3 варіант

1. **Об'є́ктно-орієнто́ване програмува́ння** (ООП) — одна з [парадигм](https://uk.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%B4%D0%B8%D0%B3%D0%BC%D0%B8_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) [програмування](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F), яка розглядає програму як множину «об'єктів», що взаємодіють між собою. Основу ООП складають три основні концепції: [інкапсуляція](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D0%BA%D0%B0%D0%BF%D1%81%D1%83%D0%BB%D1%8F%D1%86%D1%96%D1%8F), [успадкування](https://uk.wikipedia.org/wiki/%D0%A3%D1%81%D0%BF%D0%B0%D0%B4%D0%BA%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)) та [поліморфізм](https://uk.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D1%96%D0%BC%D0%BE%D1%80%D1%84%D1%96%D0%B7%D0%BC_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)). Одною з переваг ООП є краща [модульність](https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D1%8C_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)) програмного забезпечення (тисячу функцій процедурної мови, в ООП можна замінити кількома десятками класів із своїми методами). Попри те, що ця парадигма з'явилась в [1960-тих](https://uk.wikipedia.org/wiki/1960-%D1%82%D1%96) роках, вона не мала широкого застосування до [1990-тих](https://uk.wikipedia.org/wiki/1990-%D1%82%D1%96), коли розвиток комп'ютерів та комп'ютерних мереж дозволив писати надзвичайно об'ємне і складне програмне забезпечення, що змусило переглянути підходи до написання програм. Сьогодні багато [мов програмування](https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) або підтримують ООП ([PHP](https://uk.wikipedia.org/wiki/PHP), [Lua](https://uk.wikipedia.org/wiki/Lua)) або ж є цілком об'єктно-орієнтованими (зокрема, [Java](https://uk.wikipedia.org/wiki/Java_(%D0%BC%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)), [C#](https://uk.wikipedia.org/wiki/C_Sharp_(%D0%BC%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)), [C++](https://uk.wikipedia.org/wiki/C%2B%2B), [Python](https://uk.wikipedia.org/wiki/Python), [Ruby](https://uk.wikipedia.org/wiki/Ruby) та [Objective-C](https://uk.wikipedia.org/wiki/Objective-C), [ActionScript 3](https://uk.wikipedia.org/wiki/ActionScript), [Swift](https://uk.wikipedia.org/wiki/Swift_(%D0%BC%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)), [Vala](https://uk.wikipedia.org/wiki/Vala)).
2. abstract class A {  
       int p1;  
       A() {  
           p1 = 1;  
       }  
       void print() {  
           System.out.println(p1);  
       }  
   }  
   class B extends A {  
   }  
   public class Main {  
       public static void main(String[] args) {  
           A ob1;  
           // ошибка: ob1 = new A();  
           B ob2 = new B(); // будет вызван конструктов по умолчанию из A  
           ob2.print();  
       }  
   }

Пасивна поведінка – дія, яку виконує об’єкт під впливом іншого об’єкту.

Активна поведінка – дія, яку об’єкт виконує над іншим об’єктом.

**3)Написати перевантажувані функції, що обчислюють середнє масиву чисел типу float або double. Продемонструвати виконання.**

#include <stdio.h>

double func(float\* arr, int length);

double func(double\* arr, int length);

int main(void)

{

static double arr1[] = { -2,4,6,8,9,5,3 };

float arr2[] = { 2.3,4.1,4.6 };

printf("Function with float is:%0.1f\n", func(arr2, 3));

printf("Function with double is:%0.1f\n", func(arr1, 7));

return 0;

}

double func(float\* arr, int length)

{

float sum = 0;

double result = 0;

if (length > 0)

{

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

sum += arr[i];

result = sum / length;

}

return result;

if (length <= 0)

return 0;

}

double func(double\* arr, int length)

{

double sum = 0;

double result = 0;

if (length > 0)

{

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

sum += arr[i];

result = sum / length;

}

return result;

if (length <= 0)

return 0;

}

4) Написати програму, що виводить у стандартний потік масив чисел типу int по 4 значення в рядку з вирівненням за стовпцями.

