МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет "Запорізька політехніка"

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт із дисципліни
"Програмування"

для студентів спеціальності 123 "Комп'ютерна інженерія"
усіх форм навчання

Робота з класами в C++

2021

Методичні вказівки до виконання лабораторних робіт із дисципліни "Програмування" для студентів спеціальності 123 "Комп'ютерна інженерія" усіх форм навчання. Робота з класами в С++ / Укл.: М.В. Єфименко, Н.В. Луценко. – Запоріжжя: НУ "Запорізька політехніка", 2021. – 22 с.

Укладачі:

М.В.Єфименко, к.т.н., доцент
Н.В. Луценко, ст. викладач

Рецензент:

Р.К.Кудерметов, к.т.н., доцент

Відповідальний за випуск

Н.В. Луценко, ст. викладач

Затверджено
на засіданні кафедри КСМ
Протокол № 6 від 05.02.2021

Затверджено
на засіданні НМК КНТ
Протокол № 7 від 26.02.2021

ЗМІСТ

ПОРЯДОК ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБІТИ

Лабораторна робота складаються з декількох програм, завдань до них і контрольних питань та одного завдання для самостійної роботи. Студент повинен розробити, відпрацювати і зберегти у робочому каталозі приведені програми й програми, що розроблені згідно заданого варіанта, та проаналізувати результати їхньої роботи. За результатами виконання лабораторної роботи студент складає й захищає письмовий звіт, вимоги до якого викладені нижче. Завдання для самостійної роботи оформлюється в вигляді окремого звіту.

Зміст звіту

1. Титульний лист, виконаний відповідно до СТП.
2. Мета роботи.
3. Відповіді на контрольні питання.

ЗАГАЛЬНІ ВІДОМОСТІ

Клас – це похідний структурований тип, що дозволяє задати деяку структуровану сукупність даних і визначити набір операцій над цими даними.

Оголошення класу починається з ключового слова (**`class`**, **`struct`**, **`union`**), після йде ім'я класу і тіло класу в фігурних дужках.

Синтаксис класу наступний:

```
ключ_класу ім'я_класу {компоненти_класу};
```

де **`ключ-класу`** – ключове слово **`class`**, **`struct`**, **`union`**;

`ім'я_класу` – будь-який допустимий ідентифікатор;

`компоненти_класу` – визначення і опис типізованих даних і функцій.

Компонентами класу можуть бути змінні, функції, класи. Змінні класу називаються елементами класу, або компонентними даними. Функції класу називаються методами, членами-функціями, або компонентними функціями.

Розглянемо визначення класу з використанням службового слова **`class`**. Всі елементи і методи класів мають права доступу. Клас створює різні рівні доступу: **`private`**, **`public`**, **`protected`**.

За замовчуванням, весь вміст класу є доступним для читання і запису тільки для нього самого. Для доступу до даних класу ззовні використовують модифікатор доступу **`public`**. Всі функції і змінні, які знаходяться після модифікатора **`public`**, стають доступними з усіх частин програми.

Закриті дані класу розміщуються після модифікатора доступу **private**. Якщо модифікатор **public** відсутній, то всі функції і змінні, за замовчуванням є закритими.

Члени **protected** (захищені) використовуються в разі побудові ієрархії класів (при спадкуванні). Використання доступу protected без похідних класів еквівалентно специфікаторами **private**.

Методи класу (компонентні функції) обов'язково повинні бути описані в тілі класу. Метод має доступ до всіх компонентів класу (з будь-яким рівнем доступу). Якщо визначення методу повністю розміщено в тілі класу, то цей метод за замовчуванням вважається підставляємим (**inline**). Це не завжди зручно. Щоб цього уникнути, в тілі функції прописують прототип методу, а сам метод визначають поза класу. При цьому потрібно вказати, до якого класу належить цей метод. Для цього використовується операція уточнення **::**.

