Лабораторна робота №1

Тема: Розробка програм з використанням найпростіших класів та об’єктів.

Мета: Набуття навичок в розробці найпростіших класів та роботі з об’єктами класів.

Порядок виконання роботи

1. Ознайомитися з теоретичними основами та принципами роботи з динамічними масивами.

2. Розробити структуру класу для роботи з динамічними масивами, яка повинна включати не менше 2-х даних-членів класу та 5-х методів класу, з яких обов'язково повинні бути метод-член класу для:

- створення масиву;
- заповнення масиву;
- виводу змісту масиву на екран;
- очищення пам'яті.

3. Розробити метод для обробки динамічних масивів, що використовує клас для виконання завдання 1 та головну програму, яка використовує розроблений клас.

4. Розробити структуру класу для роботи з двовимірними масивами, яка повинна включати не менше 3-х даних-членів класу та 5-х методів класу, з яких обов'язково повинні бути метод-член класу для:

- створення масиву;
- заповнення масиву;
- виводу змісту масиву на екран;
- очищення пам'яті.

5. Розробити метод-член класу для обробки двовимірного масиву, відповідно до завдання 2 та головну програму, яка використовує розроблений клас.

6. Доповнити клас для роботи з одновимірними динамічними масивами з пункту 2 функцією, яка задана в завданні 3.

7. Розробити 2-3 тести для перевірки вірності роботи даної програми.

8. Оформити звіт до лабораторної роботи.

Завдання 1

Варіант 1. Розробити метод-член класу для виводу на екран усіх позитивних елементів одновимірного динамічного масиву і їх суми і кількості.

Варіант 2. Розробити метод-член класу для виводу на екран усіх негативних елементів одновимірного динамічного масиву і їх суми і кількості.

Варіант 3. Розробити метод-член класу для виводу на екран кількості нульових елементів одновимірного динамічного масиву і їх порядкових номерів.

Варіант 4. Розробити метод-член класу для виводу на екран суми першої і другої половини одновимірного динамічного масиву і кількості позитивних елементів у кожній.

Варіант 5. Розробити метод-член класу для виводу на екран середнього значення і дисперсії даного одновимірного динамічного масиву.

Варіант 6. Розробити метод-член класу впорядкування по зростанню методом бульбашки одновимірного динамічного масиву.

Варіант 7. Розробити метод-член класу впорядкування по спаданню методом бульбашки одновимірного динамічного масиву.

Варіант 8. Розробити метод-член класу для визначення мінімального і максимального елементів одновимірного динамічного масиву та їх порядкових номерів.

Варіант 9. Розробити метод-член класу для перетворення одновимірного динамічного масиву таким чином, щоб його максимальний елемент став першим елементом, а мінімальний – останнім, без впорядкування всього масиву.

Варіант 10. Розробити метод-член класу для перетворення одновимірного динамічного масиву таким чином, щоб його максимальний елемент став останнім, а мінімальний – першим, без впорядкування всього масиву.

Варіант 11. Розробити метод-член класу для перетворення одновимірного динамічного масиву таким чином, щоб спочатку розташовувалися упорядковані по зростанню його позитивні елементи.

Завдання 2

Варіант 1. Розробити метод-член класу для створення нового одновимірного масиву з усіх позитивних елементів заданого двовимірного динамічного масиву.

Варіант 2. Розробити метод-член класу для створення нового одновимірного масиву з усіх негативних елементів заданого двовимірного динамічного масиву.

Варіант 3. Розробити метод-член класу для створення нового одновимірного масиву з суми усіх позитивних елементів кожного стовпця заданого двовимірного динамічного масиву.

Варіант 4. Розробити метод-член класу для створення нового одновимірного масиву з суми усіх негативних елементів кожного стовпця заданого двовимірного динамічного масиву.

Варіант 5. Розробити метод-член класу для створення нового одновимірного масиву з кількості всіх позитивних елементів кожного рядка заданого двовимірного динамічного масиву.

Варіант 6. Розробити метод-член класу для створення нового одновимірного масиву з кількості всіх негативних елементів кожного рядка заданого двовимірного динамічного масиву.