#include <iostream>

#include <iomanip>

using namespace std;

int main(int argc, char\* argv[]) {

int n;

cout << "Enter size:";

cin >> n;

int \*a = new int[n];

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

cin >> a[i];

}

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

cout.width(4);

cout <<" " << a[i] <<" " << setw(12);

if (i % 4 == 0)

cout << "\n";

}

cout << endl;

delete[] a;

}

4 варіант

1. **Об'є́ктно-орієнто́ване програмува́ння** (ООП) — одна з [парадигм](https://uk.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%B4%D0%B8%D0%B3%D0%BC%D0%B8_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) [програмування](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F), яка розглядає програму як множину «об'єктів», що взаємодіють між собою. Основу ООП складають три основні концепції: [інкапсуляція](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D0%BA%D0%B0%D0%BF%D1%81%D1%83%D0%BB%D1%8F%D1%86%D1%96%D1%8F), [успадкування](https://uk.wikipedia.org/wiki/%D0%A3%D1%81%D0%BF%D0%B0%D0%B4%D0%BA%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)) та [поліморфізм](https://uk.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D1%96%D0%BC%D0%BE%D1%80%D1%84%D1%96%D0%B7%D0%BC_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)). Одною з переваг ООП є краща [модульність](https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D1%8C_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)) програмного забезпечення (тисячу функцій процедурної мови, в ООП можна замінити кількома десятками класів із своїми методами). Попри те, що ця парадигма з'явилась в [1960-тих](https://uk.wikipedia.org/wiki/1960-%D1%82%D1%96) роках, вона не мала широкого застосування до [1990-тих](https://uk.wikipedia.org/wiki/1990-%D1%82%D1%96), коли розвиток комп'ютерів та комп'ютерних мереж дозволив писати надзвичайно об'ємне і складне програмне забезпечення, що змусило переглянути підходи до написання програм. Сьогодні багато [мов програмування](https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) або підтримують ООП ([PHP](https://uk.wikipedia.org/wiki/PHP), [Lua](https://uk.wikipedia.org/wiki/Lua)) або ж є цілком об'єктно-орієнтованими (зокрема, [Java](https://uk.wikipedia.org/wiki/Java_(%D0%BC%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)), [C#](https://uk.wikipedia.org/wiki/C_Sharp_(%D0%BC%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)), [C++](https://uk.wikipedia.org/wiki/C%2B%2B), [Python](https://uk.wikipedia.org/wiki/Python), [Ruby](https://uk.wikipedia.org/wiki/Ruby) та [Objective-C](https://uk.wikipedia.org/wiki/Objective-C), [ActionScript 3](https://uk.wikipedia.org/wiki/ActionScript), [Swift](https://uk.wikipedia.org/wiki/Swift_(%D0%BC%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)), [Vala](https://uk.wikipedia.org/wiki/Vala)).
2. **Інкапсуляція** — один з трьох основних механізмів [об'єктно-орієнтованого програмування](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F). Йдеться про те, що об’єкт вміщує не тільки дані, але і правила їх обробки, оформлені в вигляді виконуваних фрагментів ([методів](https://uk.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F))). А також про те, що доступ до стану об'єкта напряму заборонено, і ззовні з ним можна взаємодіяти виключно через заданий інтерфейс (відкриті поля та методи), що дозволяє знизити [зв'язність](https://uk.wikipedia.org/wiki/%D0%97%D0%B2%27%D1%8F%D0%B7%D0%BD%D1%96%D1%81%D1%82%D1%8C_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)). Таким чином контролюються звернення до полів класів та їхня правильна ініціалізація, усуваються можливі помилки пов'язані з неправильним викликом методу. Оскільки користувачі працюють лише через відкриті елементи класів, то розробники класу можуть як-завгодно змінювати всі закриті елементи і навіть перейменовувати та видаляти їх, не турбуючись, що десь хтось їх використовує у своїх програмах.

