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»

Выполнила:

Студентка 1 курса 2 группы

Глухова Д.В.

Преподаватель: асс. Андронова М.В.

2023, Минск

**Вариант 2.**

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 из стека первый отрицательный элемент, если такой есть.

|  |
| --- |
| Программный код |
| Программный модуль с главной функцией |
| #include "head.h"  void menu() {  cout << "\n-----------------------------------------------------\n"  << "0) Выход\n"  << "1) Запись чисел в стек\n"  << "2) Вывод стека\n"  << "3) Очистка стека\n"  << "4) Найти первый отрицательный элемент и удалить его\n"  << "5) Запись стека в файл\n"  << "6) Считать стек из файла\n"  << "\n-----------------------------------------------------\n"  << "Ваш выбор : ";  }  int main() {  SetConsoleCP(1251);  SetConsoleOutputCP(1251);  int number, var;  Stack\* MyStack = new Stack; // выделение памяти  MyStack->head = nullptr;  while (true) {  menu();  cin >> var;  switch (var) {  case 0: return 0;  case 1:  cout << "Введите элементы первого стека:" << endl;  push(MyStack);  break;  case 2:  cout << "Элементы стека:" << endl;  showStack(MyStack);  break;  case 3:  clearStack(MyStack);  break;  case 4: find(MyStack);  break;  case 5:  WriteInFile(MyStack, "text.txt");  break;  case 6:  cout << "Элементы первого стека:" << endl;  showStack(MyStack);  break;  break;  }  }  } |
| Модуль с функциями |
| #include "head.h"  // функция для записи чисел  void push(struct Stack\*& MyStack) {  char chislo[9]; // символ, который будет вводится с клавиатуры  while (cin >> chislo && \*chislo != 'q') { // пока не введем q  Stack\* data = new Stack; //запрашиваем память под структуру для стека  data->number = atoi(chislo); // приведение строки в число  data->next = MyStack->head; // заполняем поля структуры  MyStack->head = data; // устанавливаем указатель на вершину стека  }  cout << "Элементы были успешно добавлены\n";  return;  }  // для очистки стека  void clearStack(Stack\*& MyStack) {  if (!MyStack->head) { cout << "стек пуст\n"; }  else {  while (MyStack->head != NULL) { // пока стек не пустой  Stack\* delStack = MyStack->head; // хранение адреса элемента  MyStack->head = MyStack->head->next; // переход к следующему  delete delStack; // удаление элемента  }  cout << "стек пуст\n";  }  }  // вывод на экран  void showStack(Stack\*& MyStack) {  Stack\* data = MyStack->head; // хранение адреса элемента  if (!MyStack->head) { cout << "стек пуст\n"; }  else {  while (data) {  cout << data->number << "\n"; // вывод  data = data->next; // переход к следующему  }  }  }  // задание из варианта  void find(Stack\* MySt1) {  if (!MySt1->head) { cout << "стек пуст\n"; }  else {  Stack\* el = MySt1->head; // хранение адреса элемента  bool stop = true;  while (stop)  {  if (el->number > 0) // если > 0, перемещаемся дальше  {  el= el->next;  }  if (el->number < 0) // если < 0  {  el->number = NULL; // устанавливаем отрицательный элемент в ноль  el = el->next; // перемещаем дальше  stop = false;  }  }  showStack(MySt1);  }  }  // запись в файл  void WriteInFile(Stack\* MySt, string fileName) {  if (!MySt->head) { cout << "стек пуст\n"; }  else {  ofstream FILE; // открываем для записи  FILE.open(fileName);  Stack\* MyStack = new Stack; // выделение памяти  MyStack->head = MySt->head; // указатель на адрес элемента  while (MyStack->head) {  FILE << "\n\t\t" << MyStack->head->number; // запись  MyStack->head = MyStack->head->next; // перемещение дальше  }  cout << "стек записан в " << fileName << endl;  }  }  // чтение из файла  void ReadFromFile(Stack\* MySt, string fileName) {  int indicate;  ifstream FILE; // открываем для чтения  FILE.open(fileName, ofstream::app);  if (!(FILE >> indicate)) { cout << "ошибка чтения\n"; }  else {  while (indicate) {  Stack\* data = new Stack; // выделение памяти  data->number = indicate;  data->next = MySt->head;  MySt->head = data;  if (!(FILE >> indicate)) { break; }  }  cout << "данные выведены\n";  }  FILE.close();  } |
| заголовочный файл **head.h**, который содержит описание структуры стека и прототипы программ обработки его элементов.  #pragma once  #include <iostream>  #include <windows.h>  #include <fstream>  using namespace std;  // задаем наш стек  struct Stack {  int number;  Stack\* head;  Stack\* next;  };  void menu();  void push(struct Stack\*& MyStack); // функция для записи чисел  void showStack(Stack\*& MyStack);// вывод на экран  void clearStack(Stack\*& MyStack);// для очистки стека  void find(Stack\* MySt1);// задание из варианта  void WriteInFile(Stack\* MySt, string fileName);// запись в файл  void ReadFromFile(Stack\* MySt, string fileName);// чтение из файла |
| Консоль отладки |
|  |
| Конечный файл |
|  |

Дополнительные задания.