Прототип функції в тілі класу:

```
тип ім'я_методу_класу(сигнатура);
```

Визначення функції поза тілом класу:

```
тип ім'я_класу::ім'я_методу_класу(сигнатура){тіло методу}
```

Для визначення об'єкта класу використовується конструкція:

```
ім'я_класу ім'я_об'єкта;
```

Звернення до компонентів класу:

```
ім'я_об'єкта.ім'я_класу::ім'я_компонента
```

Ім'я-класу з операцією уточнення '**::**' може бути опущено і для доступу до відкритих даних використовується уточнене ім'я:

```
ім'я_об'єкта.ім'я_елемента
```

Звернення до відкритих методів класу (до компонентних функцій):

```
ім'я_об'єкта.звернення_до_методу
```

Всі дії з закритими елементами класу реалізуються через його методи. Наприклад:

```
class CUB
{ float x;  //Закритий елемент класу:  сторона куба
 public:  //  Відкриті елементи класу
  float S,V;  //  площа, об’єм  куба
  void Put_a(float a){x=a;} //  Метод завдання сторони куба
  float Get_a() {return x;} //  Метод повернення сторони куба
  void obiem(){V=x*x*x;} //  Метод обчислення об’єму куба
  void area(){S=6*x*x;} //  Метод обчислення площі куба
};
```

```
int main()
{CUB a1; // визначення об'єкта класу
 a1.Put_a(5.0);  // Звернення до методу класу Get_a()
//Доступ до закритого елементу класу x через метод Get_a()
 cout<<"x="<<a1.Get_a()<<endl;
 a1.obiem();   // Звернення до методу класу obiem()
 a1. area();    // Звернення до методу класу area()
 cout<<"V="<<a1.V;  // Доступ до відкритого елементу класу V
 cout<<" S="<<a1.S; // Доступ до відкритого елементу класу S
 return 0;}
```

При оголошенні класу елементи класу не ініціалізуються. Для ініціалізації об'єктів класу призначено конструктор. Конструктор класу – це спеціальна функція, яка автоматично викликається відразу після створення об'єкта цього класу. Він не має типу – значення, що повертається (навіть **void**), і називається так само, як клас, в якому він знаходиться. Якщо визначено конструктор з параметрами, то в об'єкті список фактичних параметрів не повинен бути порожнім.

Синтаксис визначення конструктора:

```
ім'я_класу(список_формальних_параметрів)
{Тіло конструктора}
```

Наприклад:

```
CUB(float a){x=a}  //визначення конструктора класу CUB
CUB r1(10.0); // визначення об'єкта r1 класу CUB
```

Конструктори можна ініціалізувати. Наприклад:

```
CUB(float a):x(a){ }// ініціалізація
```

Клас може мати конструктор з параметрами за замовчуванням і перевантажені конструктори.

Конструктор з параметрами за замовчуванням - це конструктор, який має в списку формальних параметрів значення за замовчуванням і може викликатися без аргументів. Наприклад:

```
CUB(float a=10.0){x=a}
```

Перевантажені конструктори - це конструктори з різними параметрами для різних видів ініціалізації, розрізняються за типом або кількості переданих параметрів. Наприклад:

```
CUB(float a, float b){x=a, y=b}
CUB(float a){x=a}
```

Деструктор – це особливий вид методу, що застосовується для звільнення пам'яті, що займає об'єкт класу.

Деструктор класу викликається при знищенні об'єкта. Ім'я деструктора аналогічно імені конструктора, тільки на початку ставиться знак тильди ~. Деструктор не має вхідних параметрів і значення, що повертається (навіть **void**). Деструктор викликається автоматично, коли об'єкт виходить з області видимості (наприклад, при завершенні програми). Якщо деструктор явно не визначений, компілятор автоматично створює порожній деструктор. Наприклад:

```
~CUB(){cout<<"end\n";}
```

У класі можна описати статичні елементи і методи класу. Їх можна розглядати як глобальні змінні або функції, які доступні тільки в межах області класу.

Статичні дані відносяться до всіх об'єктів класу. Такі дані використовуються: 1) для контролю загальної кількості об'єктів класу; 2) для одночасного доступу до всіх об'єктів або частини їх.

Статичний член класу описується за допомогою слова **static** і створюється на весь клас. Оголошення статичного члена в класі:

```
static тип ім'я_статичного_члена;
```

Після оголошення в класі статичний член класу необхідно ініціалізувати в глобальній області:

```
тип ім'я_класу::ім'я_статичного_члена=ініціалізатор;
```

Статичні поля доступні як через ім'я класу, так і через ім'я об'єкта:

```
Ім'я_класу ::ім'я_статичного_члена_класу;
Ім'я_об'єкта.ім'я_статичного_члена_класу;
```

Наприклад:

```
class AAA
{public:
 static int kol; // оголошення статичного члена в класі
};
int AAA:: kol=0; // Визначення в глобальній області
int main()
{ AAA a;
cout<<AAA::kol;  // доступ до статичного члена
cout<<a.kol;  ...}
```

Статичні методи призначені для звернення до статичних членів класу. Вони можуть звертатися безпосередньо тільки до статичних членам і викликати тільки інші статичні методи класу. Звернення до статичних методів проводиться так само, як до статичних членам - або через ім'я класу, або, якщо хоча б один об'єкт класу вже створений, через ім'я об'єкта.