**3) Написати перевантаження функції, що обчислюють середнє масиву чисел типу int або double. Продемонструвати їх використання.**

#include <iostream>

using namespace std;

double func(int\* arr, int length);

double func(double\* arr, int length);

int main(void)

{

static double arr1[] = { -2,4,6,8,9,5,3 };

int arr2[] = { 2.3,4.1,4.6 };

cout << "Function with int is:" << func(arr1, 7) << endl;

cout << "Function with double is:" << func(arr2, 3) << endl;

return 0;

}

double func(int\* arr, int length)

{

double sum = 0;

double result = 0;

if (length > 0)

{

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

sum += arr[i];

result = sum / length;

}

return result;

if (length <= 0)

return 0;

}

double func(double\* arr, int length)

{

double sum = 0;

double result = 0;

if (length > 0)

{

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

sum += arr[i];

result = sum / length;

}

return result;

if (length <= 0)

return 0;

}

4) Написати програму, що виводить у стандартний потік масив чисел типу float по 4 значення в рядку з вирівненням за стовпцями. Точність 3 знаки після коми.

#include <iostream>

#include <iomanip>

using namespace std;

float main() {

int n;

cout << "Enter size:";

cin >> n;

float \*a = new float [ n ];

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

cin >> a[i] ;

}

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

cout.width(4);

printf(" %0.3f ",a[i]);

if (i % 4 == 0)

cout << "\n";

}

cout << endl;

delete[] a;

}

5 варіант

1. **Об'є́ктно-орієнто́ване програмува́ння** (ООП) — одна з [парадигм](https://uk.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%B4%D0%B8%D0%B3%D0%BC%D0%B8_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) [програмування](https://uk.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F), яка розглядає програму як множину «об'єктів», що взаємодіють між собою. Основу ООП складають три основні концепції: [інкапсуляція](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D0%BA%D0%B0%D0%BF%D1%81%D1%83%D0%BB%D1%8F%D1%86%D1%96%D1%8F), [успадкування](https://uk.wikipedia.org/wiki/%D0%A3%D1%81%D0%BF%D0%B0%D0%B4%D0%BA%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)) та [поліморфізм](https://uk.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D1%96%D0%BC%D0%BE%D1%80%D1%84%D1%96%D0%B7%D0%BC_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)). Одною з переваг ООП є краща [модульність](https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D1%8C_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F)) програмного забезпечення (тисячу функцій процедурної мови, в ООП можна замінити кількома десятками класів із своїми методами). Попри те, що ця парадигма з'явилась в [1960-тих](https://uk.wikipedia.org/wiki/1960-%D1%82%D1%96) роках, вона не мала широкого застосування до [1990-тих](https://uk.wikipedia.org/wiki/1990-%D1%82%D1%96), коли розвиток комп'ютерів та комп'ютерних мереж дозволив писати надзвичайно об'ємне і складне програмне забезпечення, що змусило переглянути підходи до написання програм. Сьогодні багато [мов програмування](https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) або підтримують ООП ([PHP](https://uk.wikipedia.org/wiki/PHP), [Lua](https://uk.wikipedia.org/wiki/Lua)) або ж є цілком об'єктно-орієнтованими
2. **class** **Point** {
3. *//можливий доступ лише з методів даного класу*
4. int x, y;
5. bool visibility;
6. **public**:
7. void createPoint(int a, int b) {
8. x = a; y = b;
9. visibility = true;
10. }
11. void setVisibility(bool visibility) {
12. **this**->visibility = visibility;
13. }
14. int getX() {
15. **return** x;
16. }
17. int getY() {
18. **return** y;
19. }
20. };

У цьому прикладі клас Point інкапсулює (приховує) координати точки. Доступ до них можливий лише за певними правилами, які реалізовуються через відповідні методи. Такими правилами можуть бути, наприклад, операція створення точки (установка значень координат), а також операції «увімкнення» і «вимкнення» видимості точки, отримання координат.

