Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра Программной Инженерии

Лабораторная работа 7

По дисциплине «Основы алгоритмизации и программирования»

На тему «**[Полустатические структуры данных: стеки](#Лаб7)**»

Выполнил:

Студент 1 курса 3 группы

Шатерник Г.И.

Преподаватель: асс. Андронова М.В.

2023, Минск

Вариант 15

4. Создать проект, демонстрирующий работу со стеком, организованным на основе *списка*,в соответствии со своим вариантом для данных из таблицы ниже. Все операции со стеком реализовать через функции. Дополнить проект функциями очистки стека **clear()**, сохранения в файл и считывания из файла.

Проект должен содержать три части: главная функция, файл с функциями работы со стеком и заголовочный файл. Создать интерфейс в виде меню.

Разработать функцию, которая удаляет первый повторяющийся элемент стека.

|  |
| --- |
| Программный код(main cpp) |
| #include <iostream>  #include <fstream>  #include <string>  using namespace std;  struct STACK { //создание структуры типа STACK  int data;  STACK\* head; //указатель на последний элемент  STACK\* next; //указатель на следующий элемент  };  void AddEl(int elem, STACK\* stack); //функция добавления элементов  int TakeFromStack(STACK\* stack); //функция извлечения из стека  void PrintStack(STACK\* stack); //вывод  void ToFile(STACK\* stack); //вывод  void FromFile(STACK\* stack); //ввод  int DeleteSame(STACK\* stack); //удаление совпадений  void clear(STACK\* stack); //очистка стека  void DeleteSameHelp(STACK\* stack, int point); //для работы функции удаления совпадений, тк функция DeleteSame типа int, то ее удобно использовать как аргумент в других функциях, рекомендую😊  int main()  {  setlocale(LC\_ALL, "rus");  int choice, newEl, position;  STACK\* stack = new STACK;  stack->head = NULL;  while (true) //меню  {  cout << "1 - добавить элемент в стек\n";  cout << "2 - извлечь элемент из стека\n";  cout << "3 - вывод стека на экран\n";  cout << "4 - удалить из стека первый повторяющийся элемент\n";  cout << "5 - запись в файл\n";  cout << "6 - вывод из файла\n";  cout << "7 - очистка стека\n";  cout << "0 - выход\n";  cin >> choice;  switch (choice)  {  case 1: {  cout << "\n";  cout << "Значение > "; cin >> newEl;  AddEl(newEl, stack);  cout << "\n";  break;  }  case 2: {  cout << TakeFromStack(stack) << endl;  break;  }  case 3: {  PrintStack(stack);  break;  }  case 4: {  DeleteSameHelp(stack, DeleteSame(stack));  break;  }  case 5: {  ToFile(stack);  break;  }  case 6: {  FromFile(stack);  break;  }  case 7: {  clear(stack);  cout << "\n\nСтек очищен!\n\n";  break;  }  case 0: {  exit(0);  break;  }  }  }  }  void AddEl(int elem, STACK\* stack)  {  STACK\* NewEl = new STACK; //создаем локальный стек для работы с основным стеком  NewEl->data = elem; //присваеваем элемент elem, передаваемый как аргумент  NewEl->next = stack->head; //тк мы добавили элемент, то он является последним, производим процесс присваения  stack->head = NewEl;  }  int TakeFromStack(STACK\* stack)  {  if (stack->head == NULL) //проверка на пустоту стека  {  cout << "Стек пуст";  return -1;  }  else {  STACK\* intermEl = new