**Лабораторна робота №1**

**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.

**Завдання 1**

1. Розробити метод-член класу для виводу на екран усіх позитивних елементів одновимірного динамічного масиву і їх суми і кількості.
2. Розробити метод-член класу для виводу на екран усіх негативних елементів одновимірного динамічного масиву і їх суми і кількості.
3. Розробити метод-член класу для виводу на екран кількості нульових елементів одновимірного динамічного масиву і їх порядкових номерів.
4. Розробити метод-член класу для виводу на екран суми першої і другої половини одновимірного динамічного масиву і кількості позитивних елементів у кожній.
5. Розробити метод-член класу для виводу на екран середнього значення і дисперсії даного одновимірного динамічного масиву.
6. Розробити метод-член класу для виведення на екран елемента, який зустрічається найчастіше в масиві.
7. Розробити метод-член класу для визначення суми елементів масиву, розташованих між першим і другим позитивними елементами.
8. Розробити метод-член класу для визначення мінімального і максимального елементів одновимірного динамічного масиву та їх порядкових номерів.
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.
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.
11. Розробити метод-член класу для визначення суми мінімального та середнього елементів одновимірного динамічного масиву.
12. Розробити метод-член класу для перетворення одновимірного динамічного масиву таким чином, щоб спочатку розташовувалися упорядковані по зростанню його позитивні елементи.
13. Розробити метод-член класу для визначення суми модулів елементів масиву, розташованих після першого елементу, рівного нулю.
14. Розробити метод-член класу для визначення добутку елементів масиву, розташованих між першим і другим нульовими елементами.
15. Розробити метод-член класу для визначення суми елементів масиву, розташованих до останнього позитивного елемента.
16. Розробити метод-член класу для визначення суми модулів елементів масиву, розташованих після першого негативного елемента

**Завдання 2**

1. Розробити метод-член класу для створення нового одновимірного масиву з усіх позитивних елементів заданого двовимірного динамічного масиву.
2. Розробити метод-член класу для створення нового одновимірного масиву з усіх негативних елементів заданого двовимірного динамічного масиву.
3. Розробити метод-член класу для створення нового одновимірного масиву з суми усіх позитивних елементів кожного стовпця заданого двовимірного динамічного масиву.
4. Розробити метод-член класу для створення нового одновимірного масиву з суми усіх негативних елементів кожного стовпця заданого двовимірного динамічного масиву.
5. Розробити метод-член класу для створення нового одновимірного масиву з кількості всіх позитивних елементів кожного рядка заданого двовимірного динамічного масиву.
6. Розробити метод-член класу для створення нового одновимірного масиву з кількості всіх негативних елементів кожного рядка заданого двовимірного динамічного масиву.
7. Розробити метод-член класу для створення нового одновимірного масиву з середнім арифметичним значенням всіх позитивних елементів кожного стовпця заданого двовимірного динамічного масиву.
8. Розробити метод-член класу для створення нового одновимірного масиву з середнім арифметичним значенням всіх негативних елементів кожного стовпця заданого двовимірного динамічного масиву.
9. Розробити метод-член класу формування одновимірного масиву, *i*-ий елемент якого є добутком модулів непарних елементів *i*-го рядка двовимірного масиву.
10. Розробити метод-член класу для створення нового одновимірного масиву з суми усіх негативних елементів верхньої і нижньої трикутних матриць заданого двовимірного динамічного масиву.
11. Розробити метод-член класу для створення нового одновимірного масиву з парних мінімальних елементів кожного рядка двовимірного масиву.
12. Розробити метод-член класу для створення нового одновимірного масиву з непарних максимальний елементів кожного стовпця двовимірного масиву.
13. Розробити метод-член класу для створення нового одновимірного масиву з суми парних елементів кожного рядка.
14. Розробити метод-член класу формування одновимірного масиву, *i*-ий елемент якого є добутком усіх елементів *i*-го стовпчика двовимірного масиву.
15. Розробити метод-член класу для створення нового одновимірного масиву з суми усіх позитивних елементів верхньої і нижньої трикутних матриць заданого двовимірного динамічного масиву.
16. Розробити метод-член класу для створення нового одновимірного масиву, елементи якого є скалярним добутком першого стовпчика двовимірного масиву на кожний наступний.