Як видно з прикладу, для створення точки необхідно вказати координати точки і «увімкнути» дану точку (зробити її видимою). Якщо б клас був повністю відкритий, то можна б було вручну встановити відповідні поля класу х, y та встановити visible в true. Проте програміст може легко забути встановити якусь з координат або забути встановити видимість. Метод createPoint забезпечує виконання усіх необхідних дій, а закриття доступу до координат змушує діяти лише через використання даного методу. Інкапсульованими також можуть бути і методи класу.

В ООП рекомендується з самого початку створювати закриті поля і лише в разі крайньої необхідності надавати ширший доступ до них. Для роботи із закритими полями класів краще використовувати відповідний метод доступу (getter) та метод мутатор (setter).

**3) Написати перевантаження функції, що обчислюють середнє масиву чисел типу int або short. Продемонструвати їх використання.**

#include <iostream>

using namespace std;

double func(int\* arr, int length);

double func(short\* arr, int length);

int main(void)

{

short arr1[] = { 2,4,6,8,9,5,3 };

int arr2[] = { 2,3,1,4,6 };

cout << "Function with short is:" << func(arr1, 7) << endl;

cout << "Function with int is:" << func(arr2, 5) << endl;

return 0;

}

double func(int\* arr, int length)

{

double sum = 0;

double result = 0;

if (length > 0)

{

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

sum += arr[i];

result = sum / length;

}

return result;

if (length <= 0)

return 0;

}

double func(short\* arr, int length)

{

double sum = 0;

double result = 0;

if (length > 0)

{

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

sum += arr[i];

result = sum / length;

}

return result;

if (length <= 0)

return 0;

}

4) Написати програму, що виводить у стандартний потік масив чисел типуdouble по 3 значення в рядку з вирівнянням за стовпцями. Точність 5 знаки після коми.

#include <iostream>

#include <iomanip>

using namespace std;

int main() {

int n;

cout << "Enter size:";

cin >> n;

double \*a = new double [ n ];

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

cin >> a[i] ;

}

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

cout.width(4);

printf(" %0.5f ",a[i]);

if (i % 3 == 0)

cout << "\n";

}

cout << endl;

delete[] a;

return 0;

}

**Варіант 6**

1.Стан об’єкта – описує дані, що зберігаються в об’єкті.

2. Модульність - властивість системи, яка зв’язана з можливістю декомпозиції на ряд тісно зв’язаних модулів (частин).

Варіант 7

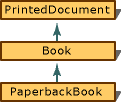
1. Для об’єктно-орієнтованого стилю концептуальна основа полягає в об’єктному підході. Цьому підходу відповідають чотири головних елементи: абстрагування, обмеження доступу, модульність та ієрархія. Ці елементи є головними у тому розумінні, що за одним з класиків об'єктно-орієнтованого проектування програм Граді Бучем [6] без будь-якого з них підхід не буде повністю об’єктно-орієнтованим.
2. Первый пример:   
   в коде приложения расставляем инструкции сдедующего содержания: соединиться, передать пароль, перейти в папку, забрать файл. Все предельно просто и понятно.   
     
   Пример второй, несколько сложнее:  
   Создаем отдельный класс, которому передадим, хост, порт, пароль и путь до файла. Использовать класс будем примерно так:   
     
   записать\_в\_файл(локальный\_путь, новое\_соединение(хост, порт, пароль, путь).закачать())  
     
   Здесь несколько усложняется код внутри класса, но несколько упрощается его использование, не так ли?  
     
   Пример третий, еще более сложный:   
   создадим класс соединения, который будем использовать так:  
     
   Соединение({конфигурация})   
     
   Конфигурация для разных соединений может быть разная. Примеры:   
   {protocol: ftp, port: 224, user: vasia, password: secret}   
   или   
   [proto->http; domain->example.com; file->foo/bar/1.zip]

Варіант 9

1. **абстрагува́ння** — спосіб та метод відокремлення деталей з метою отримання можливості зосередитись на найважливіших особливостях об'єкта.

