|  |  |
| --- | --- |
| **Задание** | **Краткие теоретические сведения** |
| 1. Изучить реализацию стека на основе *динамического массива*, выполнив программу, приведенную в правой части. | Пример. Разработать калькулятор целых чисел на основе стека.  #include <iostream>  #define MAX 100  int\* p; // указатель на область свободной памяти  int\* tos, \* bos; // указатель на вершину и дно стека  void push(int i); //прототип  int pop(void); //прототип  void main(void)  {  setlocale(LC\_CTYPE, "Russian");  int a, b; char s[80];  p = new int[MAX \* sizeof(int)];  if (!p)  {  printf("Ошибка при выделении памяти\n");  exit(1);  }  tos = p; bos = p + MAX - 1;  printf("Калькулятор \n Для выхода нажать 'q'\n");  do  {  printf(": "); gets\_s(s); //ввод первого числа, второго и знака операции  switch (\*s)  {  case '+': a = pop(); b = pop(); //сложение  printf("%d\n", a + b);  push(a + b); break;  case '-': a = pop(); b = pop(); //вычитание  printf("%d\n", b - a);  push(b - a); break;  case '\*': a = pop(); b = pop(); //умножение  printf("%d\n", b \* a);  push(b \* a); break;  case '/': a = pop(); b = pop(); //деление  if (a == 0) { printf("Деление на 0\n"); break; }  printf("%d\n", b / a);  push(b / a); break;  case '.': a = pop(); push(a); //вывод вершины стека  printf("Текущее значение на вершине стека: %d\n", a);  break;  default: push(atoi(s)); //конвертация из символа в число  }  } while (\*s != 'q');  }  void push(int i) // Занесение элемента в стек  {  if (p > bos) { printf("Стек полон\n"); return; }  \*p = i; p++;  }  int pop(void) // Получение верхнего элемента из стека  {  p--;  if (p < tos) { printf("Стек пуст\n"); return 0; }  return \*p;  } |
| 2. В программе, приведенной справа, демонстрируется реализация стека на основе *односвязного* *списка*. | #include <iostream>  using namespace std;  struct stack {  int data;  stack\* next;  };  int pop(stack\*& st); // Извлечение элемента  void push(stack\*& st, int d); // Добавление элемента  int main() {  stack\* p = nullptr;  push(p, 100); //число 100 – в стек  push(p, 200); //число 200 – в стек  pop(p); //вывод текущего элемента = 200  pop(p); // вывод текущего элемента = 100  return 0;  }  int pop(stack\*& st) { // Извлечение элемента  int tmp = st->data;  stack\* pv = st;  st = st->next; // вершиной становится предшествующий элемент  delete pv; // освобождается память  cout << tmp << endl; //вывод текущего элемента  return tmp;  }  void push(stack\*& st, int d) { // Добавление элемента  stack\* pv = new stack;  pv->data = d; // значение помещается в стек  pv->next = st;  st = pv;  } |
| 3. В правой части представлен *проект, состоящий из трех* *частей*: программный модуль с главной функцией, программный модуль с функциями пользователя, осуществляющими операции со стеком, и заголовочный файл.  В программе создается стек из символов английского алфавита на основе *односвязного списка*. | Программный модуль с главной функцией:  #include <iostream>  #include "myStack.h"  using namespace std;  int main()  {  setlocale(LC\_ALL, "Rus");  int choice; char x;  Stack\* myStk = new Stack; //выделение памяти для стека  myStk = NULL; //инициализация первого элемента  for (;;)  {  cout << "Выберите команду:" << endl;  cout << "1 - Добавление элемента в стек" << endl;  cout << "2 - Извлечение элемента из стека" << endl;  cout << "3 - Запись в файл" << endl;  cout << "4 - Чтение из файла" << endl;  cout << "5 - Вывод стека с очисткой" << endl;  cout << "6 - Выход" << endl;  cin >> choice;  switch (choice)  {  case 1: cout << "Введите элемент: " << endl;  cin >> x;  push(x, myStk); break;  case 2: x = pop(myStk);  if (x != -1)  cout << "Извлеченный элемент: " << x << endl;  break;  case 3: toFile(myStk); break;  case 4: fromFile(myStk); break;  case 5: cout << "Весь стек: " << endl;  show(myStk); break;  case 6: return 0; break;  }  }  return 0;  }  Программный модуль с функциями пользователя, осуществляющими операции со стеком:  #include <iostream>  #include <fstream>  #include "myStack.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 data  e->next = myStk; //перенос вершины на следующий элемент  myStk = e; //сдвиг вершины на позицию вперед  }  char pop(Stack\*& myStk) //Извлечение (удаление) элемента из стека  {  if (myStk == NULL)  {  cout << "Стек пуст!" << endl;  return -1; //если стек пуст - возврат (-1)  }  else  {  Stack\* e = myStk; //е-переменная для хранения адреса элемента  char x = myStk->data; //запись элемента из поля data в перем. x  if (myStk)  myStk = myStk->next; //перенос вершины  delete e;  return x;  }  }  void toFile(Stack\*& myStk) //Запись в файл  {  Stack\* e = myStk;  Stack buf;  ofstream frm("mStack.dat");  if (frm.fail())  {  cout << "\n Ошибка открытия файла";  exit(1);  }  while (e)  {  buf = \*e;  frm.write((char\*)&buf, sizeof(Stack));  e = e->next;  }  frm.close();  cout << "Стек записан в файл mStack.dat\n";  }  void fromFile(Stack\*& myStk) //Считывание из файла  {  Stack buf, \* p = nullptr, \* e = nullptr;  ifstream frm("mStack.dat");  if (frm.fail())  {  cout << "\n Ошибка открытия файла";  exit(1);  }  frm.seekg(0);  frm.read((char\*)&buf, sizeof(Stack));  while (!frm.eof())  {  push(buf.data, e);  frm.read((char\*)&buf, sizeof(Stack));  }  frm.close();  while (e != NULL)  {  buf.data = pop(e);  push(buf.data, p);  myStk = p;  }  cout << "\nСтек считан из файла mStack.dat\n\n";  }  void show(Stack\*& myStk) //Вывод стека с очисткой  {  Stack\* e = myStk;  if (e == NULL)  cout << "Стек пуст!" << endl;  while (e != NULL)  cout << pop(e) << " ";  cout << endl;  }  Заголовочный файл:  #pragma once  struct Stack  {  char data; //информационный элемент  Stack\* next; //указатель на следующий элемент  };  void show(Stack\*& myStk);  char pop(Stack\*& myStk);  void push(char x, Stack\*& myStk);  void toFile(Stack\*& myStk);  void fromFile(Stack\*& myStk); |
| 10. Разработать функцию, которая определяет, есть ли в стеке хотя бы один элемент, лежащий в заданном диапазоне  14. Разработать функцию, которая формирует стек **Stack**, включив в него повторяющиеся элементы стеков **Stack1** и **Stack2**  15. Разработать функцию, которая удаляет первый повторяющийся элемент стека  16. Разработать функцию, которая удаляет из стека **Stack1** элементы, входящие в стек **Stack2**, но не входящие в стек **Stack1**, и наоборот. | #pragma once  struct Stack  {  int data; //информационный элемент  Stack\* next; //указатель на следующий элемент  };  void show(Stack\*& myStk);  int pop(Stack\*& myStk);  void push(int x, Stack\*& myStk);  void toFile(Stack\*& myStk);  void fromFile(Stack\*& myStk);  void checkDiapason(Stack\*& myStack, int min, int max);  void clear(Stack\*& myStk);  