1. Дана величина *a* строкового типа из четного количества символов. Получить и напечатать величину *b*, состоящую из символов первой половины величины *a*, записанных в обратном порядке, после которых идут символы второй половины величины *a*, также записанные в обратном порядке (например, при *а* = “привет” *b* должно быть равно «ипртев»).

|  |
| --- |
| Программный код |
| #include <iostream>  #include <windows.h>  using namespace std;  struct Stack  {  char str;  Stack\* head;  Stack\* next;  };  void dop1(Stack\* MyStk);  int main() {  SetConsoleCP(1251);  SetConsoleOutputCP(1251);  setlocale(LC\_ALL, ".1251");  Stack\* MyStk = new Stack; //выделение памяти для стека  MyStk->head = NULL;  dop1(MyStk);  }  void dop1(Stack\* MyStk)  {  string a;  short i = 0;  cout << "Введите вашу строку:\n";  cin >> a;  short length = a.length();  for (i = 0; i < (length / 2); i++) // цикл для вывода первой части слова, тк стек работает по принципу LIFO мы сможем вывести первым последний элемент  {  Stack\* e = new Stack; //выделение памяти для нового элемента  e->str = a[i]; //запись элемента в поле  e->next = MyStk->head; //перенос вершины на следующий элемент  MyStk->head = e; //сдвиг вершины на позицию вперед  }  while (MyStk->head)  {  cout << MyStk->head->str; // выводим букву в конце  MyStk->head = MyStk->head->next; // перенос вершины на след  }  for (i = (length / 2); i < length; i++) // цикл для вывода второй части  {  Stack\* e = new Stack; //выделение памяти для нового элемента  e->str = a[i]; //запись элемента в поле  e->next = MyStk->head; //перенос вершины на следующий элемент  MyStk->head = e; //сдвиг вершины на позицию вперед  }  while (MyStk->head)  {  cout << MyStk->head->str;  MyStk->head = MyStk->head->next;  }  cout << '\n';  } |
| Консоль отладки |
|  |

2. Создать стек с целочисленным информационным полем. Заполнить его *длинами строк*, считанных из файла. Распечатаьть на экране содержимое стека. Указать номер и длину последней самой короткой строки файла.

|  |
| --- |
| Программный код |
| #include <iostream>  #include <windows.h>  #include <fstream>  #include <string>  using namespace std;  struct Stack  {  int data; // целочисленное информационное поле  Stack\* head;  Stack\* next;  };  void dop2(Stack\* MyStk);  int main() {  SetConsoleCP(1251);  SetConsoleOutputCP(1251);  setlocale(LC\_ALL, ".1251");  Stack\* MyStk = new Stack; //выделение памяти для стека  MyStk->head = NULL;  dop2(MyStk);  }  void dop2(Stack\* MyStk)  {  ifstream file("text.txt");  string str;  short counter = 1, seqNum = 1, min = SHRT\_MAX, minData;  int\* Array = new int[10];  while (getline(file, str))  {  Stack\* e = new Stack; // выделение памяти  e->data = str.length(); // заполняем целочисленное поле  if (e->data <= min) // поиск мин строки  {  min = e->data;  minData = min; // для мин длины  seqNum = counter; // для номера строки  }  e->next = MyStk->head;  MyStk->head = e;  counter++;  }  cout << "Длины строк: \n"; // с конца файла, тк LIFO  while (MyStk->head)  {  cout << MyStk->head->data << "\t";  MyStk->head = MyStk->head->next;  }  cout << "\nCамая короткая строка номер " << seqNum << " длина " << minData << ".\n";  } |
| Консоль отладки |
|  |
| Исходный файл |
|  |

Дополнительный вариант 3

Разработать функцию, которая удаляет первый положительный элемент, если такой есть.