Грубо кажучи, в програмуванні, абстрагування може бути або керування, або даних. **Абстрагування керування** є абстрагуванням від деталей дій, а **абстрагування даних** є абстрагуванням структур даних. Наприклад, абстрагування керування в [структурному програмуванні](https://uk.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) полягає у використанні [підпрограм](https://uk.wikipedia.org/wiki/%D0%9F%D1%96%D0%B4%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B0) та визначених керівних конструкцій. Абстрагування даних дозволяє обробляти одиниці даних у змістовний спосіб. Наприклад, абстрагування є основною мотивацією створення [типів даних](https://uk.wikipedia.org/wiki/%D0%A2%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85). [Об'єктно-орієнтоване програмування](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) може розглядатись як спроба абстрагувати як керування так і дані.

2.



3) Написати вбудовану функцію, що підносить значення аргументу типу int до квадрату, та продемонструвати її використання.

#include <iostream>

using namespace std;

inline void sqr(int a)

{

int x = a \* a;

cout << "the number of square is:" << x;

}

int main()

{

int x;

sqr(5); //Call it like a normal function...

cin.get();

}

4) Написати програму, що вводить зі стандартного потоку масив довжину N чисел типу float, обчислює середнє значення елементів цього масиву.

#include "stdafx.h"

#include <iostream>

#include <iomanip>

using namespace std;

void input(float\* arr, int n);

double average(float\* arr, int n);

int main()

{

int n = 0;

float\* arr = new float[n];

n = 5;

input(arr,n);

double aver = average(arr, n);

cout << aver;

delete[] arr;

system("pause");

return 0;

}

void input(float\* arr,int n)

{

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

{

cin >> arr[i];

}

}

double average(float\* arr, int n)

{

float sum = 0;

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

{

sum += arr[i];

}

double ser = sum / n;

return ser;

}

5) Створити клас, що асоціюється з датою(рік, місяць, день) та підтримує операції збільшення , зменшення , ініціалізацію місцями та днями і днями, видачу результуту у місяцях або днях. Продемонструйте його виконання.

// ConsoleApplication5.cpp: определяет точку входа для консольного приложения.

//

#include "stdafx.h"

#include <iostream>

#include <iomanip>

using namespace std;

class Date

{

private:

int year;

int mounth;

int day;

public:

Date(int s\_year = 1, int s\_mounth = 1, int s\_day = 1);

virtual ~Date();

void add(int,int,int);

void minus(int,int,int);

void SetMDay(int, int);

void SetDay(int);

int GetDay();

int GetMounth();

};

Date::Date(int s\_day, int s\_mounth, int s\_year)

{

day = s\_day;

mounth = s\_mounth;

year = s\_year;

}

Date::~Date()

{

}

void Date::add(int n, int k, int j)

{

day += n;

mounth += k;

year += j;

}

void Date::minus(int n, int k, int j)

{

day -= n;

mounth -= k;

year -= j;

}

void Date::SetMDay(int s\_day, int s\_mounth)

{

day = s\_day;

mounth = s\_mounth;

}

void Date::SetDay(int s\_day)

{

day = s\_day;

}

int Date::GetDay()

{

if (day > 31 || day < 1)

return 0;

return day + 30 \* mounth + 365 \* year

}

int Date::GetMounth()

{

if (mounth > 12 || mounth < 1)

return 0;

return mounth + year \* 12;

}

int main()

{

Date a;

a.add(4, 4, 4);

a.minus(4, 4, 4);

a.SetMDay(30, 12);

a.SetDay(15);

cout << a.GetDay() << endl;

cout << a.GetMounth() << endl;

system("pause");