Варіант 7. Розробити метод-член класу для створення нового одновимірного масиву з середнім арифметичним значенням всіх позитивних елементів кожного стовпця заданого двовимірного динамічного масиву.

Варіант 8. Розробити метод-член класу для створення нового одновимірного масиву з середнім арифметичним значенням всіх негативних елементів кожного стовпця заданого двовимірного динамічного масиву.

Варіант 9. Розробити метод-член класу для створення нового одновимірного масиву з суми усіх позитивних елементів верхньої і нижньої трикутних матриць заданого двовимірного динамічного масиву.

Варіант 10. Розробити метод-член класу для створення нового одновимірного масиву з суми усіх негативних елементів верхньої і нижньої трикутних матриць заданого двовимірного динамічного масиву.

Варіант 11. Розробити метод-член класу для перетворення заданого двовимірного динамічного масиву таким чином, щоб кожен його стовпець був упорядкований по зростанню методом бульбашки.

Завдання 3

Варіант 1. Розробити функцію для визначення елемента кратного 5 і метод-член класу, який використовує цю функцію для визначення кількості і суми елементів даного одновимірного динамічного масиву, кратних п’яти.

Варіант 2. Розробити функцію для визначення елемента кратного 7 і метод-член класу підпрограму, яка використовує цю функцію для визначення кількості і суми елементів даного одновимірного динамічного масиву, кратних семи.

Варіант 3. Розробити функцію визначення елемента кратного 9 і підпрограму, що використовує цю функцію для визначення кількості і суми елементів даного одновимірного динамічного масиву, кратних дев'яти.

Варіант 4. Розробити функцію визначення елемента кратного 11 і підпрограму, що використовує цю функцію для визначення кількості і суми елементів даного одновимірного динамічного масиву, кратних одинадцяти.

Варіант 5. Розробити функцію визначення елемента кратного 12 і підпрограму, що використовує цю функцію для визначення кількості і суми елементів даного одновимірного динамічного масиву, кратних дванадцяти.

Варіант 6. Розробити функцію визначення кубічного кореня з заданого члена і підпрограму, що використовує цю функцію для перетворення кожного елемента одновимірного динамічного масиву.

Варіант 7. Розробити функцію визначення квадратного кореня з заданого члена і підпрограму, що використовує цю функцію для перетворення кожного парного елемента одновимірного динамічного масиву.

Варіант 8. Розробити функцію визначення a^3 для заданого елемента a і підпрограму, що використовує цю функцію для перетворення кожного елемента заданого одновимірного динамічного масиву.

Варіант 9. Розробити функцію визначення a^2 для заданого елемента a і підпрограму, що використовує цю функцію для перетворення кожного елемента заданого одновимірного динамічного масиву.

Варіант 10.Розробити функцію визначення парного елемента і метод, що використовує цю функцію для визначення суми, кількості і середнього арифметичного всіх парних елементів динамічного масиву.

Варіант 11. Розробити функцію визначення непарного елемента і метод, що використовує цю функцію для формування нового динамічного масиву, що містить тільки парні елементи вихідного одновимірного динамічного масиву.

Теоретичні відомості

Об'єктно-орієнтований підхід використовує наступні базові поняття:

- ·об'єкт;
- ·властивість об'єкта;
- ·метод обробки;
- ·подія ;
- ·клас об'єктів.

Розглянемо кожен з цих понять.

Об'єкт це сукупність властивостей (параметрів) визначених сутностей і методів їх обробки (програмних засобів). Об'єкт містить інструкції (програмний код), що визначають дії, які може виконувати об'єкт, та оброблювані дані.

Властивість - характеристика об'єкта, його параметр. Всі об'єкти наділені певними властивостями, що у сукупності виділяють об'єкт із множини інших об'єктів. Об'єкт має якісну визначеність, що дозволяє виділити його з множини інших об'єктів і обумовлює незалежність створення й обробки від інших об'єктів.

