**ЛАБОРАТОРНА РОБОТА № 22**

**Покажчики**

**Мета:** набути навичок роботи з покажчиками.

**Хід роботи:**

**Завдання 1.** Написати програму з використанням покажчиків.



Лістинг программи:

1. По вашим поясненням:

#include <iostream>

#include <math.h>

#include <windows.h>

#include <time.h>

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

srand(time(NULL));

// Тема завдання

printf("Завдання 1:\nВивести адреси елементів масиву 4х4,\n що знаходяться нижче головної діагоналі.\n");

int choise, count, i0, j0, a, b;

int\*\* arr = NULL; int\*\* arr1 = NULL;

printf("Почати програму(1) / Завершити програму(0):"); scanf\_s("%d", &choise);

do {

// Введення даних

printf("Введіть кількість рядочків:"); scanf\_s("%d", &i0);

printf("Введіть кількість стовпчиків:"); scanf\_s("%d", &j0);

printf("Введіть початкове значення масиву a (ціле):"); scanf\_s("%d", &a);

printf("Введіть кінцеве значення масиву b (ціле):"); scanf\_s("%d", &b);

//Виведення початкового двовимірного масиву:

printf("Двовимірний масив за допомогою покажчиків:\n");

arr = (int\*\*)malloc(j0 \* sizeof(int)); // виділення памяті для стовпчиків

for (int i = 0; i < j0; i++) { // цикл для виділення памяті рядочків

\*(arr+i) = (int\*)malloc(i0 \* sizeof(int)); // виділення памяті для рядочків

}

for (int i = 0; i < i0; i++) {

for (int j = 0; j < j0; j++) {

\*(\*(arr + i) + j) = a + rand() % (b - a + 1);

printf("%3d", \*(\*(arr + i) + j));

}

printf("\n");

}

printf("\nАдреси елементів головної діагоналі:\n");

for (int i = 0; i < i0; i++) {

for (int j = 0; j < j0; j++) {

if (i == j) {

printf("%-10d", &(\*(\*(arr + i) + j))); // вивід адрес головної діагоналі 2--овимірного масиву

}

else {

printf("%-10d", \*(\*(arr + i) + j));

}

}

printf("\n");

}

printf("\nПродовжити програму(1) / Завершити програму(0):"); scanf\_s("%d", &choise);

} while (choise);

}

1. За методичкою:

#include <iostream>

#include <math.h>

#include <windows.h>

#include <time.h>

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

srand(time(NULL));

// Тема завдання

printf("Завдання 1:\nВивести адреси елементів масиву 4х4,\n що знаходяться нижче головної діагоналі.\n");

int choise, count, i0, j0, a, b;

int\* arr = NULL; int\* arr1 = NULL;

printf("Почати програму(1) / Завершити програму(0):"); scanf\_s("%d", &choise);

do {

// Введення даних

printf("Введіть кількість рядочків:"); scanf\_s("%d", &i0);

printf("Введіть кількість стовпчиків:"); scanf\_s("%d", &j0);

printf("Введіть початкове значення масиву a (ціле):"); scanf\_s("%d", &a);

printf("Введіть кінцеве значення масиву b (ціле):"); scanf\_s("%d", &b);

//Виведення початкового двовимірного масиву:

printf("Двовимірний масив за допомогою покажчиків:\n");

arr = (int\*)malloc(i0\*j0 \* sizeof(int)); // виділення памяті для стовпчиків

for (int i = 0; i < i0; i++) {

for (int j = 0; j < j0; j++) {

\*(arr + i \* j0 + j) = a + rand() % (b - a + 1);

printf("%3d", \*(arr + i \* j0 + j));

}

printf("\n");

}

printf("\nАдреси елементів головної діагоналі:\n");

for (int i = 0; i < i0; i++) {

for (int j = 0; j < j0; j++) {

if (i == j) {

printf("%-10d", &arr + i \* j0 + j); // вивід адрес головної діагоналі 2--овимірного масиву

}

else {

printf("%-10d", \*(arr + i \* j0 + j));