ЛАБОРАТОРНА РОБОТА

Введення у класи

Мета роботи – вивчити структурований похідний тип мови С++ – класи, навчитися визначати методи класу, конструктори, деструктори і статичні члени класу, дізнатися про засоби доступності до компонентів класу.

Програма 1

Створено клас **RECTANGLE** (Прямокутник), закритими елементами якого є довжина, ширина і периметр, а відкритими – площа прямокутника. Клас має конструктор (ініціалізуються довжина і ширина) і методи для обчислення периметру та площі і повернення закритих елементів (довжини, ширини, периметру). У головній функції main**()** визначено два екземпляри класу (об'єкти) і покажчик. Показані методи доступу до членів класу.

```
#include <iostream>
#include <stdlib.h>
using namespace std;
class RECTANGLE
{// Закриті члени класу:
   float x,y;  // довжина і ширина
   float P;    // периметр
 public:  // Відкриті члени класу:
   float S;// площа
 RECTANGLE(float a=1.0, float b=1.0) //Конструктор
   {x=a; y=b;}
// Прототипи методів класу
 float Get_a();  // повернення довжини
 float Get_b();  // повернення ширини
 float Get_P();  // повернення периметру
 void area();    // обчислення площі
 void perim();   // обчислення периметру
};
// визначення методів класу
float RECTANGLE::Get_a(){return x;}//повернення довжини
float RECTANGLE::Get_b(){return y;}//повернення ширини
float RECTANGLE::Get_P(){return P;}//повернення периметру
void RECTANGLE::area(){S=x*y;} //обчислення площі
void RECTANGLE::perim() //обчислення периметру
  {P=2*(x+y);}
```

```
int main()
{RECTANGLE r1(5.5,2.0); //перший об'єкт класу
 r1.area();    // обчислення площі прямокутника
 r1.perim();   // обчислення периметру прямокутника
 //виведення на екран результатів
 cout<<"  Rectangle_1\n";
 cout<<"length="<<r1.Get_a();
 cout<<" width="<<r1.Get_b()<<endl;
 cout<<"area="<<r1.S<<endl;
 cout< <"perimeter="<<r1.Get_P()<<endl;
 RECTANGLE r2(6.3,5.0);// другий об'єкт класу
 RECTANGLE *pr2=&r2; );//визначення покажчика
 pr2->area();   //обчислення площі прямокутника
 pr2->perim();  //обчислення периметру прямокутника
  //виведення на екран результатів
 cout<<"  Rectangle_2\n";
 cout<<"length="<<pr2->Get_a();
 cout<<" width="<<pr2->Get_b()<<endl;
 cout<<"area="<<pr2->S<<endl;
 cout <<"perimeter="<<pr2->Get_P()<<endl;
 system("pause");
 return 0;
}
```

Завдання до програми 1. Розробіть клас з методами відповідно до варіанту. У головній функції **main()** визначить два об'єкти класу (екземпляри) з різними параметрами. Застосуєте до класу операцію **sizeof()** і поясніть результат. Продемонструйте роботу всіх методів класу.

1. Розробіть клас **CIRCLE** (Коло). Конструктор повинен ініціалізувати радіус кола. Клас повинен мати методи обчислення діаметра, довжини кола, площі круга і об'єму кулі. Радіус і довжина кола повинні бути закритими членами, а діаметр, площа кола і об'єм кулі – відкритими.

2. Розробіть клас **SQUARE** (Квадрат). Конструктор повинен ініціалізувати довжину сторони квадрата. Клас повинен мати методи обчислення діагоналі, площі, периметра квадрата і об'єму куба. Довжина сторони і площа квадрата повинні бути закритими членами, а діагональ, периметр квадрата і об'єм куба – відкритими.

3. Розробіть клас **RECTAN_TRIANGLE** (Прямокутний трикутник). Конструктор повинен ініціалізувати довжини двох катетів. Клас повинен мати методи обчислення гіпотенузи, площі і периметра трикутника. Довжина катетів і площа трикутника повинні бути закритими членами, а гіпотенуза, периметр трикутника – відкритими.

