Міністерство освіти і науки, молоді та спорту України

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Алгоритми та структури даних

ЗВІТ ДО

КОМП’ЮТЕРНОГО ПРАКТИКУМУ №3

“ Робота з динамічними змінними. Операції з показниками”

Варіант № 1

Оцінка «\_\_\_\_\_\_\_\_\_»

Дата «\_\_\_» \_\_\_\_\_\_\_ 20\_\_\_

Виконав: студент 2 курсу

гр. ТВ-61

Артамонов Олексій Юрійович

Перевірила: Васильєва О.Б.

Київ – 2017

**Завдання:** Утворити масив показників на рядки, розробити процедуру, яка вилучає і-й рядок. Для перевірки вивести масив показників.

**Лістинг програми**

// алгоритмы.cpp: определяет точку входа для консольного приложения.

//

#include "stdafx.h"

#include "ctime"

#include "iostream"

#include "conio.h"

using namespace std;

class Complex {

private:

int real;

int imag;

public:

Complex() {}

Complex(int a, int b) {

this->real = a;

this->imag = b;

cout << "(" << this->real << ", " << this->imag << ") ";

}

Complex(Complex \*\*arr, int j) {

cout << "(" << arr[j]->real << ", " << arr[j]->imag << ") ";

}

void Sqrt(double \*arr1[], Complex \*\*arr, int i) {

\*arr1[i] = sqrt(arr[i]->real \* arr[i]->real + arr[i]->imag \* arr[i]->imag);

}

};

void Zap(Complex x, int n, Complex \*\*arr, double \*\*arr1) {

int z, y;

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

z = rand() % 10;

y = rand() % 10;

\*arr[i] = Complex(z, y);

x.Sqrt(arr1, arr, i);

}

}

void Max(Complex x, Complex \*\*arr, double \*\*arr1, int n) {

double \*max = arr1[0];

int j;

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

if (arr1[i] > max) {

max = arr1[i];

j = i;

}

}

cout << "\n\nMax = ";

Complex(arr, j);

cout << "\nMax(abs) = " << \*max << endl;

}

void Delete(Complex \*\*arr, double \*\*arr1, int n) {

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

{

delete arr[i];

delete arr1[i];

}

}

int main()

{

srand(time(NULL));

Complex x;

const int n = 10;

int v = 0;

do {

printf("\n\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_%d\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n\n", v);

double \*\*arr1 = new double\*[n];

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

arr1[i] = new double;

}

Complex \*\*arr = new Complex\*[n];

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

arr[i] = new Complex;

}

cout << "Array: ";

Zap(x, n, arr, arr1);

Max(x, arr, arr1, n);

Delete(arr, arr1, n);

printf("\n\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n\n");

printf("\n\nPress \"Enter\"\n");

v++;

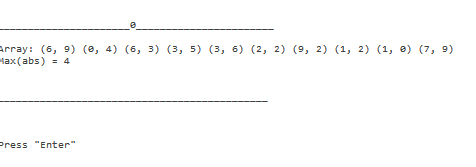
//\_getch();

} while (getchar() == '\n');

return 0;

}

**Результати:**



**Контрольні питання:**

Вказівник

1.Вказівник не є самостійним типом, він завжди пов'язаний з будь-яким іншим конкретним типом об'єкта.

2.Вказівник на void застосовується в тих випадках, коли конкретний тип об'єкта, адреса якого потрібно зберігати, не визначений (наприклад, якщо в одній і тій же змінній в різні моменти часу потрібно зберігати адреси об'єктів різних типів).

3.Присвоювання без явного приведення типів допускається вказівниками типу void \*:

4.Значення вказівника змінюється на величину sizeof (тип).

5.Вказівник займає лише цілу кількість байтів.

6. Якщо ми створюємо вказівник-константу, то змінити його значення вже неможливо, в той час як вказівник на константу може вказувати на іншу змінну або константу, якщо ми дамо йому їх адресу.

7. При розіменуванні ми отримуємо різні результати, при взятті адреси - одне і те ж.

8.Адреса розіменування вказівника.

9.При розіменуванні можна дізнатися скільки байтів займає тип вказівника, стільки байтів пам'яті і доступно.

Динамічна пам'ять

1. Це дозволяє по ходу роботи програми контролювати і коригувати обсяг використовуваної пам'яті і, отже, створювати програми здатні обробляти великі обсяги даних, обходячи обмеженість фізичної пам'яті машини.

2. Пам'ять під динамічні дані виробиться в процесі виконання програми.

3.Дані в динамічної пам'яті зберігаються, поки пам'ять під них не буде звільнена функцією наприклад delete.

4.Доступ до динамічної пам'яті можливий тільки через вказівники, тобто не можна створювати в неї змінні, але можна виділяти її фрагменти і пов'язувати з з деяким покажчиком.

5.Місце на область в пам'яті.

6. Якщо new не може виділити пам'ять в купі, то він передасть (throw) виключення типу std :: bad\_alloc.

7.void \* - узагальнений тип вказівника, тобто не прив'язаний до якогось конкретного типу.

8. Якщо не звільняє динамічну пам'ять, то вона буде зайнята до завершення програми, що є неприйнятним.

9.Ми можемо звільнити тільки ту пам'ять, яку ми зарезервували.