STACK;  int elData = stack->head->data;  stack->head = stack->head->next;  delete intermEl;  return elData;  }  }  void PrintStack(STACK\* stack)  {  STACK\* sthead = stack->head;  int elData;  if (sthead == NULL)  cout << "Стек пуст";  while (sthead != NULL)  {  elData = sthead->data;  cout << elData << " ";  sthead = sthead->next;  }  cout << "\n";  }  void ToFile(STACK\* stack)  {  ofstream getf;  getf.open("File13.txt");  if (getf.is\_open())  {  STACK\* StackHead = stack->head;  while (StackHead != NULL)  {  getf << StackHead->data << " ";  StackHead = StackHead->next;  }; cout << "\n";  }  getf.close();  cout << "Информация записана в файл!\n\n";  }  void FromFile(STACK\* stack)  {  string line;  ifstream outf("File13.txt");  if (outf.is\_open())  {  while (getline(outf, line))  {  outf >> line;  cout << line << " ";  }  }  outf.close();  cout << "\nИнформация выведена из файла!\n\n";  }  int DeleteSame(STACK\* stack)  {  STACK\* sthead = stack->head;  int elData, arr[20], i = 0, point = -1;  if (sthead == NULL)  cout << "Стек пуст";  while (sthead != NULL)  {  elData = sthead->data;  arr[i] = elData;  sthead = sthead->next;  i++;  }  for (int j = 0; j < i; j++)  {  int interm = arr[j], count = 0;  for (int p = 0; j + p - 1 < i; p++)  {  if (arr[p] == interm)  count++;  if (count > 1)  {  point = p;  j = i + 1;  break;  }  }  }  return point;  }  void DeleteSameHelp(STACK\* stack, int point)  {  STACK\* sthead1 = stack->head;  for (int k = 0; k < point - 1; k++)  sthead1 = sthead1->next;  sthead1->next = sthead1->next->next;  }  void clear(STACK\* stack)  {  if (stack->head == NULL)  cout << "Стек пуст";  while (stack->head != NULL)  {  STACK\* f = stack->head;  stack->head = stack->head->next;  delete f;  }  cout << "\n";  } |
| Программный код (stack cpp) |
| #include <iostream>  #include "stack.h"  using namespace std;  void push(char x, Stack\* myStk) //Добавление элемента х в стек  {  Stack\* e = new Stack; //выделение памяти для нового элемента  e->data = x; //запись элемента x в поле v  e->next = myStk->head; //перенос вершины на следующий элемент  myStk->head = e; //сдвиг вершины на позицию вперед  }  int pop(Stack\* myStk) //Извлечение (удаление) элемента из стека  {  if (myStk->head == NULL)  {  cout << "Стек пуст!" << endl;  return -1; //если стек пуст - возврат -1  }  else  {  Stack\* e = myStk->head; //е - переменная для хранения адреса элемента  char a = myStk->head->data; //запись числа из поля data в переменную a  myStk->head = myStk->head->next; //перенос вершины  delete e; //удаление временной переменной  return a; //возврат значения удаляемого элемента  }  }  void show(Stack\* myStk) //Вывод стека  {  Stack\* e = myStk->head; //объявляется указатель на вершину стека  char a;  if (e == NULL)  cout << "Стек пуст!" << endl;  while (e != NULL)  {  a = e->data; //запись значения в переменную a  cout << a << " ";  e = e->next;  }  cout << endl;  } |
| Программный код (stack.h)  #pragma once  struct Stack  {  char data;  Stack\* head;  Stack\* next;  };  void show(Stack\* myStk);  int pop(Stack\* myStk);  void push(char x, Stack\* myStk); |
| Вывод: |