}

**Завдання 7**

1. Основні елементи об’єктно – орієнтованого стилю програмування.
2. **Об'єкт ООП** - Окремий екземпляр класу.
3. **Клас ООП** - це спеціальна конструкція, яка використовується для групування пов'язаних змінних та функцій.
4. **Властивість класу** (звана також змінна класу, атрибут або поле-член) - Один з іменованих компонентів визначення даних у визначенні класу.
5. **Метод класу** (званий також функцією-членом) - підпрограма (процедура, функція), що використовується виключно разом із класом (методи класу) або з об'єктом (методи екземпляра)
6. **Конструктор** - метод, що встановлює початковий стан об'єкта;
7. **Деструктор** - метод, що скидає стан об'єкта;
8. **Інкапсуляція (приховування)** - це принцип, згідно з яким будь-який клас повинен розглядатися як чорний ящик - користувач класу повинен бачити і використовувати тільки інтерфейсну частину класу (тобто список декларованих властивостей і методів класу) і не вникати в його внутрішню реалізацію.
9. **Успадкування**- механізм утворення нових класів на основі використання вже існуючих.
10. **Поліморфізм** - концепція в програмуванні, відповідно до якої використовується спільний інтерфейс для обробки різних спеціалізованих типів.
11. **Абстракція** - це надання об'єкту характеристик, які відрізняють його від усіх інших об'єктів, чітко визначаючи його концептуальні кордони.

2) /\* Файл L14.H \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Описание структуры, представляющей монастырь \*/

#define MON struct mon

#define SMON sizeof(struct mon)

struct mon {

char name[15]; /\* название \*/

char sc; /\* школа \*/

int cnt; /\* количество монахов \*/

float sq; /\* площа \*/

};

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Лабораторнаф работа ╧14 \*/

/\* Файл L14-1.H \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Описания функций файла L14-1.C \*/

int check\_number(int);

void del\_item(int);

void show\_all(void);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Лабораторная работа ╧14 \*/

/\* Файл L14-2.H \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Описания функций файла L14-2.C \*/

void print\_head(void);

void print\_line(void);

void show\_row(MON \*);

int get\_number(void);

void show\_1(MON \*);

int ent\_data(MON \*);

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Лабораторная работа ╧14 \*/

/\* Файл L14.C \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <stdio.h>

#include <conio.h>

#include "l14.h"

#include "l14-1.h"

#include "l14-2.h"

#define N 100

MON mmm[N]; /\* массив-таблица \*/

int n=0; /\* количество элементов в массиве \*/

/\*\*\*\* главная функция \*\*\*\*/

int main(void) {

int op; /\* операция \*/

int num; /\* номер элемента \*/

char eoj; /\* признак конца \*/

/\* \*/

for (eoj=0; !eoj; ) {

/\* выводення меню \*/

printf("1 - Добавить элемент\n");

printf("2 - Удалить элемент\n");

printf("3 - Показать элемент по номеру\n");

printf("4 - Показать все\n");

printf("0 - Выход\n");

printf("Вводите >");

/\* вибор из меню \*/

scanf("%d",&op);

switch(op) {

case 0: /\* выход \*/

eoj=1;

break;

case 1: /\* добавить \*/

if (!ent\_data(mmm+n)) n++;

break;

case 2: /\* удалить \*/

if (!check\_number(num=get\_number())) {

del\_item(num);

n--;

}

break;

case 3: /\* показать один \*/

if (!check\_number(num=get\_number()))

show\_1(mmm+num-1);

break;

case 4: /\* показать все \*/

show\_all();

break;

default:

printf("Неправильная операция\n");

break;

} /\* switch \*/

if (op) {

printf("Нажмите любую клавишу\n");

getch();

} /\* if \*/

} /\* for \*/

return 0;

} /\* main \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Лабораторная работа ╧14 \*/

/\* Файл L14-1.C \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Функции, которые используют глобальные переменные \*/

#include <stdio.h>

#include <mem.h>

#include "l14.h"

#include "l14-2.h"

extern MON mmm[];

extern int n;

/\*\*\*\* проверка номера элемента \*\*\*\*/

int check\_number(int a) {

if (a<1) {

printf("Минимальный номер : 1\n");

return -1;

}

if (a>n) {

printf("Максимальный номер : %d\n",n);

return -1;

}

return 0;

}

/\*\*\*\* удаление элемента \*\*\*\*/

void del\_item(int m) {

int i;

for (; m<n; m++)

memcpy(mmm+m-1,mmm+m,SMON);

}

/\*\*\*\* выво всего массива \*\*\*\*/

void show\_all() {

int i;

print\_head();

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

show\_row(mmm+i);

print\_line();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* Лабораторная работа ╧14 \*/