4. Розробіть клас **RHOMB** (Ромб). Конструктор повинен ініціалізувати довжини діагоналей ромбу. Клас повинен мати методи обчислення сторін, площі і периметра ромбу. Довжина катетів і площа ромбу повинні бути закритими членами, а сторона, периметр ромбу – відкритими.

5. Розробіть клас **ISOSC_TRIANGLE** (Рівнобедрений трикутник). Конструктор повинен ініціалізувати довжини підстави та висоти трикутника. Клас повинен мати методи обчислення других сторін, площі і периметра трикутника. Довжини підстави, висоти і площа трикутника повинні бути закритими членами, а сторона, периметр трикутника – відкритими.

Програма 2

У програмі демонструється клас для створення матриць в динамічної пам'яті. Клас має конструктор з параметрами за замовчуванням: ініціалізуються розміри матриці. Матриця моделюється за допомогою одновимірного масиву. В конструкторі виділяється динамічна пам’ять для матриці і елементам матриці привласнюються випадкові значення. Клас має метод виведення значень елементів матриці на екран. Деструктор призначено для звільнення пам’яті.

```
#include <iostream>
#include <stdlib.h>
#include <ctime>
using namespace std;
class Matrix
{int m,n;
 double *matr;
public:
 Matrix(int, int =1);
~Matrix()
 {delete [] matr; cout<<"matr delete\n";}
 void display();
};
```

```
Matrix::Matrix(int M, int N) //конструктор
{ m=M; n=N;
 if((matr=new double[n*m])==NULL)  exit(1);
 for(int i=0,j;i<m;i++)
  for(j=0;j<n;j++)
   *(matr+i*n+j)=((double)rand()/RAND_MAX)*100-50;
}
void Matrix::display() //метод виведення матриці на екран
{ for(int i=0,j;i<m;i++)
  {for(j=0;j<n;j++)
    {cout.width(10);
     cout<< *(matr+i*n+j);  }
   cout<<"\n";
  }
 cout<<"\n";
 system("pause");}
int main()
{srand(time(NULL));
 Matrix A(2,2),B(4,5); //визначено два об'єкти класу
 Matrix *pMatr=&B; //визначено покажчик
 Matrix Vec(5); //визначено об'єкт класу,
 cout<<"The matrix A:\n";
 A.display(); //виведення матриці на екран
 cout<<"The matrix B:\n";
 pMatr->display(); //виведення матриці на екран
 cout<<"The vector Vec:\n";
 Vec.display(); //виведення вектору на екран
 system("pause");
 return 0;
}
```

Завдання до програми 2. Допрацюйте клас **Matrix**. Додайте метод введення елементів матриці (вектора) з клавіатури та методи, відповідно до варіанту. Реалізуйте деструктор поза описом класу. У головній функції **main()** визначте три об'єкти (екземпляри) класу з різними параметрами. Елементи одного об'єкту введіть з клавіатури. Продемонструйте роботу всіх методів класу.

1. Розробіть метод класу для обчислення суми всіх елементів матриці та метод для складання кожного з елементів матриці з заданим числом.

2. Розробіть метод класу для обчислення максимального елемента матриці і метод для обчислення мінімального елемента матриці.

3. Розробіть метод класу для обчислення добутку всіх елементів матриці в діапазоні від 1 до 30, та метод для множення кожного з елементів матриці на задане число.

4. Розробіть метод класу для обчислення кількості додатних елементів матриці та метод для обчислення суми додатних елементів матриці.

5. Розробіть метод класу для обчислення суми від'ємних елементів матриці та метод для обчислення кількості від'ємних елементів матриці.

