**МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ**  
Федеральное государственное бюджетное образовательное учреждение высшего образования  
**«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА (МИИТ)»  
 РУТ (МИИТ)**  
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
  
Кафедра: «Вычислительные системы и сети»  
  
  
  
  
  
  
  
  
  
  
**О Т Ч Ё Т  
П О Л А Б О Р А Т О Р Н О Й Р А Б О Т Е № 2**  
по дисциплине «Структуры и алгоритмы обработки данных»  
на тему: «Линейные списки при последовательном распределении памяти»  
  
  
  
  
  
  
  
  
  
 Выполнил: студент группы УИБ-311   
 Владыка П.А.

Москва 2018

**Задание**

Разработать и отладить программу, выполняющую обработку нескольких линейных списков в едином адресном пространстве при последовательном распределении памяти. Предусмотреть возможность вставки, удаления и просмотра элементов, используя меню выбора.

**Таблица имён**

|  |  |
| --- | --- |
| Исходные данные | |
| space | Адресное пространство (динамически создаваемый массив) |
| Промежуточные данные | |
| size | Размер создаваемого динамического массива |
| place | Массив индексов (границ) расположения структур |
| m | Неинициализированный элемент, используется для обнуления элемента массива |
| i | Счётчик |
| Save1, Save2 | «Ячейка» обмена |
| element | Промежуточное значение, вводимое с клавиатуры, которое присваивается элементу массива |

**Блок-схема**

**Главная программа Конструктор класса** LIST



**Исходный код программы**

//Created by Petya from UIB-311 14.10.2018 11:30

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<iostream>

#include<stdio.h>

#include<conio.h>

#include<windows.h>

#include <iomanip>

using namespace std;

class LIST{

public:

LIST() {

cout << "Введите размер адресного пространства: ";

(cin >> size).get();

space = new int[size];

Reset();}

~LIST() {

delete[] space;}

void menu() {

system("cls");

cout<< " {{>================================================<}}\n"

<< " || Списки ||\n"

<< " {{>================================================<}}\n"

<< " || 1. Выбрать \"Стек\" ||\n"//Стек будет расположен слева в простанстве и расширяться вправо

<< " || 2. Выбрать \"Очередь\" ||\n"//Очередь будет распологаться перед деком и расширяться влево (с перезаписью самой себя)

<< " || 3. Выбрать \"Дек\" ||\n"//Дек будет располагаться справа и расширяться влево (с перезаписью самого себя)

<< " || 4. Посмотреть память ||\n"

<< " || 5. Очистить память ||\n"

<< " || 6. Выход ||\n"

<< " {{>================================================<}}\n"

<< "\n Выбранная опция: ";

(cin >> choice[0]).get();

switch (choice[0]) {

case 49:

case 50:

case 51: {

Options();

break;}

case 52: {

Checker();

break;}

case 53: {

system("cls");

Reset();

cout << " Очищено!";

\_getch();

menu();

break;}

case 54: {

break;}

default: menu();}}

private:

int\* space; //Адресное пространство

int size, m;

int place[3][2]; //Размещение структур в адресном пространстве

char choice[2]; //Выбор номера опций меню/вторичного меню

void Options() {

system("cls");

cout<< " {{>================================================<}}\n"

<< " || 1. Внести элемент ||\n"

<< " || 2. Удалить элемент ||\n"

<< " || 3. Назад ||\n"

<< " {{>================================================<}}\n"

<< "\n Выбранная опция: ";

(cin >> choice[1]).get();

switch (choice[1]) {

case 49: {

switch (choice[0]) {

case 49: {

AddIntoStack();

break;}

case 50: {

AddIntoQueue();

break;}

case 51: {

char n;

do {

system("cls");

cout << " 1. Добавить справа\n"

<< " 2. Добавить слева\n"

<< " 3. Отмена\n"

<< "\n Выбранная опция: ";

(cin >> n).get();

} while ((n != 49) && (n != 50) && (n != 51));

switch (n) {

case 49: {

AddIntoDeck(1);

break;}

case 50: {

AddIntoDeck(0);

break;}

case 51: {

Options();

break;}}

break;}}

break;}

case 50: {

switch (choice[0]) {

case 49: {

DelFromStack();

break;}

case 50: {

DelFromQueue();

break;}

case 51: {

char n;

do {

system("cls");

cout << " 1. Удалить справа\n"

<< " 2. Удалить слева\n"

<< " 3. Отмена\n"

<< "\n Выбранная опция: ";

(cin >> n).get();

} while ((n != 49) && (n != 50) && (n != 51));

switch (n)

{

case 49: {

DelFromDeck(1);

break;}

case 50: {

DelFromDeck(0);

break;}

case 51: {

Options();

break;}}

break;}}

break;}

case 51: {

menu();

break;}

default: Options();}}

/\*Добавить в стек\*/

void AddIntoStack() {

int element;

system("cls");

if (place[0][1] + 1 < place[1][0]){

place[0][1]++;

cout << " Введите добавляемый элемент: ";