1. **Список клиентов гостиницы.** Паспортные данные, даты приезда и отъезда, номер, тип размещения (люкс, одноместный, двухместный, трехместный, апартаменты). Поиск гостя по фамилии.

|  |
| --- |
| Программный код  #include <iostream>  #include <fstream>  #include <string>  using namespace std;  # define str\_len 256  # define size 100  #define \_CRT\_SECURE\_NO\_WARNINGS  void enter\_new(); //новая запись  void out(); //вывод записи  void del(); //удаление записи  void poisk(); //поиск записи  union hote  {  int pasport;  int kom;  int per;  };  union Day {  unsigned short day : 8;  unsigned mounth : 8;  unsigned year : 16;  };  int i = 0;  Day d;  union Perce  {  int Lux;  int Odnomest;  int Dvymest;  int Trehmest;  int Apart;  } percent;  hote products[size];  hote pasport[size];  hote all;  int choice;  int current\_size = 0;  int sell;  int main()  {  percent.Lux = 1;  percent.Odnomest = 2;  percent.Dvymest = 3;  percent.Trehmest = 4;  percent.Apart = 5;  setlocale(LC\_CTYPE, "Russian");  cout << "введите:" << endl;  cout << "1-для ввода записи" << endl;  cout << "2-для вывода записи" << endl;  cout << "3-для удаления" << endl;  cout << "4-для поиска информации" << endl;  cout << "5-для выхода" << endl;  cin >> choice;  do  {  switch (choice)  {  case 1: enter\_new(); break;  case 2: out(); break;  case 3: del(); break;  case 4: poisk(); break;  }  } while (choice != 5);  }  void enter\_new() //ввод записи  {  cout << "ввод информации" << endl;  if (current\_size < size)  {  cout << "строка номер ";  cout << current\_size + 1 << endl;  cout <<"Введите паспортные данные:" <<endl;  cin >>products[current\_size].pasport;  cout << "введите: день" << endl;  cin >> i;  if (i < 31)  {  d.day = i;  }  cout << "месяц" << endl;  cin >> i;  if (i < 13)  {  d.mounth = i;  }  cout << "год" << endl;  cin >> i;  if (i < 2024)  {  d.year = i;  }  cout <<"Выберете тип комнаты:"<<endl;  cout <<"1-Люкс" <<" " <<"2-Одноместный" <<" " <<"3-Двухметсный" <<"4-Трехместный" <<"5-Апартаменты" <<endl;  cin >> products[current\_size].per;  cout <<"Введите номер комнаты:" <<endl;  cin >>products[current\_size].kom;  current\_size++;  }  else  cout << "введено максимальное кол-во строк";  cout << "что дальше?" << endl;  cin >> choice;  }  void out() //вывод записи  {  int n;  cout << "введите № выводимой строки: ";  cin >> n;  cout <<"Паспортные данные:" <<endl;  cout << products[n-1].pasport <<endl;  cout << "дата вашего рейса: ";  cout << d.day << "\t" << d.mounth << "\t" << d.year << endl;  cout << "Комната: ";  cout << products[n - 1].kom << endl;  if (products[n - 1].per == 1)  {  cout << "Тип размещения:" << percent.Lux << endl;  }  else if (products[n - 1].per == 2)  {  cout << "Тип размещения:" << percent.Odnomest << endl;  }  else if (products[n - 1].per == 3)  {  cout << "Тип размещения:" << percent.Dvymest << endl;  }  else if (products[n - 1].per == 4)  {  cout << "Тип размещения:" << percent.Trehmest << endl;  }  else if (products[n - 1].per == 5)  {  cout << "Тип размещения:" << percent.Apart << endl;  }  cout << "что дальше?" << endl;  cin >> choice;  }  void del() //удаление записи  {  int d;  cout << "\nномер строки, которую надо удалить (для удаления всех строк нажать 99)" << endl;  cin >> d;  if (d != 99)  {  for (int de1 = (d - 1); de1 < current\_size; de1++)  products[de1] = products[de1 + 1];  current\_size = current\_size - 1;  }  if (d == 99)  for (int i = 0; i < size; i++)  products[i] = all;  cout << "что дальше?" << endl;  cin >> choice;  }  void poisk() { //поиск записи  cout << "введите Фамилию:" <<endl;  int fam;  cin >> fam;  for (int z = 0; z < current\_size; z++)  {  if (fam == products[z].pasport) {  cout <<"День-месяц-год" <<endl;  cout <<d.day <<" " <<d.mounth <<" " <<d.year;  cout << endl << "Комната:" <<endl;  cout <<products[z].kom <<endl;  if (products[z].per == 1)  {  cout << "Тип размещения:" << percent.Lux << endl;  }  else if (products[z].per == 2)  {  cout << "Тип размещения:" << percent.Odnomest << endl;  }  else if (products[z].per == 3)  {  cout << "Тип размещения:" << percent.Dvymest << endl;  }  else if (products[z].per == 4)  {  cout << "Тип размещения:" << percent.Trehmest << endl;  }  else if (products[z].per == 5)  {  cout << "Тип размещения:" << percent.Apart << endl;  }  else {  cout << "никаких комнат не обнаружено" << endl;  }  }  }  cout << "что дальше?" << endl;  cin >> choice;  } |
| Вывод: |