/\* Файл L14-2.C \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <stdio.h>

#include <string.h>

#include "l14.h"

#include "l14-2.h"

/\*\*\*\* ввод номера \*\*\*\*/

int get\_number() {

int b;

printf("Введите номер>");

scanf("%d",&b);

return b;

}

/\*\*\*\* ввод данных про один монастырь \*\*\*/

int ent\_data(MON \*m) {

float sqx;

/\* Ввод данных \*/

printf("Введите название, школу, количество, площадь>");

scanf("%s %c %d %f",m->name,&m->sc,&m->cnt,&sqx);

m->sq=sqx;

if (!strcmp(m->name,"\*\*\*")) return -1;

if (strchr("ТСД",m->sc)==NULL) {

printf("Ошибка\n");

return -2;

}

return 0;

}

/\*\*\*\* вывод данных про один монастырь \*\*\*/

void show\_1(MON \*m) {

printf("\nНазвание : %s\n",m->name);

printf("Школа : ");

switch(m->sc) {

case 'Т': printf("Тендай"); break;

case 'С': printf("Сингон"); break;

case 'Д': printf("Дзедзицу"); break;

}

printf("\nКоличество монахов : %d\n",m->cnt);

printf("Площадь земель : %6.2f\n",m->sq);

print\_line();

}

/\*\*\*\* вывод строки таблицы \*\*\*\*/

void show\_row(MON \*m) {

printf("| %9s | %c | %3d | %-5.1f |\n",

m->name,m->sc,m->cnt,m->sq);

}

/\*\*\*\* вывод подчеркивания \*\*\*\*/

void print\_line() {

printf("-----------------------------------------------\n");

}

/\*\*\*\* вывод заголовка таблицы \*\*\*\*/

void print\_head() {

print\_line();

printf("|Буддийские монастыри Японии периода Нара |\n");

printf("|---------------------------------------------|\n");

printf("| Название | Школа | Количество| Площадь |\n");

printf("| | | монахов | земель (га) |\n");

printf("|-----------|-------|-----------|-------------|\n");

}

3) Написати вбудовану функцію, що підносить значення аргументу типу float до квадрату, та продемонструвати її використання.

#include <iostream>

using namespace std;

inline void sqr(float a)

{

float x = a \* a;

cout << "the number of square is:" << x;

}

float main()

{

float x;

sqr(5.5); //Call it like a normal function...

cin.get();

}

**Завдання 9**

**1) абстрагува́ння** — спосіб та метод відокремлення деталей з метою отримання можливості зосередитись на найважливіших особливостях об'єкта.

Грубо кажучи, в програмуванні, абстрагування може бути або керування, або даних. **Абстрагування керування** є абстрагуванням від деталей дій, а **абстрагування даних** є абстрагуванням структур даних. Наприклад, абстрагування керування в [структурному програмуванні](https://uk.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) полягає у використанні [підпрограм](https://uk.wikipedia.org/wiki/%D0%9F%D1%96%D0%B4%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B0) та визначених керівних конструкцій. Абстрагування даних дозволяє обробляти одиниці даних у змістовний спосіб. Наприклад, абстрагування є основною мотивацією створення [типів даних](https://uk.wikipedia.org/wiki/%D0%A2%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%B8%D1%85). [Об'єктно-орієнтоване програмування](https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) може розглядатись як спроба абстрагувати як керування так і дані.

**2)Приклад ієрархії одиничного наслідування**

lass ArrayWithAdd : public Array {  
  
ArrayWithAdd(int n) : Array(n) {  
  
} ArrayWithAdd() : Array() {}  
  
ArrayWithAdd(const Array& a) : Array(a) {}  
  
void Add(const Array& a);

**3)** Написати вбудовану функцію, що обчислює двійковий логарифм зі значенням аргументу short, та продемонструвати її використання.

#include <iostream>

#include <cstdlib>

#include <cmath>

inline short log(short b)

{

return log(b) / log(2);

}

int main()

{

std::cout << log(4)<< std::endl;

return EXIT\_SUCCESS;

}