(cin >> element).get();

space[place[0][1]] = element;

cout << " Добавлено в стек!";}

else {

cout << "Нет места!";}

\_getch();

Options();}

/\*Удалить из стека\*/

void DelFromStack() {

system("cls");

if (place[0][1] != -1) { //Если есть стек не пустой

space[place[0][1]] = m;

place[0][1]--;

cout << " Удалено!";}

else {

cout << "В стеке пусто!";}

\_getch();

Options();}

/\*Добавить в очередь\*/

void AddIntoQueue() {

int element, save1, save2;

system("cls");

if (place[1][0] - 1 > place[0][1]) { //Если есть место слева

cout << " Введите добавляемый элемент: ";

(cin >> element).get();

if (place[1][0] > place[1][1]) {

place[1][0]--;

space[place[1][1]] = element; //Добавление в конец очереди

cout << " Добавлено в очередь!";}

else {

place[1][0]--;

save1 = space[place[1][1]]; //Сохраняем последний элемент в очереди

space[place[1][1]] = element; //Записываем на его место введённый элемент

for (int i = place[1][1]; i > place[1][0]; i--) { //Перезапись новой очереди

save2 = space[i - 1];

space[i - 1] = save1;

save1 = save2;}

cout << " Добавлено в очередь!";}}

else {

cout << "Нет места!";}

\_getch();

Options();}

/\*Удалить из очереди\*/

void DelFromQueue() {

system("cls");

if (place[1][0] <= place[1][1]) { //Если есть хотя бы 1 элемент

space[place[1][0]] = m;

place[1][0]++;

cout << " Удалено!";}

else {

cout << " В очереди пусто!"; }

\_getch();

Options();}

/\*Добавить в дек слева/справа\*/

void AddIntoDeck(int a) {

int element, save1, save2;

system("cls");

if (place[1][0] - 1 > place[0][1]) { //Если есть место слева от очереди

cout << " Введите элемент: ";

(cin >> element).get();

if (place[1][0] > place[1][1]) { //Если есть место слева от очереди

place[1][0]--;

place[1][1]--;}

else { //Сдвиг очереди влево на 1

place[1][0]--; //Сдвиг левой границы очереди влево на 1

save1 = space[place[1][1]]; //Сохраняем последний элемент в очереди

for (int i = place[1][1]; i > place[1][0]; i--) {

save2 = space[i - 1];

space[i - 1] = save1;

save1 = save2; }

place[1][1]--; //Сдвиг правой границы очереди влево на 1}

if (place[2][0] > place[2][1]) {

place[2][0]--;

space[place[2][a]] = element;//Добавление в левый/правый конец дека

cout << " Добавлено в дек!";}

else {

place[2][0]--;

save1 = space[place[2][a]]; //Сохраняем левый/правый элемент дека

space[place[2][a]] = element; //Записываем на его место введённый элемент

if (a == 1) { //Если добавляем справа

for (int i = place[2][1]; i > place[2][0]; i--) {

save2 = space[i - 1];

space[i - 1] = save1;

save1 = save2;}}

cout << " Добавлено в дек";}}

else {cout << "Нет места!"; }

\_getch();

Options();}

/\*Удалить из дека справа/слева\*/

void DelFromDeck(int a) {

system("cls");

if (place[2][0] <= place[2][1]) { //Если есть хотя бы 1 элемент в деке

if (a == 1) { //Если удаляем справа

for (int i = place[2][1]; i > place[2][0]; i--)

space[i] = space[i - 1];}

space[place[2][0]] = m; //Удаление слева

place[2][0]++; //Изменяем левую границу дека

if (place[1][0] > place[1][1]) {

place[1][0]++;

place[1][1]++;}

else { //Сдвиг очереди вправо на 1

place[1][1]++;//Сдвиг правой границы очереди вправо на 1

for (int i = place[1][1]; i > place[1][0]; i--) //Перезаписываем новую очередь

space[i] = space[i - 1];

space[place[1][0]] = m;

place[1][0]++;//Сдвиг левой границы очереди вправо на 1}

cout << " Удалено";}

else {cout << " В деке пусто!"; }

\_getch();

Options();}

/\*Проверить память\*/

void Checker() {

system("cls");

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

if (i % 20 == 0)

cout << endl;

cout << setw(4) << space[i] << " ";}

\_getch();

menu();}

/\*Сбросить память\*/

void Reset() {

for (int i = 0; i < size; i++)

space[i] = m;

//Размещение структур

place[0][1] = -1; //Размещение стека слева, занимает 0 ячеек

place[1][0] = size; //Размещение головы очереди

place[1][1] = size-1; //Размещение хвоста очереди

place[2][0] = size; //Размещение первого конца дека

place[2][1] = size-1; //Размещение второго конца дека}};