Одним із властивостей об'єкта є метод його обробки. **Метод** - програма дій над об'єктом чи його властивостями. Метод розглядається як програмний код, пов'язаний з певним об'єктом; здійснює перетворення властивостей, змінює поведінку об'єкта. Об'єкт може мати набір заздалегідь визначених вбудованих методів обробки, або створених користувачем чи запозичених у стандартних бібліотеках, що виконуються при настанні заздалегідь визначених подій, наприклад, однократне натискання лівої

кнопки миші, вхід у поле введення, вихід з поля введення, натискання певної клавіші і т.п.

В міру розвитку систем обробки даних створюються стандартні бібліотеки методів, до складу яких включаються типізовані методи обробки об'єктів певного класу (аналог - стандартні підпрограми обробки даних при структурному підході), які можна запозичати для різних об'єктів.

Подія - зміна стану об'єкта. Зовнішні події генеруються користувачем (наприклад, клавіатурне введення чи натискання кнопки миші, вибір пункту меню, запуск макроса); внутрішні події генеруються системою.

Об'єкти можуть поєднуватися в класи (групи чи набори - у різних програмних системах можлива інша термінологія).

Клас - це структурований користувацький тип, що поєднує дані і функції, що їх перетворюють, в єдине ціле. Механізм класів дозволяє створювати типи в повної відповідності з принципами абстракції даних, тобто клас задає певну структурну сукупність типизовіаних даних та дозволяє визначити набір операцій над цими даними.

Формат класу в алгоритмічной мові С++ має наступний вигляд:

class ім’я_класу { список_компонентів класу};

де class – службове слово мови С++, ім’я_класу – вільно обраний програмистом ідентифікатор, список_компонентів – опис та визначення типізованих даних та функцій класу.

Компонентами класу можуть бути дані, функції, класи, бітові поля, ім’я типів. Список компонентів в фігурних дужках називається тілом класу. Визначення класу в С++ завжди закінчується символом “;”. Функції, що належать класу, називають функції-члени класу (member functions), а дані, що належать класу, називають дані-члени класу (data members).

В програмуванні клас визначає множину об’єктів, які об’єднаються за однаковими властивостями (єдиною групою даних) та сукупністю

однакових функцій. Для опису об'єкту класу в С++ використовують наступний формат:

ім'я_класу ім'я_об'єкту;

Як тільки об'єкт класу визначається в програмі, виникає можливість звертатися до його компонент (або даних, або функцій) за допомогою оператора доступу, формат якого представлений нижче:

ім'я_об'єкту.ім'я_даного; або

ім'я_об'єкту.ім'я_функції();

або через вказівник на об'єкт класу формат має вигляд:

вказівник_на_об'єкт_класу -> ім'я_даного;

вказівник_на_об'єкт_класу->і'м'я_функції();

Вказівник на об'єкт класу дозволяє викликати функції-члени класу для обробки даних того об'єкту, який адресується вказівником. Так, наприклад, викликати фунцію-член `Display()` класу D можна оператором виду

```
D.Display();
```

або через вказівник `pointer` на об'єкт класу оператором виду:

```
pointer-> Display();
```

Розробимо структуру класу `CTime`, що містить дані-члени класу: `year, month, day, hour, minute` і функції-члени класу для їхнього введення, виведення та обробки. Структура класу наведена нижче.

```
class CTime{
// специфікатор доступу до членів класу зі зовнішнього середовища
Public:
// дані – члени класу
          int year;
          int month;
          int day;
          int hour;
          int minute;
         void Display (void);// функція-член класу
```

Опис класу схожий на опис структури. Специфікатор доступу `public` - контролює можливість використання членів класу в зовнішніх програмах, і відкриває доступ до всіх членів класу, що знаходяться за ним, для всіх користувачів класу, тому такі члени класу називаються **відкритими.**

Дані-члени класу можуть бути: змінними, покажчиками, посиланнями, масивами, структурами, об'єктами класу, і т.д. Структура класу відображує перший принцип об’єктно-орієнтованого програмування - інкапсуляції. Інкапсуляція означає сполучення даних із методами їх обробки в абстрактних типах даних - класах об'єктів (рисунок 1.1). Тому функції-члени класу описуються в тілі класу прототипами функцій і презначені для виконання певних операції над даними-членами класу.