|  |
| --- |
| Программный код |
| // Разработать функцию, которая удаляет первый положительный элемент, если такой есть.  #include <iostream>  #include <windows.h>  #include <fstream>  using namespace std;  // задаем наш стек  struct Stack {  int number;  Stack\* head;  Stack\* next;  };  void menu();  void push(struct Stack\*& MyStack); // функция для записи чисел  void showStack(Stack\*& MyStack);// вывод на экран  void clearStack(Stack\*& MyStack);// для очистки стека  void find(Stack\* MySt1);// задание из варианта  void WriteInFile(Stack\* MySt, string fileName);// запись в файл  void ReadFromFile(Stack\* MySt, string fileName);// чтение из файла  void menu() {  cout << "\n-----------------------------------------------------\n"  << "0) Выход\n"  << "1) Запись чисел в стек\n"  << "2) Вывод стека\n"  << "3) Очистка стека\n"  << "4) Найти первый положительный элемент и удалить его\n"  << "5) Запись стека в файл\n"  << "6) Считать стек из файла\n"  << "\n-----------------------------------------------------\n"  << "Ваш выбор : ";  }  int main() {  SetConsoleCP(1251);  SetConsoleOutputCP(1251);  int number, var;  Stack\* MyStack = new Stack; // выделение памяти  MyStack->head = nullptr;  while (true) {  menu();  cin >> var;  switch (var) {  case 0: return 0;  case 1:  cout << "Введите элементы первого стека:" << endl;  push(MyStack);  break;  case 2:  cout << "Элементы стека:" << endl;  showStack(MyStack);  break;  case 3:  clearStack(MyStack);  break;  case 4: find(MyStack);  break;  case 5:  WriteInFile(MyStack, "text.txt");  break;  case 6:  cout << "Элементы первого стека:" << endl;  showStack(MyStack);  break;  break;  }  }  }  // функция для записи чисел  void push(struct Stack\*& MyStack) {  char chislo[9]; // символ, который будет вводится с клавиатуры  while (cin >> chislo && \*chislo != 'q') { // пока не введем q  Stack\* data = new Stack; //запрашиваем память под структуру для стека  data->number = atoi(chislo); // приведение строки в число  data->next = MyStack->head; // заполняем поля структуры  MyStack->head = data; // устанавливаем указатель на вершину стека  }  cout << "Элементы были успешно добавлены\n";  return;  }  // для очистки стека  void clearStack(Stack\*& MyStack) {  if (!MyStack->head) { cout << "стек пуст\n"; }  else {  while (MyStack->head != NULL) { // пока стек не пустой  Stack\* delStack = MyStack->head; // хранение адреса элемента  MyStack->head = MyStack->head->next; // переход к следующему  delete delStack; // удаление элемента  }  cout << "стек пуст\n";  }  }  // вывод на экран  void showStack(Stack\*& MyStack) {  Stack\* data = MyStack->head; // хранение адреса элемента  if (!MyStack->head) { cout << "стек пуст\n"; }  else {  while (data) {  cout << data->number << "\n"; // вывод  data = data->next; // переход к следующему  }  }  }  // задание из варианта  void find(Stack\* MySt1) {  if (!MySt1->head) { cout << "стек пуст\n"; }  else {  Stack\* el = MySt1->head; // хранение адреса элемента  bool stop = true;  while (stop)  {  if (el->number < 0) // если < 0, перемещаемся дальше  {  el = el->next;  }  if (el->number > 0) // если > 0  {  el->number = NULL; // устанавливаем отрицательный элемент в ноль  el = el->next; // перемещаем дальше  stop = false;  }  }  showStack(MySt1);  }  }  // запись в файл  void WriteInFile(Stack\* MySt, string fileName) {  if (!MySt->head) { cout << "стек пуст\n"; }  else {  ofstream FILE; // открываем для записи  FILE.open(fileName);  Stack\* MyStack = new Stack; // выделение памяти  MyStack->head = MySt->head; // указатель на адрес элемента  while (MyStack->head) {  FILE << "\n\t\t" << MyStack->head->number; // запись  MyStack->head = MyStack->head->next; // перемещение дальше  }  cout << "стек записан в " << fileName << endl;  }  }  // чтение из файла  void ReadFromFile(Stack\* MySt, string fileName) {  int indicate;  ifstream FILE; // открываем для чтения  FILE.open(fileName, ofstream::app);  if (!(FILE >> indicate)) { cout << "ошибка чтения\n"; }  else {  while (indicate) {  Stack\* data = new Stack; // выделение памяти  data->number = indicate;  data->next = MySt->head;  MySt->head = data;  if (!(FILE >> indicate)) { break; }  }  cout << "данные выведены\n";  }  FILE.close();  } |
| Консоль отладки |
|  |
| Исходный файл |
|  |