//Точка входа

int main(){

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

LIST \*inc = new LIST();

inc->menu();

delete inc;

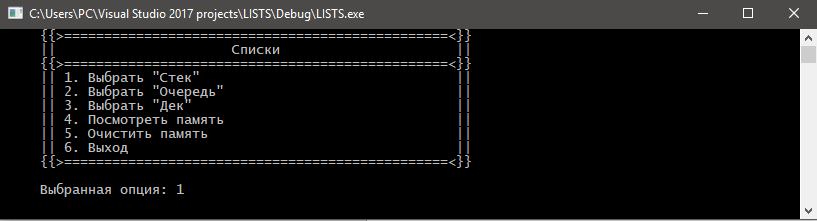
return 0;}

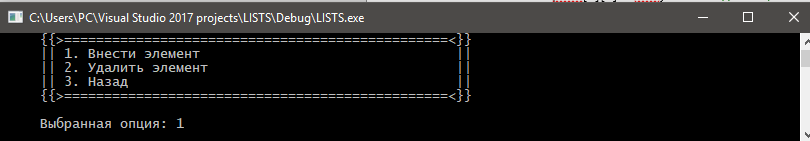
**Отладка**

Ввод размера адресного пространства:



Выбор стека из меню выбора и добавление 3 элементов подряд из подменю выбора:



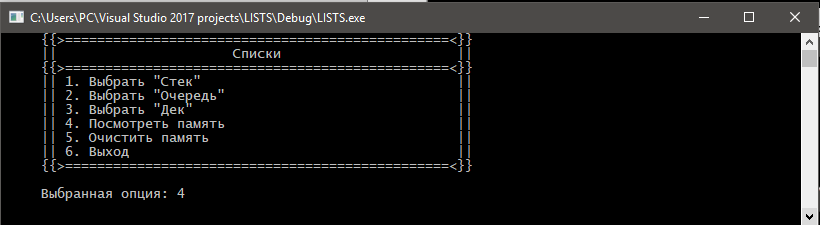








Просмотр добавленных элементов в адресном пространстве:





Выбор очереди и добавление 7-и элементов:







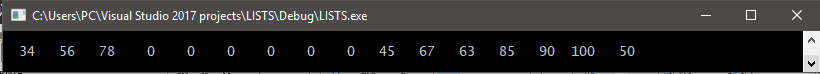




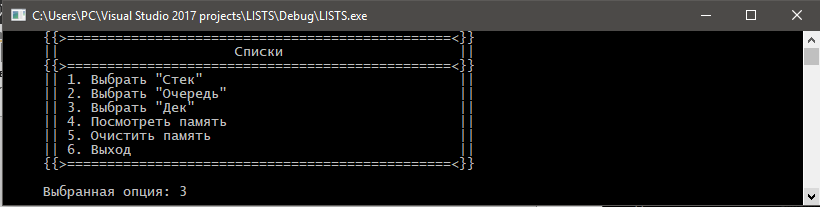


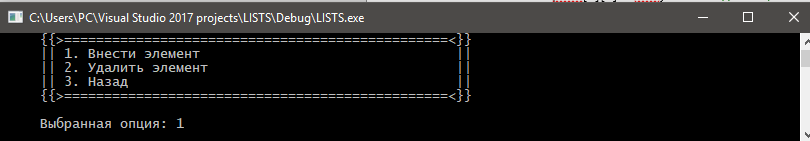


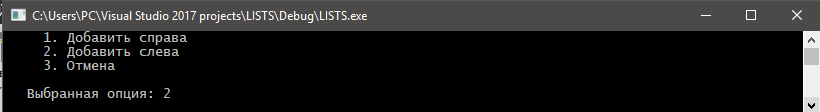
Просмотр добавленных элементов в адресном пространстве:



Выбор дека и добавление 3 элементов слева и 3 элементов справа:





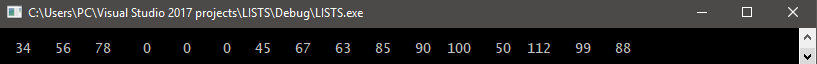


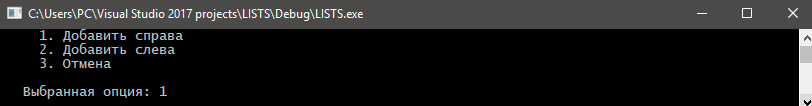






Просмотр адресного пространства:











Просмотр адресного пространства:



Попытка добавить в очередь:



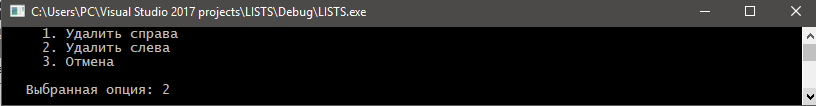
Попытка добавить стек:



Попытка добавить в дек слева и справа:



Удаление всех элементов дека слева:













Попытка удаления несуществующего элемента из очереди:



Очистка адресного пространства