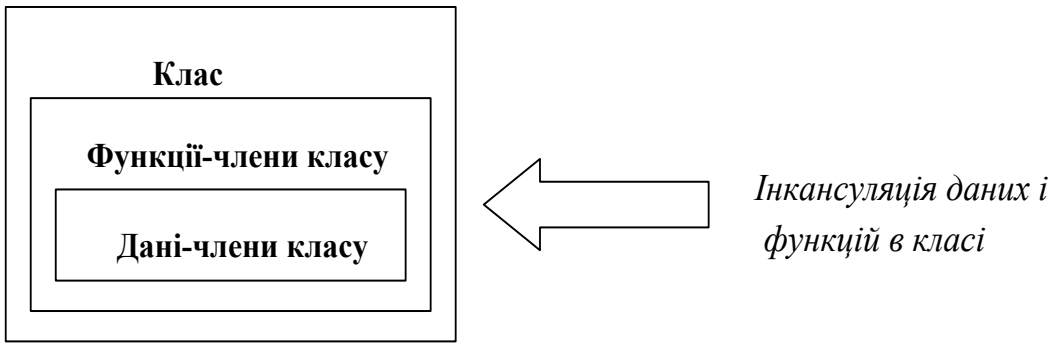

Рисунок 1.1 - Графічна інтерпретація властивості классу - інкапсуляції

Нижче наведений приклад використання об’єктів класу в функції `main ()`.

Приклад

```
# include < iostream.h >
# include < stdio.h >
      class  CTime {
          public:    // специфікатор
            int year;
             int month;
             int day;
```

```
            int hour;
            int minute;
      void Display ( void );   // функція-член
           };
int main ()
{
  CTime  object1;        // об'єкт типу CTime; екземпляр типу
класа
 object. month=7;  // ініціалізація даних-членів об'єкту;
object. day=14;
object. year=2003;    ініціалізація даних об'єкту
object. hour=8;
object. minute=30;
object. Display ();    //  виклик функції об'єкту
cout << ' \ n The end ';
return 0;
}
   //опис функції-члена класу
 void CTime :: Display ( void )
{
   char  s [32];

sprintf(s,"Data:%02d/%02d%/%04dTime:%02d:%02d\
    n"=,month, day, year, hour, minute );
   cout << s ;}
```

У даному прикладі клас `CTime` - усього лише шаблон (схема), що описує формат членів класу, та для роботи з ним в функції `main()` створений об'єкт цього класу `object`. При ініціалізації даних-членів класу для доступу до даних об'єкту класу використовується оператор крапки (`object.day`).

Якщо в програмі використовують декілька об'єктів одного класу `CTime`:

```
CTime  today;
CTime  tomorrow;
CTime   yesterday;
```

то для цих об'єктів функцію `Display( )` можна викликати таким чином:

`today.Display();` - виклик функції об'єкту `today`;

`tomorrow.Display();` - виклик функції об'єкту `tomorrow;`

`yesterday.Display();` - виклик функції об'єкту `yesterday;`

Організація доступу до даних-членів класу. **Специфікатори доступу**

В одному класі можуть бути дані-члени, для деяких з них доступ з зовнішнього середовища відкритий, а для деяких - закритий.

Для організації доступу до даних та функцій класу використовують спеціальні специфікатори доступу: `private` - доступ закритий*;* `public` **-** доступ відкритий.

Звичайно `private` використовується для даних-членів класу з метою сховати від користувача деталі збереження даних в об'єктах, у той же час забезпечуючи їх методами можливість використання цих даних. У результаті в програмі можна модернізувати способи збереження й обробки даних у середині класу, не переписуючи при цьому методи доступу і виклику їх у зовнішньому коді.