Додаткове завдання. Розробіть метод класу, який заповнює матрицю випадковими числами в заданому діапазоні

Програма 3

У програмі показано створення масиву об'єктів класу **Worker** (Працівник) з конструктором та зі статичним членом класу. Конструктор класу ініціалізує початковий капітал працівника – член класу **MoneyOfWorker**. За допомогою статичного члена класу **MoneyOfFirm** задається капітал фірми. Метод класу **work()** в залежності від тривалості роботи (**period**) збільшує капітали працівника та фірми. Коефіцієнти збільшення капіталу працівника і фірми задані константами, відповідно 2 та 10. Методи класу **display()** та **display_Firm()** виводять на екран відповідно капітал працівника і фірми.

```
#include <iostream>
#include <cstdlib>
using namespace std;
class Worker
 {float MoneyOfWorker; // Капітал працівника
 //оголошення статичного члена класу, капітал фірми
  static float MoneyOfFirm;
 public:
  Worker(float m){MoneyOfWorker=m;} // конструктор
//прототип методу підрахунку капіталу працівника і фірми
  void work(float);
  void display() // метод виведення капіталу працівника
   {cout<<MoneyOfWorker<<endl;}
  void display_Firm() //метод виведення капіталу фірми
    {cout<<MoneyOfFirm<<endl;}
};
```

```
// метод підрахунку капіталу працівника і фірми
 void Worker::work(float period)
 {MoneyOfWorker+=2*period;
  MoneyOfFirm+=10*period;
 }
//ініціалізація статичного члена
float Worker::MoneyOfFirm=1000;
int main()
{const int n=5;// кількість працівників
//Визначення масиву об'єктів класу і їх ініціалізація
 Worker M[n]={50,28,35,45,49};
 cout<<"Initial worker's money:\n";
 for(int i=0;i<n;i++)
   {cout<<"worker "<<i<<": ";
    M[i].display();} // виведення капіталу працівників
   }
 cout<<"Initial company's money:\n";
 M[0].display_Firm(); // виведення капіталу фірми
 cout<<"\nFinal worker's money:\n";
 for(int i=0;i<n;i++)
  { M[i].work(10); // period=10
    cout<<"worker "<<i<<": ";
    M[i].display(); // виведення капіталу а працівників
  }
 cout<<"Final company's money:\n";
 M[0].display_Firm();// виведення капіталу фірми
 system("pause");
 return 0;
}
```

Завдання до програми 3. Допрацюйте клас **Worker** відповідно до варіанту. Визначення конструктора зробіть зовнішнім з параметрами за замовчуванням. Продемонструйте роботу всіх методів класу.

1. Додайте в клас **Worker** ще один метод - **input_W()**, який дозволяє вводити значення початкового капіталу кожного працівника.

2. У головній функції **main()** визначить та проініціалізуйте масив періодів роботи для кожного працівника. Метод **work()** класу **Worker** повинен збільшувати капітали робітника і фірми в залежності від періоду роботи.

3. У головній функції **main()** визначить та введіть з клавіатури період роботи, визначить та проініціалізуйте масив тривалості

відпочинку для кожного працівника. Метод **work()** класу **Worker** повинен збільшувати капітали робітника і фірми в залежності від періоду роботи та зменшувати в залежності від тривалості відпочинку кожного працівника.

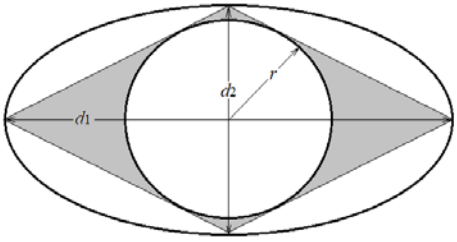
4. Введіть в клас **Worker** нові змінні **float k_W, k_F**, що позначають відповідно коефіцієнти збільшення капіталу працівника і капіталу фірми. Додайте в клас **Worker** методи завдання цих коефіцієнтів. Метод **work()** класу **Worker** повинен нараховувати збільшення капіталів робітника і фірми в залежності від заданих коефіцієнтів.

5. Введіть в клас **Worker** нову змінну **float effic** (від 0.5 до 1.5), яка враховує продуктивність кожного працівника. Допрацюйте конструктор, він повинен ініціювати капітал і продуктивність кожного працівника. Метод **work()** класу **Worker** повинен враховувати продуктивність кожного працівника.

Програма 4

Призначена для самостійної роботи. Треба розробити програму згідно варіанту. Програма оформлюється в вигляді звіту. В звіті приводяться формули розрахунків та текст програми з коментарями.

Приклад. В ромб з діагоналями *d*1, *d2* вписано коло, а навколо ромба описано еліпс. Треба визначити периметр та площу ромба, довжину та площу кола, периметр та площу еліпса (див. рисунок). Розробіть клас **Rhomb** (ромб). Конструктор класу повинен ініціалізувати довжини діагоналей ромбу *d*1, *d*2.



