МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Навчально науковий інститут комп’ютерних інформаційних технологій

Кафедра інженерії програмного забезпечення

Лабораторна робота №1.9.

з дисципліни «Основи програмування»

на тему: «ДОСЛІДЖЕННЯ БЛОЧНОЇ ОРГАНІЗАЦІЇ ПРОГРАМ ТА

СТРАТЕГІЙ РОЗПОДІЛУ ПАМ'ЯТІ»

Виконав: студент групи ІКІТ – 114

Бойко А. М.

Прийняла: Гарнець А.А.

Київ – 2016

1. Мета: вивчити блочну організацію програм та стратегії розподілу пам'яті.
2. Завдання: Написати програми, які складаються з наступних дій:
3. Вивчити блочну структуру програм. Для цього:

а) написати програму, що складається з головного блоку main, декількох вкладених в *main* блоків, що є динамічно розподіленими по відношенню один до іншого, та ще декількох вкладених один в другий блоків.

2. Дослідити статичний розподіл пам’яті. Для цього:

а) додати в програму опис статичних змінних у зовнішньому (поза блоком *main*) та у вкладених блоках та ініціалізувати змінні при створенні та за допомогою привласнення;

б) додати в різні блоки оператори привласнення, в яких використовуються описані змінні;

в) виконати програму по кроках та дослідити у вікні Watch, як будуть змінюватись значення імені та вмісту змінних (&ідентифікатор, ідентифікатор);

г) побудувати та виконати програму (використовуючи вікно Watch) таким чином, щоб продемонструвати всі особливості статичного розподілу пам’яті, описані в лекціях.

3. Дослідити динамічний розподіл пам’яті в блоці. Для цього:

а) додати в програму опис динамічних в блоці змінних у головному (*main*) та вкладених блоках та ініціалізувати змінні при створенні та за допомогою привласнення;

б) додати в різні блоки оператори привласнення, в яких використовуються описані змінні;

в) дослідити, як будуть змінюватись значення імені та вмісту змінних;

г) описати у вкладених блоках та використати в операторах привласнення змінні з однаковим позначенням і дослідити мембранний ефект, аналізуючи значення імен та вмісту змінних при виконанні програми;

д) описати у динамічно розподілених блоках та використати в операторах привласнення змінні з однаковим позначенням, дослідити та пояснити зміну значеннь імен та вмісту змінних при виконанні програми;

є) додати в програму наступний код:

// оператор, що дозволяє увійти в блок декілька разів

for (int i = 0; i < 5; i = i + 1)

{

static int iA = 0;

auto int iB = 0;

iA = iA + 1;

iB = iB + 1;

}

дослідити у вікні Watch та пояснити поведінку статичної та динамічної змінних;

ж) побудувати та виконати програму (використовуючи вікно Watch) таким чином, щоб продемонструвати всі особливості динамічного розподілу пам’яті в блоці, описані в лекціях.

4. Додати в програму приклад застосування операції глобального дозволу видимості імен.

5. Дослідити динамічний розподіл пам’яті в «купі». Для цього:

а) додати в програму опис першої типізованої вказівної змінної, захопити пам'ять в «купі», занести в «купу» значення та звільнити пам'ять (див. приклад);

б) виконати нижче опис дій п. 5.а по відношенню до другої вказівної змінної;

в) дослідити у вікні Watch значення імені, вмісту та розіменованого вмісту покажчиків ((&ідентифікатор, ідентифікатор, \*ідентифікатор);

г) звільняти і не звільняти пам'ять у «купі» та пояснити зміни у вікні Watch;

д) виконати дії п. 5.а з одним покажчиком у вкладеному блоці, а з іншим – нижче, поза вкладеним. Звільняти і не звільняти пам'ять у «купі», захоплену першим, локальним покажчиком, та за допомогою Watch пояснити, як утворюється «сміття» в «купі»;

є) навести в програмі інші приклади некоректних дій з покажчиками,що призводять до утворення «сміття».

1. Виконання:

Створив проект та записав туди наступний код:

#include "stdafx.h"

static int si = 9523;

int main()

{

static int si1 = 493;

float f1 = 34,

f2 = 566,

f3 = 99;

{

float f1 = 57;

f2 = 78;

}

f3 = 8;

{

f2 = 89;

{

static int si2 = 93;

float f4 = 7;

{

si2 = 87;

si = 76;

f3 = 5;

float f4 = 4;

si = 76;

}

si2 = 13;

}

si = 43;

}

for (int i = 0; i < 5; i = i + 1)

{

static int iA = 0;

int iB = 0;

iA = iA + 1;

iB = iB + 1;

}

int \*pI;

pI = new int;

\*pI = 25;

delete pI;

{

pI = new int;

\*pI = 25;

delete pI;

}

for (int i = 0; i < 20; i++)

{

int \*pi1;

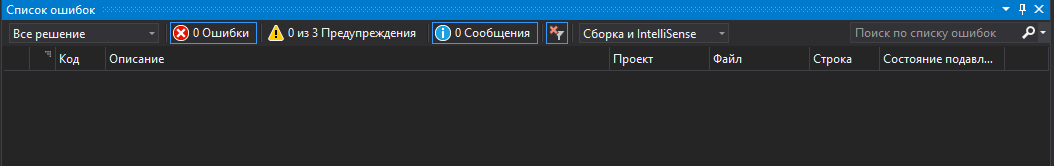
pi1 = new int;

}

return 0;

}

При відлагоджені та компіляції помилок не виявлено:



1. Висновок: ми вивчили блочну організацію програм та стратегії розподілу пам'яті.