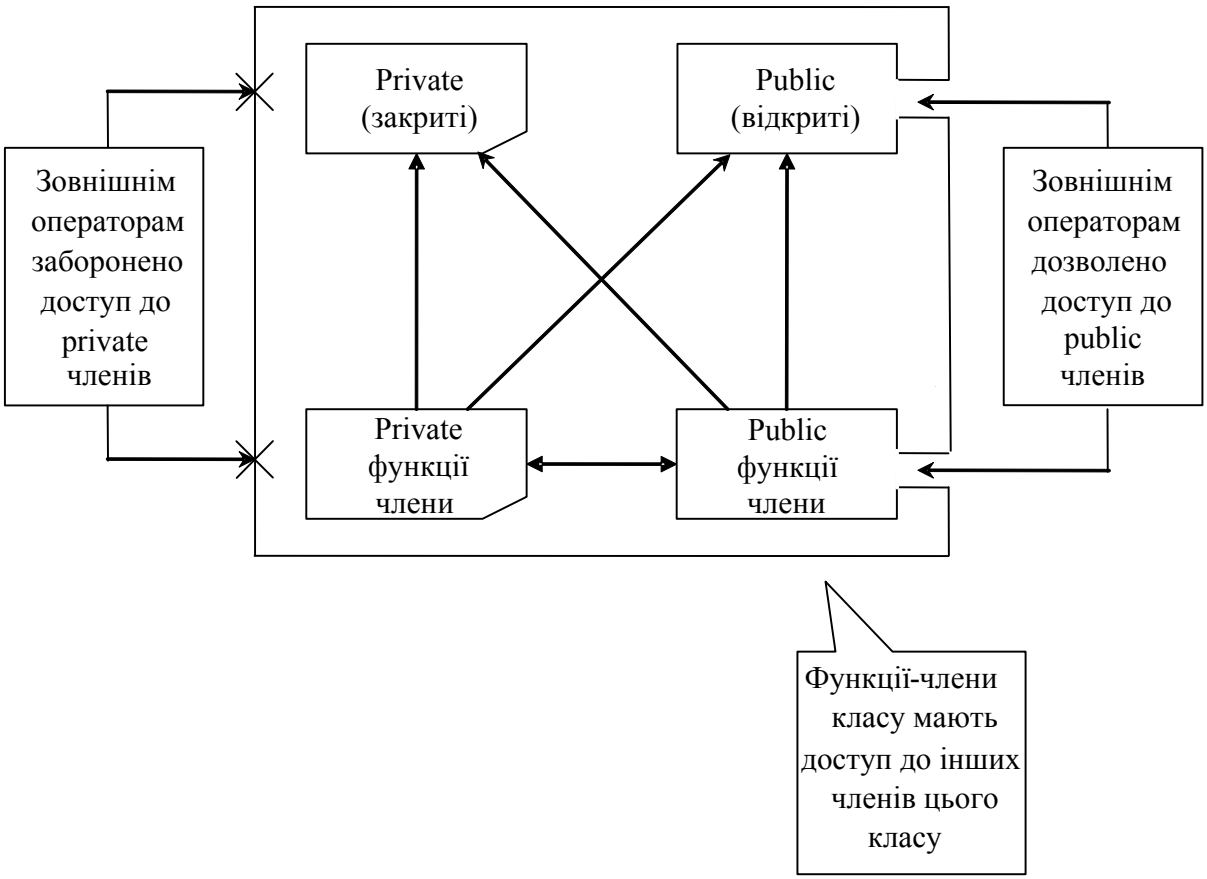

Рисунок 1.2 - Графічна інтерпретація відів доступу до даних та методів класу з зовнішнього середовища

Члени класу можуть бути закриті за замовчуванням, але використання в в структурі класу специфікатора private бажано. В класі можуть бути декілько секцій відкритих і закритих компонентів класу, що розташовуються в довільному порядку

Для використання закритих `(private)` даних-членів існує лише один спосіб – через виклик відкритих для зовнішнього середовища функцій-членів класу, тому що закриті `(private)` члени класу доступні тільки членам цього класу і нікому більше, тому що закриті члени невидимі поза класом. Графічна інтерпретація спосібів доступу до даних та функцій-членів класу з зовнішнього середовища представлена на рисунку 1.2.