void fuseStack(Stack\*& stack1, Stack\*& stack2, Stack\*& resultStack);  void removeFirstDuplicate(Stack\*& myStk);  void delStacks(Stack\*& stack1, Stack\*& stack2, Stack\*& res);  #include <iostream>  #include <fstream>  #include "myStack.h"  using namespace std;  void push(int 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 data  e->next = myStk; //перенос вершины на следующий элемент  myStk = e; //сдвиг вершины на позицию вперед  }  int pop(Stack\*& myStk) //Извлечение (удаление) элемента из стека  {  if (myStk == NULL)  {  cout << "Стек пуст!" << endl;  return -1; //если стек пуст - возврат (-1)  }  else  {  Stack\* e = myStk; //е-переменная для хранения адреса элемента  int x = myStk->data; //запись элемента из поля data в перем. x  if (myStk)  myStk = myStk->next; //перенос вершины  delete e;  return x;  }  }  void toFile(Stack\*& myStk) //Запись в файл  {  Stack\* e = myStk;  Stack buf;  ofstream frm("mStack.txt");  if (frm.fail())  {  cout << "\n Ошибка открытия файла";  exit(1);  }  while (e)  {  buf = \*e;  cout << buf.data << ' ';  e = e->next;  }  frm.close();  cout << "Стек записан в файл mStack.txt\n";  }  void fromFile(Stack\*& myStk) //Считывание из файла  {  Stack buf, \* p = nullptr, \* e = nullptr;  ifstream frm("mStack.txt");  if (frm.fail())  {  cout << "\n Ошибка открытия файла";  exit(1);  }  frm.seekg(0);  frm >> buf.data;  while (!frm.eof())  {  push(buf.data, e);  frm >> buf.data;  }  frm.close();  while (e != NULL)  {  buf.data = pop(e);  push(buf.data, p);  myStk = p;  }  cout << "\nСтек считан из файла mStack.txt\n\n";  }  void show(Stack\*& myStk) //Вывод стека с очисткой  {  Stack\* e = myStk;  if (e == NULL)  cout << "Стек пуст!" << endl;  while (e != NULL)  cout << pop(e) << " ";  cout << endl;  }  void checkDiapason(Stack\*& myStack, int min, int max) {  Stack\* bufStack = NULL;  int k = 1;  int buf;  bool flag = false;  while (myStack != NULL) {  buf = pop(myStack);  push(buf, bufStack);  if (min <= buf && buf <= max) {  flag = true;  break;  }  k++;  }  while (bufStack != NULL) {  push(pop(bufStack), myStack);  }  if (flag) {  cout << "Элемент найден под номером: " << k << endl;  }  else {  cout << "Элемент не найден";  }  return;  }  void fuseStack(Stack\*& stack1, Stack\*& stack2, Stack\*& resultStack) {  Stack\* temp1 = NULL;  Stack\* temp2 = NULL;  Stack\* temp = NULL;  while (stack2 != NULL) {  int buf=pop(stack2);  push(buf, temp);  push(buf, temp2);  }  while (temp2 != NULL) {  push(pop(temp2), stack2);  }  while (stack1 != NULL) {  int value = pop(stack1);  push(value, temp1);  while (temp != NULL) {  int value2 = pop(temp);  if (value2 == value) {  push(value, resultStack);  break;  }  push(value2, temp2);  }  while (temp2 != NULL) {  push(pop(temp2), temp);  }  }  while (temp1 != NULL) {  push(pop(temp1), stack1);  }  }  void removeFirstDuplicate(Stack\*& myStk) {  if (!myStk || !myStk->next) return;  Stack\* tempStack = NULL;  int removedValue = -1;  bool found = false;    while (myStk != NULL && !found) {  int value = pop(myStk);  Stack\* temp = NULL;  while (tempStack != NULL) {  int value1 = pop(tempStack);  if (value1 == value) {  removedValue = value;  found = true;  break;  }  push(value1, temp);  }  while (temp != NULL) {  push(pop(temp), tempStack);  }  if (found) {  while (tempStack != NULL) {  