**Завдання 3**

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.
2. Розробити функцію для визначення елемента кратного 7 і метод-член класу, який використовує цю функцію для визначення кількості і суми елементів даного одновимірного динамічного масиву, кратних семи.
3. Розробити функцію визначення елемента кратного 9 і член класу підпрограму, що використовує цю функцію для визначення кількості і суми елементів даного одновимірного динамічного масиву, кратних дев’яти.
4. Розробити функцію визначення елемента кратного 11 і метод класу, що використовує цю функцію для визначення кількості і суми елементів даного одновимірного динамічного масиву, кратних одинадцяти.
5. Розробити функцію визначення елемента кратного 12 і метод класу, що використовує цю функцію для визначення кількості і суми елементів даного одновимірного динамічного масиву, кратних дванадцяти.
6. Розробити функцію визначення кубічного кореня з заданого члена і метод класу, що використовує цю функцію для перетворення кожного елемента одновимірного динамічного масиву.
7. Розробити функцію визначення квадратного кореня з заданого члена і метод класу, що використовує цю функцію для перетворення кожного парного елемента одновимірного динамічного масиву.
8. Розробити функцію визначення а3 для заданого елемента а і метод класу, що використовує цю функцію для перетворення кожного елемента заданого одновимірного динамічного масиву.
9. Розробити функцію визначення а2 для заданого елемента а і метод класу, що використовує цю функцію для перетворення кожного елемента заданого одновимірного динамічного масиву.
10. Розробити функцію визначення парного елемента і метод, що використовує цю функцію для визначення суми, кількості і середнього арифметичного всіх парних елементів динамічного масиву.
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.
12. Розробити функцію для визначення елемента кратного 4 і метод-член класу, який використовує цю функцію для визначення кількості і суми елементів даного одновимірного динамічного масиву, кратних чотири.
13. Розробити функцію для визначення логарифма елемента і метод-член класу, який використовує цю функцію для перетворення одновимірного масиву.
14. Розробити функцію визначення оберненого значення з заданого елемента і метод класу, що використовує цю функцію для перетворення кожного парного елемента одновимірного динамічного масиву.
15. Розробити функцію визначення функції sign для заданого елемента і метод класу, що використовує цю функцію для перетворення кожного парного елемента одновимірного динамічного масиву.
16. Розробити функцію для визначення елемента кратного 6 і метод-член класу, який використовує цю функцію для визначення кількості і суми елементів даного одновимірного динамічного масиву, кратних шести.

**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:

**сlass ім’я\_класу { список\_компонентів класу};**

де сlass – 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 (единою 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 CTіme, що містить дані-члени класу: year, month, day, hour, minute і функції-члени класу для їхнього введення, виведення та обробки. Структура класу наведена нижче.

classCTime{

*//* 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 publіc - 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 CTіme - 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 декілько об’єктів одного класу CTіmе:

CTіme today;

CTіmе tomorrow;

CTіmе yesterday;

то для цих об'єктів функцію Dіsplay( ) можна викликати таким чином: today.Dіsplay(); - виклик функції об'єктуtoday;

tomorrow.Dіsplay();  ***-*** виклик функції об'єктуtomorrow;

yesterday.Dіsplay(); *-* виклик функції об'єктуyesterday*;*

**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: prіvate - доступ закритий*;* publіc **-** доступ відкритий.

Звичайно prіvate 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 prіvate бажано. В класі можуть бути декілько секцій відкритих і закритих компонентів класу, що розташовуються в довільному порядку

Для використання закритих (prіvate) даних-членів існує лише один спосіб – через виклик відкритих для зовнішнього середовища функцій-членів класу, тому що закриті (prіvate) 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 використовуєть спеціфікатори доступу private і public

**Приклад**

# include < iostream.h >

Специфікатори

доступа

 #include <stdio.h>

class CTime {

private:

int year;

** int month*;* закриті дані члени

** 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();

*Оператор дозволу області видимості*

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;

}

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,тобтоякщо 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.