Нижче наведений приклад програми, що використовує специфікатори доступу `private` і `public`

Приклад

```
# include < iostream.h >
#include <stdio.h>
 class  CTime {
 private:
     int year;
     int month;
     int day;
     int hour;
     int minute;
 public:
     void Display(void);
     void GetTime(int &m,int &d,int&y,int &h2,int
&min);
     void SetTime(int m,int d,int y,int h,int
min);
  };
   int main ( )
      {
       CTime  obj1;  // об'єкт типу  CTime
       int month, day, year, hous, minute;

       obj1.SetTime (7, 14, 2003, 8, 30);
       cout <<"obj1==";  obj1.Display();
       obj1.GetTime(month, day, year, hous, minute);
       obj1.GetTime(month,day,year, ++hous, minute);
       cout <<" Next hous==";  obj1.Display();
       return ();}

void CTime::Display(void)
{
   char s [32];
   Sprintf(s,"Data:%02d/%02d%/%04d
  Time:%02d:%02d\n" =,month, day, year, hour, minute);
     cout <<s;
}
```

Специфікатори доступа

закриті дані члени

Оператор дозволу області видимості

```
void CTime::Get Time(int&m,int&d,int&y,int&h2,int&
min);
}
   m=month; // Повернення даних-членів тому, хто викликав функцію
  d=day;
  y=year;
  h2=hare;
   min=minute;
}
void CTime :: Set Time (int m, int d, int y, int h, int
min);
     {
        month = m;//Присвоювання аргументів даним-членам класу
        day = d;
        year = y;
        hare = h2;
        minute = min;
     }
```

Вбудовані функції-члени

В мові С++ дозволена можливість використання класу з вбудованими функціями-членами, тобто якщо тіло функції складається з декількох операторів, то її можна повністю описати в тілі класу.

Функції-члени класу, що вбудовуються в тіло класу, використовуються так само, як і інші функції, але при проектуванні структури класу треба пом’ятати, що компілятор вбудовує код функції в кожне місце її виклику замість оператору виклику, тобто у скомпільованому коді таки функції не викликаються, а вставляються безпосередньо в скомпільовану програму (рисунок 1.3).