}

}

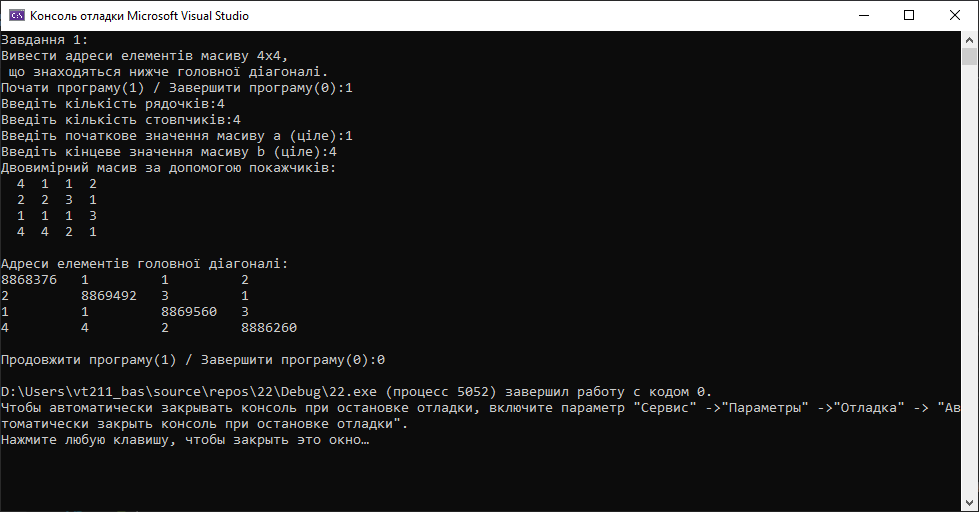
printf("\n");

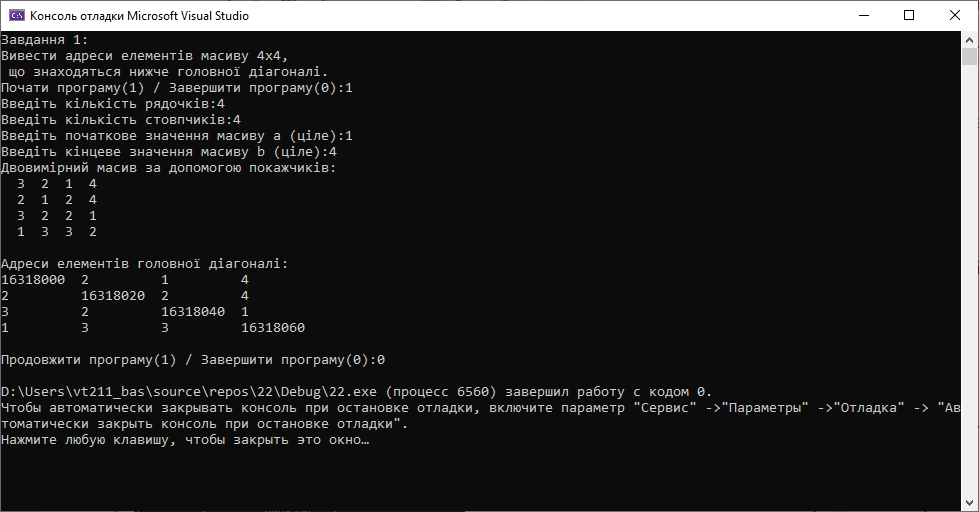
}

printf("\nПродовжити програму(1) / Завершити програму(0):"); scanf\_s("%d", &choise);

} while (choise);

Результат програми:





**Завдання 2.** Написати програму, в якій потрібно створити двовимірний динамічний масив і виконати обробку даних.



Лістинг программи:

#include <iostream>

#include <math.h>

#include <windows.h>

#include <time.h>

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

srand(time(NULL));

// Тема завдання

printf("Завдання 2:\nВидалити із масиву рядок із заданим номером\nі стовпець, де заходиться максимальний елемент.\n");

int choise, count, i0, j0, a, b, i0k, j0k, max = -INFINITY;

int\*\* arr = NULL;

printf("Почати програму(1) / Завершити програму(0):"); scanf\_s("%d", &choise);

do {

// Введення даних

printf("Введіть кількість рядочків:"); scanf\_s("%d", &i0);

printf("Введіть кількість стовпчиків:"); scanf\_s("%d", &j0);

printf("Введіть початкове значення масиву a (ціле):"); scanf\_s("%d", &a);

printf("Введіть кінцеве значення масиву b (ціле):"); scanf\_s("%d", &b);

printf("Введіть індекс рядка k, який хочете видалити:"); scanf\_s("%d", &i0k);

