Министерство науки и высшего образования РФ

Севастопольский государственный университет

Кафедра информатики и управления в технических системах

ОТЧЕТ

по лабораторной работе №4

УПРАВЛЕНИЕ ДИНАМИЧЕСКИМ ВЫДЕЛЕНИЕМ ПАМЯТИ

по дисциплине «Операционные системы»

Выполнил:

Студент группы ИВТ/б 22-о

Черняев Н.Г.

Проверил:

Корепанова Н.Л.

г. Севастополь 2019

**Цель работы**

Изученить алгоритмы выделения памяти и программирование основных операций, выполняемых с линейными списковыми структурами.

**Вариант №4**

Задание:

Разработать подсистему динамического выделения памяти, язык программирования С. Для отладки разработать программу, в которой предусмотреть формирование списков свободной и выделенной памяти, имитацию запроса и выполнения заданной операции. Элемент списка должен содержать адрес первого байта блока памяти и размер блока. В случае, когда освобождающийся блок примыкает к соседнему блоку, он должен объединяться с ним в один блок. В программе обеспечить вывод исходных данных и результатов. Главная программа должна формировать меню, содержащее следующие пункты: ПОКАЗАТЬ СОСТОЯНИЕ ПАМЯТИ, ВЫДЕЛИТЬ ПАМЯТЬ, ОСВОБОДИТЬ ПАМЯТЬ, ВЫХОД.

Стратегия выделения памяти: первый подходящий.

Метод сортировки: сортировка обменом.

Тип списка: двунаправленный.

**Текст программы**

#include "pch.h"

#include <conio.h>

#include <math.h>

#include <stdio.h>

#include <iostream>

#include <stdlib.h>

#include <iomanip>

using namespace std;

struct Write {

int num;

struct Write \*next;

struct Write \*prev;

};

void Add(int x, Write \*MyList)

{

Write \*temp = new Write;

temp->num = x;

temp->next = MyList->prev;

MyList->prev = temp;

}

void Show(Write \*MyList) {

Write \*temp = MyList->prev;

while (temp != NULL) {

cout << temp->num << " ";

temp = temp->next;

}

cout << endl;

}

void Clear(Write \*MyList) {

while (MyList->prev != NULL) {

Write \*temp = MyList->prev->next;

delete MyList->prev;

MyList->prev = temp;

}

}

void sort(int n, Write \*MyList) {

Write \*x = new Write;

Write \*temp = MyList->prev;

for (int i = 1; i < n; i++) {

for (int j = n; j >= i; j--) {

if (temp->num > temp->next->num) {

x = temp->prev;

temp->prev = temp->next;

temp->next = x;

}

}

}

}

int main() {

Write \*obj = new Write;

obj->prev = NULL;

int k = 0,c;

cout << "1 - Add element" << endl << "2 - Delete elements" << endl << "3 - Show elemets" << endl;

cout << "4 - Show memory" << endl << "5 - Exit" << endl;

cin >> c;

while (c != 5) {

switch (c) {

case 1:

int x;

cout << "Add" << endl;

cin >> x;

Add(x, obj);

k++;

cout << "Choose from 1 to 6" << endl;

break;

case 2:

Clear(obj);

k = 0;

cout << "Choose from 1 to 6" << endl;

break;

case 3:

Show(obj);

cout << "Choose from 1 to 6" << endl;

break;

case 4:

cout << "Memory: " << k\*sizeof(obj) << endl;

cout << "Choose from 1 to 6" << endl;

break;

};

cin >> c;

}

delete obj->prev;

delete obj;

return 0;

}

**Тестовые примеры**

Добавим 2 элемента в список, выведем их на экран и проверим сколько памяти они занимают, затем удалим эти элементы и заново проверим сколько памяти занято.

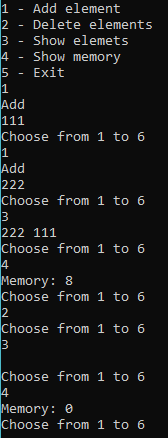


рис. 1 – результат работы программы

**Выводы**

В данной лабораторной работе были изучены алгоритмы выделения памяти и были получены навыки программирование основных операций, выполняемых с линейными списковыми структурами.