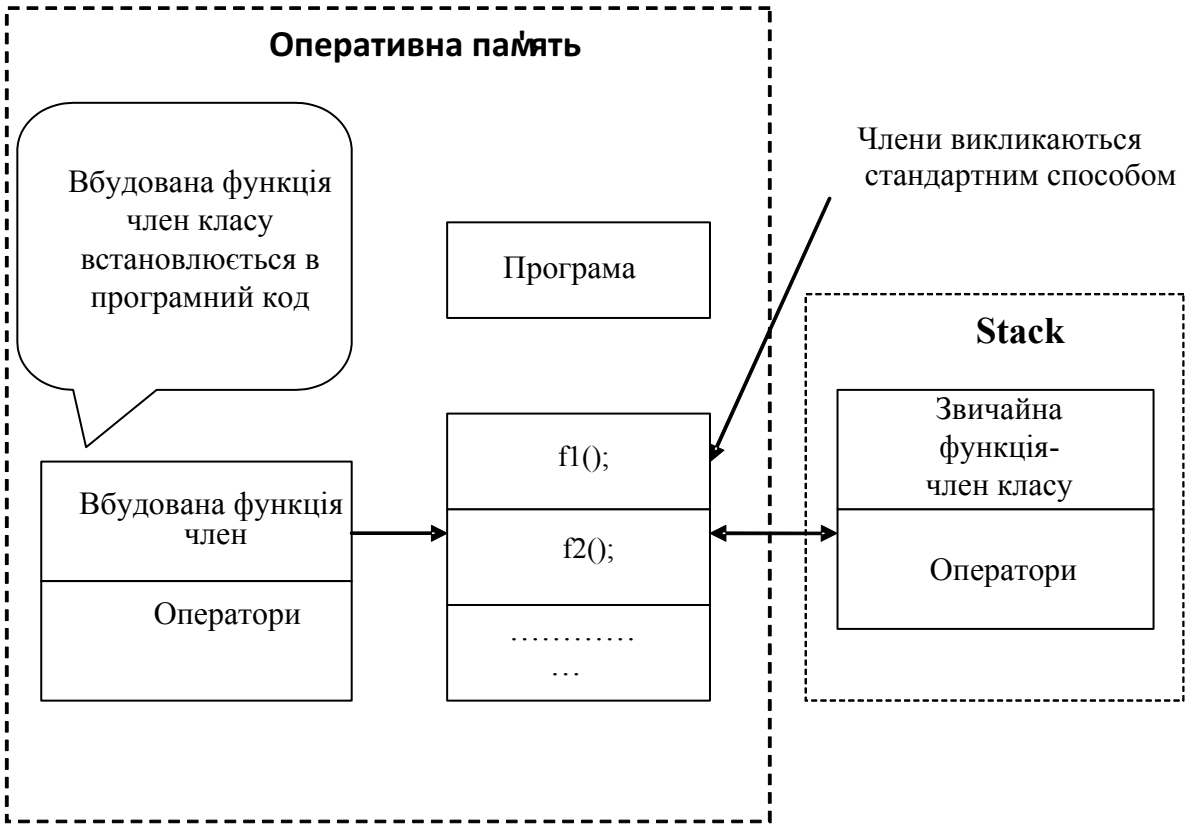

Рисунок 1.3 – Геометрична інтерпретація вбудованих функцій членів класу

Наприклад,

```
# include < iostream.h >
# include < time.h >
# include < string.h >
 class  CTime {
  private:
    long  dt;  //дані дата і час перетворяться в секунди і
```

представлені у виді довгого цілого. Це зручно для використання різних бібліотечних функцій, використаних такою формою збереження даних повертає у виді символьного рядка дату і час.

```
 public:
    void Display(void){cout<<ctime(&dt);}
```

Вбудована функція-член класу

```
    void Get Time (int & m, int & d, int & y, int &
h2, int & min);
    void Set Time (int m, int d, int y, int h, int min);
   char  * Get STime ( void )
```

```
     {char *cp = strdup(ctime (&dt)); }
      return cp;
}
//зсув в часі додає до поточного часу n minutes хвилин
void Change Time(long n minutes){dt+=(n minutes×60);}
}
```

Приклад виконання завдання 1

Варіант 1. Розробити функцію для виводу на екран усіх негативних елементів динамічного масиву і їх суми і кількості.

Дана програма буде складатися з семи функцій членів класу. Функція `Create_ar1D(int n)` призначена для створення одномірного масиву з розміром `n`. Функція `Clear()` призначена для очищення масиву. `Input_ar1D()` призначенна для заповнення масиву з клавіатури. Функція `Print_ar1D()` призначена для виводу масиву на екран. Функція `Negative_ar1D()` повертає кількість від'ємних елементів масиву. Функція `NegativeSum_ar1D()` повертає суму всіх від'ємних елементів масиву. Функція `PrintNegative_ar1D()` виводить на екран всі від'ємні елементи масиву.

Клас буде мати наступну структуру:

```
class CArray_1D{
    int* array; //дані-члени класу
    int size;
 public:
    void Create_ar1D(int n);  //функції-члени класу
    void Clear();
    void Input_ar1D();
    void Print_ar1D();
    int Negative_ar1D();
    int NegativeSum_ar1D();
    void PrintNegative_ar1D();
};
```

Далі потрібно описати кожну з функцій класу. Головна програма працює з об'єктом `obj1` класу `CArray_1D`, за допомогою якої можна звертатися до функцій класу.