//Виведення початкового двовимірного масиву:

printf("Двовимірний масив за допомогою покажчиків:\n");

arr = (int\*\*)malloc(sizeof(int) \* i0); // виділення памяті для двовимірного масиву для рядків та стовпців

for (int i = 0; i < i0; i++) { // цикл для виділення памяті для двовимірного масиву для стовпців

\*(arr + i) = (int\*)malloc(sizeof(int) \* j0);

}

for (int i = 0; i < i0; i++) {

for (int j = 0; j < j0; j++) {

\*(\*(arr + i) + j) = a + rand() % (b - a + 1);

printf("%5d", \*(\*(arr + i) + j));

}

printf("\n");

}

for (int i = 0; i < i0; i++) {

for (int j = 0; j < j0; j++) {

if (i >= i0k && i != i0 - 1) {

\*(\*(arr + i) + j) = \*(\*(arr + i + 1) + j);

}

}

}

i0--;

printf("Двовимірний масив за допомогою покажчиків після видалення рядочка за номером k:\n");

arr = (int\*\*)realloc(arr, sizeof(int) \* i0 \* j0); // перевизначення потрібної памяті для масиву

for (int i = 0; i < i0; i++) {

for (int j = 0; j < j0; j++) {

printf("%5d", \*(\*(arr + i) + j));

}

printf("\n");

}

for (int i = 0; i < i0; i++) {

for (int j = 0; j < j0; j++) {

if (\*(\*(arr + i) + j) > max) {

max = \*(\*(arr + i) + j);

j0k = j;

}

}

}

for (int i = 0; i < i0; i++) {

for (int j = 0; j < j0; j++) {

if (j >= j0k && j != j0 - 1) {

\*(\*(arr + i) + j) = \*(\*(arr + i) + j + 1);

}

}

}

j0--;

printf("\nДвовимірний масив за допомогою покажчиків після видалення стовпчика за максимальним елементом у масиві:\n");

arr = (int\*\*)realloc(arr, sizeof(int) \* i0 \* j0); // перевизначення потрібної памяті для масиву

for (int i = 0; i < i0; i++) {

for (int j = 0; j < j0; j++) {

printf("%5d", \*(\*(arr + i) + j));

}

printf("\n");

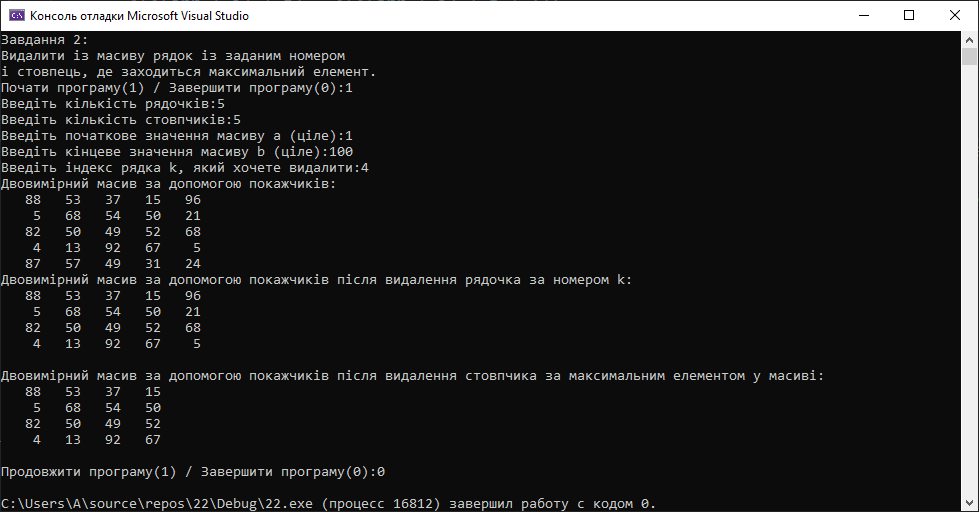
}

printf("\nПродовжити програму(1) / Завершити програму(0):"); scanf\_s("%d", &choise);

} while (choise);

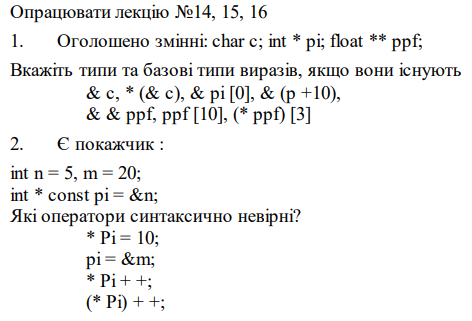
}

Результат програми:

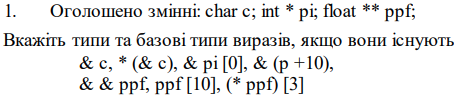


**Самостійна робота**

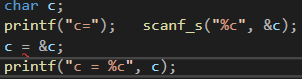
**Завдання.**

****

**Завдання 1.**

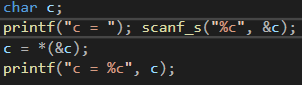
****

1. &c – не можливо присвоїти значення адреса не змінній «покажчик». Якщо ж взяти змінну покажчика, то можна присвоїти значення адреса цієї змінної до цієї змінної.



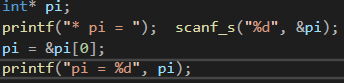


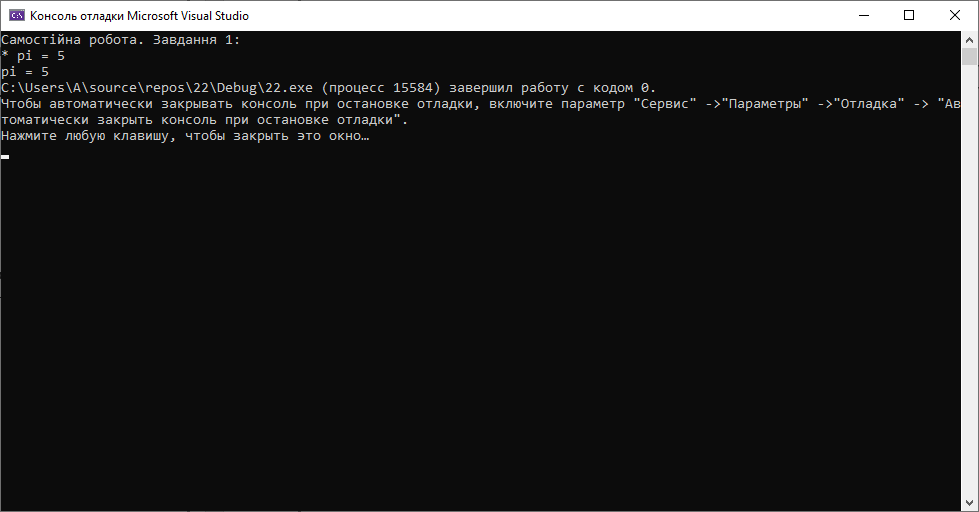
1. \*(&c) – так можна присвоїти значення с указавши що це показчик та передати змінну через адрес покажчика , тобто якщо ввести будь яку букву чи символ, то до змінної с присвоється значення цієї змінної через її адрес. Такий вираз існує.





1. & pi[0] – так, можна у pi передати значення через його адрес, вказавши що це елемент [0] – це вказує, що це перший елемент, а отже його значення і є першим елементом, навіть якщо це не масив. Такий вираз існує.





1. & (p+10) – такої змінної, як p не вказано в оголошенні, тому адрес цієй змінної взяти не можна. Якщо навіть замість р вказати рі ( pi = &(pi + 10); ), то не вдасться взяти адрес таким шляхом. Такий вираз не існує.



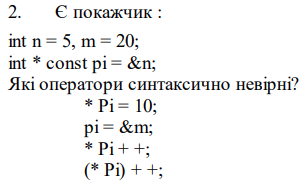


1. & & ppf – в такому ввипадку взяття адреса адресу компілятор буде сваритися:

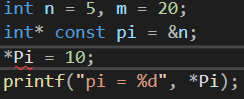
  , тому такий вираз не існує.

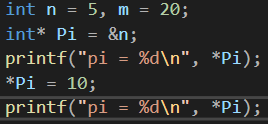
1. ppf[10] – в такому випадку, якщо масив не ініціалізований чи не зарандомлений, то неможливо присвоїти 10 елемент масиву ppf:  Але , якщо об’явити його таким способом:  це зробити можливо, але знову ж таки, якщо масив ініціалізований. Такий вираз не існує.
2. (\*ppf)[3] – знову ж таки, якщо масив не ініціалізований чи не зарандомлений, то неможливо присвоїти 3 елемент масиву ppf через покажчик, лише об’явивши його таким способом:  і ініціалізувати масив. Такий вираз не існує.