Вбудована

функція- член класу

рublic:

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

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 mіnutes 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:  6  Enter numnbers of array:  array[0] = 5  array[1] = -3  array[2] = -12  array[3] = 0  array[4] = 3  array[5] = -1 | Numbers of array:  array[0] = 5  array[1] = -3  array[2] = -12  array[3] = 0  array[4] = 3  array[5] = -1  Number of negative = 3  Amount of negative = -16  Negative numbers of array:  array [1]=-3  array [2]=-12  array [5]=-1 |

**Приклад виконання завдання 2**

**Варіант 2.** Розробити функцію для виводу на екран усіх негативних елементів кожного рядка двовимірного динамічного масиву і їх порядкових номерів.

Клас CArray\_2D в даному завданні має таку ж структуру, що і клас із завдання 1, відмінність є те, що в ньому використовується двомірний динамічний масив.

Структура класу:

class CArray\_2D{

int\*\* array;

int row, col;

public:

void Create\_ar2D(int rows, int cols);

void Clear();

void Input\_ar2D();

void Print\_ar2D();

void PrintNegative\_ar2D();

};

**Лістинг програми**

#include <iostream>

#include <malloc.h>

#include <stdlib.h>

using namespace std;

class CArray\_2D{

int\*\* array; //дані-члени класу

int row, col;

public:

//функції-члени класу

void Create\_ar2D(int rows, int cols);

void Clear();

void Input\_ar2D();

void Print\_ar2D();

void PrintNegative\_ar2D();

};

int main(){

int rows,colums;

cout << "Enter the rows of array:" << endl;

cin >> rows;

cout << "Enter the colums of array:" << endl;

cin >> colums;

CArray\_2D obj1;

obj1.Create\_ar2D(rows,colums);

obj1.Input\_ar2D();

obj1.Print\_ar2D();

obj1.PrintNegative\_ar2D();

system("pause");

obj1.Clear();

}

void CArray\_2D::Create\_ar2D(int rows, int cols)

{

row = rows;

col = cols;

//виділення пам’яті під двомірний масив

array = (int\*\*)malloc(row\*sizeof(int\*));

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

array[i] = (int\*)malloc(col\*sizeof(int));

}

void CArray\_1D::Clear(){free(array);}

void CArray\_2D::Input\_ar2D()

//функція введення елементів масиву

{

cout<<endl<< "Enter numbers of array:" <<endl;

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

{

for(int j=0; j<col; j++)

{

cout <<"array["<< i <<"]["<< j << "] = ";

cin >> array[i][j];

}

}

}

void CArray\_2D::Print\_ar2D()

//функція виводу на екран елементів масиву

{

cout << endl << "Numbers of array:" << endl;

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

{

for(int j=0; j<col; j++)

{

cout <<"array["<< i <<"]["<< j <<"] = " << array[i][j] << endl;

}

}

}

void CArray\_2D::PrintNegative\_ar2D()

//функція виводу від’ємних елементів масиву на екран

{

cout <<endl<<"Negative numbers of array:"<<endl;

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

{

for(int j=0; j<col; j++)

{

if(array[i][j] < 0)

cout<<"array["<<i<<"]["<<j<<"] = "<<array[i][j]<<endl;

}

}

}

**Тестування**

Для перевірки правильності роботи програми введемо в неї дані при введені, яких результат заздалегідь відомий. Наприклад сформуємо двомірний масив 4\*3 такого виду:

-4 -7 0

3 7 -14

5 -5 10

17 19 20

З введених даних наочно видно, що від’ємними є [0][0],[0][1],[1][2] і [2][1] елемент. Тепер введемо ці ж дані в програму і перевіримо результат.