push(pop(tempStack), myStk);  }  cout << "Удалён элемент: " << removedValue << endl;  return;  }  push(value, tempStack);  }  while (tempStack != NULL) {  push(pop(tempStack), myStk);  }  cout << "Повторяющихся элементов нет" << endl;  }  void remDupl(Stack\*& stack1,Stack\*& same, Stack\*& res){  Stack\* temp = NULL;  Stack\* buftemp = NULL;  Stack\* res1 = NULL;  while (stack1 != NULL) {  int buf = pop(stack1);  push(buf, temp);  push(buf, res1);  }  while (res1 != NULL) {  push(pop(res1), stack1);  }  while (same != NULL) {  int buf = pop(same);  push(buf, buftemp);  while (temp != NULL) {  int value1 = pop(temp);  if (value1 == buf) {  break;  }  push(value1, res1);  }  while (res1 != NULL) {  push(pop(res1), temp);  }  }  while (temp != NULL) {  push(pop(temp), res);  }  while (buftemp != NULL) {  push(pop(buftemp), same);  }  }  void delStacks(Stack\*& stack1,Stack\*& stack2, Stack\*& res) {  Stack\* same = NULL;  Stack\* res1 = NULL;  fuseStack(stack1, stack2, same);  remDupl(stack1, same, res1);  while (res1 != NULL) {  push(pop(res1), res);  }  remDupl(stack2, same, res1);  while (res1 != NULL) {  push(pop(res1), res);  }  }  void clear(Stack\*& myStk) {  while (myStk != NULL) {  pop(myStk);  }  return;  }  #include <iostream>  #include "myStack.h"  using namespace std;  int main()  {  setlocale(LC\_ALL, "Rus");  int choice; int x;  Stack\* myStk = NULL, \* stack1 = new Stack, \* stack2 = new Stack; //выделение памяти для стека  cout << "Выберите стек для заполнения (0/1)" << endl;  stack1 = NULL;  stack2 = NULL;  bool k;  cin >> k;  if (k) {  myStk = stack2;  }  else {  myStk = stack1;  }  for (;;) {  cout << "Выберите команду:" << endl;  cout << "0 - Выход" << endl;  cout << "1 - Добавление элемента в стек" << endl;  cout << "2 - Извлечение элемента из стека" << endl;  cout << "3 - Запись в файл" << endl;  cout << "4 - Чтение из файла" << endl;  cout << "5 - Вывод стека с очисткой" << endl;  cout << "6 - Найти элемент в заданном диапазоне" << endl;  cout << "7 - Поменять stack" << endl;  cout << "8 - Сформировать новый Stack из одинаковых элементов" << endl;  cout << "9 - Удалить первый повторяющийся элемент стека" << endl;  cout << "10 - Объеденить в стек без повторяющихся элементов" << endl;  cin >> choice;  switch (choice)  {  case 1: {  cout << "Введите элемент: " << endl;  cin >> x;  push(x, myStk); break;  }  case 2: {  x = pop(myStk);  if (x != -1)  cout << "Извлеченный элемент: " << x << endl;  break;  }  case 3: { toFile(myStk); break; }  case 4: { fromFile(myStk); break; }  case 5: {  cout << "Весь стек: " << endl;  show(myStk); break;  }  case 6: {  cout << "Введите диапазон от ";  int min, max;  cin >> min;  cout << "до ";  cin >> max;  checkDiapason(myStk, min, max);  break;  }  case 7: {  cout << "Выберите стек для заполнения (0/1)" << endl;  cin >> k;  if (k) {  stack1 = myStk;  myStk = stack2;  }  else {  stack2 = myStk;  myStk = stack1;  }  break;  }  case 8: {  Stack\* tempSteck = NULL ;  if (k) {  stack2 = myStk;  }  else {  stack1 = myStk;  }  fuseStack(stack1, stack2, tempSteck);  show(tempSteck);  break;  }  case 9: {  removeFirstDuplicate(myStk);  break;  }  case 10: {  Stack\* res = NULL;  if (k) {  stack2 = myStk;  }  else {  stack1 = myStk;  }  delStacks(stack1, stack2, res);  show(res);  }  case 0: return 0; break;  }  }  return 0;  } |