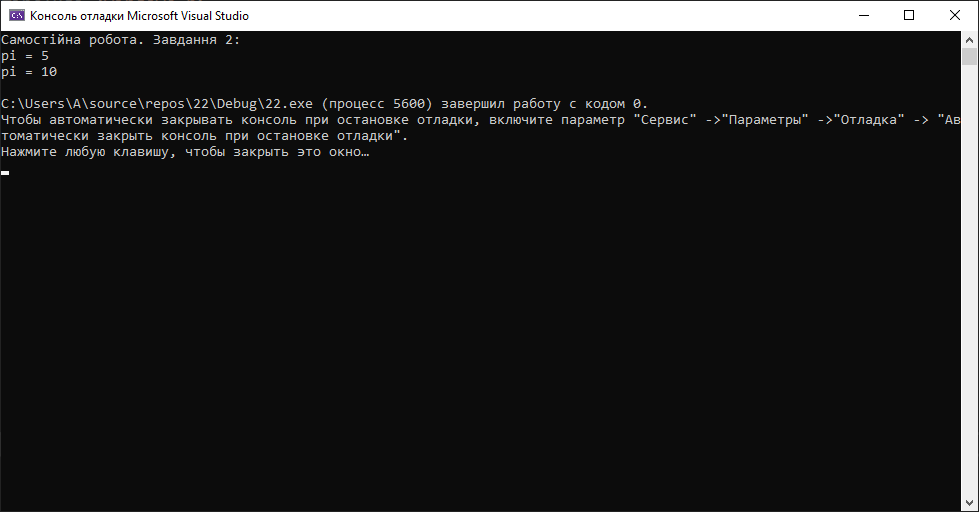
**Завдання 2.**

****

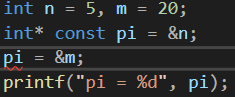
1. \* Pi = 10; – такий оператор синтаксично не вірний у випадку завдання, бо покажчик з іменем Pi не ініціалізований. Лише якщо змінити назву оператора з Pi на pi він буде працювати. Але якщо загалом, то такий оператор можливий – він присвоїть покажчику Pi значення 10.

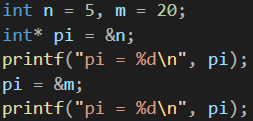


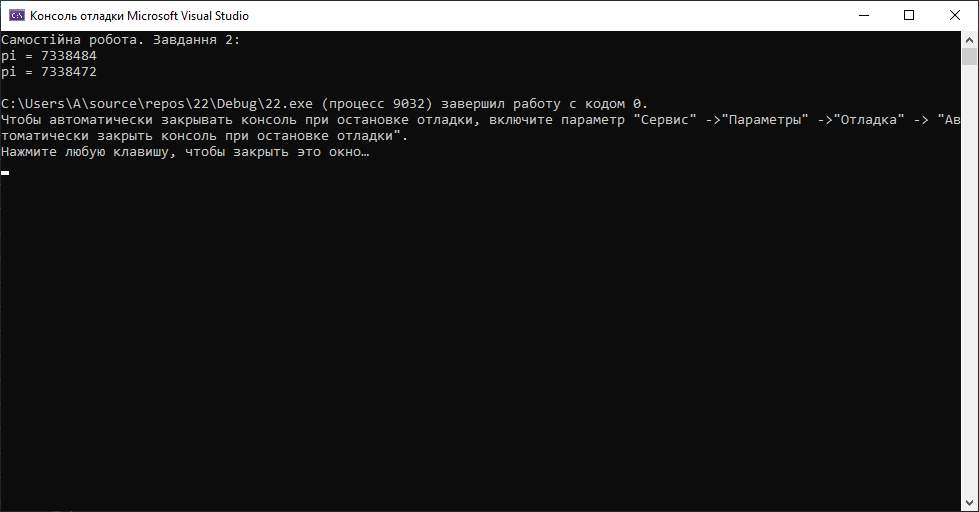




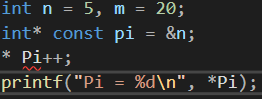
1. pi = &m; – такий оператор синтаксично не вірний у випадку завдання, бо покажчик є константою і він змінюватись не може. Лише в тому випадку, якщо прибрати const цей оператор буде працювати. Але якщо загалом, то такий оператор можливий – він покаже адрес змінної m.