|  |  |
| --- | --- |
| Enter the rows of array:  4  Enter the colums of array:  3  Enter numbers of array:  array[0][0] = -4  array[0][1] = -7  array[0][2] = 0  array[1][0] = 3  array[1][1] = 7  array[1][2] = -14  array[2][0] = 5  array[2][1] = -5  array[2][2] = 10  array[3][0] = 17  array[3][1] = 19  array[3][2] = 20 | negativne element  array[0][0] = -4  array[0][1] = -7  array[1][2] = -14  array[2][1] = -5 |

**Приклад виконання завдання 3**

**Варіант 2.** Розробити метод (функцію) визначення члена кратного 7 і метод, що використовує цю функцію для визначення кількості і суми елементів даного одномірного динамічного масиву, кратних 7.

Дана програма буде складатися з класу, що містить функцію виділення пам’яті під масив, функцію вводу даних, функцію виводу на екран всіх елементів масиву, функцію для знаходження елементів кратних 7 і функцію для виводу результатів на екран, функцію,що повертає суму елементів кратних 7 та функцію, що повертає кількість елементів, що кратні 7.

Клас буде мати наступну структуру:

class CArray\_1D{

int\* array;

int size;

public:

void Create\_ar1D(int n);

void Clear();

void Input\_ar1D();

void Print\_ar1D();

void PrintKr7\_ar1D();

int SumKr7\_ar1D();

int NumKr7\_ar1D();

};

**Лістингї 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();

void PrintKr7\_ar1D();

int SumKr7\_ar1D();

int NumKr7\_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.NumKr7\_ar1D();

int negsum = obj1.SumKr7\_ar1D();

cout << endl << "Number = " << negnum << endl;

cout << endl << "Amount = " << negsum << endl;

obj1.PrintKr7\_ar1D();

system("pause");

obj1.Clear();

}

void CArray\_1D::Create\_ar1D(int n)

{

size = n;

//виділення пам’яті під масив

array = (int\*)malloc(size\*sizeof(int));

}

void CArray\_1D::Clear(){free(array);}

void CArray\_1D::Input\_ar1D()

{

cout << endl <<"Enter numnbers of array:"<< endl;

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

{

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;

}

}

int CArray\_1D::NumKr7\_ar1D()

//функція, що повертає кількість елементів кратних 7

{

int count = 0;

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

{

if(array[i]%7==0)

count++;

}

return count;

}

int CArray\_1D::SumKr7\_ar1D()

//функція, що повертає суму елементів кратних 7

{

int sum = 0;

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

{

if(array[i]%7==0)

sum = sum + array[i];

}

return sum;

}

void CArray\_1D::PrintKr7\_ar1D()

//функція виводу на екран елементів кратних 7

{

cout << "Numbers of array/7:" << endl;

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

{

if(array[i]%7==0)

cout<<"array["<<i<<"]="<<array[i]<<endl;

}

}

Тестування

Для перевірки правильності роботи програми введемо в неї дані при введені, яких результат заздалегідь відомий. Наприклад сформуємо масив з 8 елементів такого виду:

21 -63 1 5 15 7 49 40

З введених даних наочно видно, що кратними 7 є 1,2,6 і 7 елемент.

Сума їх =14. Кількість 4.

Тепер введемо ці ж дані в програму і перевіримо результат. Нижче наведені результати тестування.

|  |  |
| --- | --- |
| Enter the size of array:  k= 8  Enter numnbers of array:  array[0]=21  array [1]=-63  array [2]=1  array [3]=5  array [4]=15  array [5]=7  array [6]=49  array [7]=40 | Numbers of array:  array[0]=21  array [1]=-63  array [2]=1  array [3]=5  array [4]=15  array [5]=7  array [6]=49  array [7]=40  Number = 4  Amount = 14  Numbers of array/7:  array[0] = 21  array[1] = -63  array[2] = 7  array[3] = 49 |

**Контрольні запитання**

* 1. Що таке об’єкт та клас?
  2. Структура класу в С++?
  3. Формат типу class в С++.
  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?