Нижче наведений лістинг програми.

```
#include <iostream>
#include <malloc.h>
#include <stdlib.h>
using namespace std;
class CArray_1D{
    int* array;  //дані-члени класу
    int size;
 public:
    void Create_ar1D(int n); //функції-члени класу
    void Clear();
    void Input_ar1D();
    void Print_ar1D();
    int Negative_ar1D();
    int NegativeSum_ar1D();
    void PrintNegative_ar1D();
};
int main()
{
    int size;
    cout << "Enter the size of array:" << endl;
    cin >> size;
    CArray_1D obj1;
    obj1.Create_ar1D(size);
    obj1.Input_ar1D();
    obj1.Print_ar1D();
    int negnum = obj1.Negative_ar1D();
    int negsum = obj1.NegativeSum_ar1D();
    cout<<endl<<"Number of negative = "<<
    negnum<<endl;
    cout<<endl<<"Amount of negative = "<<
    negsum<<endl;
    obj1.PrintNegative_ar1D();
    system("pause");
}
void CArray_1D::Create_ar1D(int n)
{
    size = n;
```

```
    array = (int*)malloc(size*sizeof(int));
    //виділення пам'яті під масив
}
void CArray_1D::Clear()
{
    free(array);
    //звільнення памяті
}
void CArray_1D::Input_ar1D()
{
    cout<<endl<<"Enter numbers of array:"<<
    endl;
    for(int i=0; i<size; i++) //заповнення масиву
    {
        cout << "array[" << i << "] = ";
        cin >> array[i];
    }
}
void CArray_1D::Print_ar1D()
{
    cout << endl << "Numbers of array:" << endl;
    for(int i=0; i<size; i++)
    {
        cout<<"array["<<i<<"] = "<<array[i]<<
        endl;
        //виведення масиву на екран
    }
}
int CArray_1D::Negative_ar1D()
{
    int count = 0;
    for(int i=0; i<size; i++)
    {
        if(array[i] < 0) count++;
        //якщо елемент менше нуля то збільшуємо
        //лічильник на одиницю
    }
    return count;
}
int CArray_1D::NegativeSum_ar1D()
{
    int sum = 0;
    for(int i=0; i<size; i++)
```

```
        {
            if(array[i] < 0) sum = sum + array[i];
            //якщо елемент менше нуля то додаємо
            //до суми
        }
        return sum;
    }
    void CArray_1D::PrintNegative_ar1D()
    {
        cout<<endl<<"Negative numbers of array:"<<
        endl;
        for(int i=0; i<size; i++)
        {
            if(array[i] < 0)
                cout<<"array["<<i<<"] = "<<
                array[i] << endl;
                //якщо елемент менше нуля, то
                // виводимо значення на екран
        }
    }
```

Тестування

Для перевірки правильності роботи програми введемо в неї дані при введені, яких результат заздалегідь відомий. Наприклад сформуємо масив з 6 елементів такого виду:

5 -3 -12 0 3 -1

З введених даних наочно видно, що від'ємними є 2,3 і 6 елемент. Сума їх =-16. Кількість 3.

Тепер введемо ці ж дані в програму і перевіримо результат. Нижче представлені результати тестування даної програми.

Enter the size of array:	Numbers of array:
6	array[0] = 5
Enter numnbers of array:	array[1] = -3
array[0] = 5	array[2] = -12
array[1] = -3	array[3] = 0
array[2] = -12	array[4] = 3

array[3] = 0
array[4] = 3
array[5] = -1

array[5] = -1
Number of negative = 3
Amount of negative = -16
Negative numbers of array:
array [1]=-3
array [2]=-12
array [5]=-1

Контрольні запитання

1. Що таке об’єкт та клас?

2. Структура класу в C++?

3. Формат типу `class` в C++.

4. Основні принципи ООП?

5. В чому сутність принципу інкапсуляції?

6. Для чого використовують інкапсуляцію?

7. Які ви знаете специфікатори доступу.

8. Як описуються функції-члени класу?

9. Що таке екземпляр класу?

10. Що відносять до даних-членів класу?

11. Як здійснюється ініціалізація даних-членів класу?

12. Які специфікатори доступу використовують для даних-членів класу?

13. Що таке об’єкт класу?

14. Наведіть приклад об’єкту класу.

15. Вкажіть способи опису класів.

16. Що таке закриті члени класу?

17. Що таке відкриті члени класу?

18. Як організувати доступ до закритих членів класу?

19. Як задати початкові значення даним-членам класу (всі способи)?

20. Скільки об'єктів класу може бути використано в програмі, яка застосовує клас?