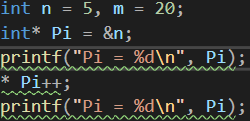


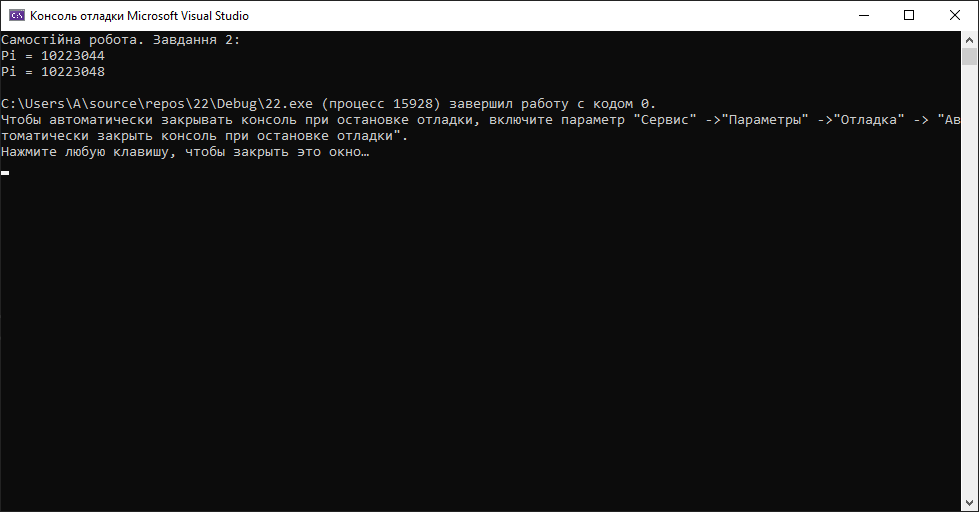




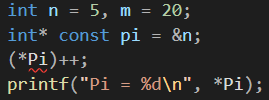
1. \* Pi++; - такий оператор також є синтаксично вірним у випадку завдання, бо покажчик з іменем Pi не ініціалізований. Але якщо загалом, то такий оператор можливий - він збільшує адрес змінної на 4(тип даних int).

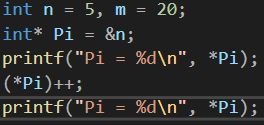


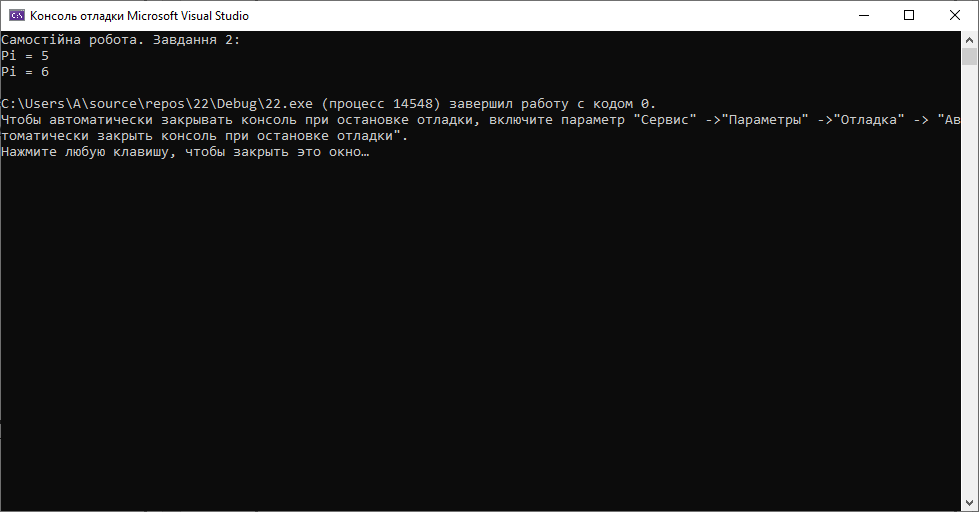




1. (\* Pi)++; - такий оператор також є синтаксично вірним у випадку завдання, бо покажчик з іменем Pi не ініціалізований. Але якщо загалом, то такий оператор можливий – він збільшує значення на 1, тобто якщо int\* Pi = &n; а n = 5, то n стане 6(інкримент збільшість значення n на 1).







***Висновки:*** в ході виконання лабораторної роботи було ознайомлено з середовищем MS Visual Studio. Досліджено та отримано практичні навики щодо створення програм.