи оборудования

Затем из стека удаляется проданное количество оборудования.

При введении некорректного значения продажи оборудования выводится ошибка, рисунок 8.

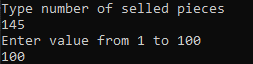


Рисунок 8 – Ввод некорректного значения

При нажатии «0» срабатывает деструктор и происходит выход из программы, что показано на рисунке 10.



Рисунок 10 – Завершение программы