Клас повинен мати методи обчислення сторони ромба *a*, периметра *P_romb* і площі *S_romb* ромба, радіуса *r*, довжини *L_cir* та площі *S_cir* кола, периметра *P_el* і площі *S_el* еліпса. У головній функції **main()** визначте об'єкт класу **Rhomb**. Результати виведіть на екран з коментарями.

Формули для методів програми

Діагоналі ромба: d_1, d_2.

Довжина сторони ромба: $a = \sqrt{(d_1/2)^2 + (d_2/2)^2} = \frac{\sqrt{d_1^2 + d_2^2}}{2}$.

Периметр ромба: $P = 4a = 2\sqrt{d_1^2 + d_2^2}$.

Площа ромба: $S = \frac{d_1 d_2}{2}$.

Радіус кола, що вписане в ромб: $r = \frac{d_1 d_2}{4a} = \frac{d_1 d_2}{2\sqrt{d_1^2 + d_2^2}}$

Довжина кола: $L = 2\pi r$.

Площа кола: $S = \pi r^2$.

Периметр еліпса: $P = 2\pi\sqrt{(d_1^2 + d_2^2)/8} = \pi\sqrt{(d_1^2 + d_2^2)/2}$.

Площа еліпса: $S = \pi \frac{d_1 d_2}{4}$.

Лістинг програми

```
#include <iostream>
#include <stdlib.h>
#include <math.h>
using namespace std;

class RHOMB
{// Закриті члени класу:
 float d1,d2; // діагоналі ромба
 float x; // сторона ромба
 float r; //  радіус кола
public: // Відкриті члени класу:
 float P_r, L_cir, P_el;//периметри ромба, кола, еліпса
 float S_r, S_cir, S_el; //площі ромба, кола, еліпса

RHOMB(float a=2.0, float b=1.0)//конструктор
 {d1=a; d2=b;} // ініціалізування діагоналей ромба
// Визначення методів
 float Get_d1()// повернення діагоналі_1
{return d1;}
 float Get_d2()// повернення діагоналі_2
{return d2;}
```

```
 float Get_x()// повернення сторони ромба
{return x;}
 float Get_r()// повернення радіуса кола
{return r;}

// Прототипи методів
 void side_x();   // обчислення сторони ромба
 void perim_Pr(); // обчислення периметру ромба
 void area_Sr();  // обчислення площі ромба
 void side_r();   // обчислення радіуса кола
 void perim_Pc(); // обчислення довжини кола
 void area_Sc();  // обчислення площі кола
 void perim_Pel(); // обчислення периметру еліпса
 void area_Sel();  // обчислення площі еліпса
};

// Визначення методів
void RHOMB::side_x()  // обчислення сторони ромба
  {x=(sqrt(d1*d1+d2*d2))/2;}
void RHOMB::perim_Pr() // обчислення периметру ромба
  {P_r=4*x;}
void RHOMB::area_Sr() // обчислення площі ромба
  {S_r=(d1*d2)/2;}
void RHOMB::side_r() // обчислення радіуса кола
  {r=(d1*d2)/(2*(sqrt(d1*d1+d2*d2)));}
void RHOMB::perim_Pc() // обчислення довжини кола
  {L_cir=2*M_PI*r;}
void RHOMB::area_Sc() // обчислення площі кола
  {S_cir=M_PI*r*r;}
void RHOMB::perim_Pel() // обчислення периметру еліпса
  {P_el=M_PI*(sqrt((d1*d1+d2*d2)/2));}
void RHOMB::area_Sel() // обчислення площі еліпса
  {S_el=M_PI*d1*d1/4;}

int main()
{ RHOMB r1(8.0,6.0);  //об'єкт з діагоналями 8.0, 6.0
  r1.side_x();   // обчислення сторони ромба
  r1.perim_Pr(); // обчислення периметру ромба
  r1.area_Sr();  // обчислення площі ромба
//Виведення результатів
  cout<<"Rhomb_1\n";
  cout<<"diagonal_1="<<r1.Get_d1();
```

```
  cout<<" diagonal_2="<<r1.Get_d2();
  cout<<"\nside_x="<<r1.Get_x();
  cout<<"\nperimeter="<<r1.P_r<<" area="<<r1.S_r<<endl;
  r1.side_r();   // обчислення радіуса кола
  r1.perim_Pc(); // обчислення довжини кола
  r1.area_Sc();  // обчислення площі кола
//Виведення результатів
  cout<<"\nCIRCLE_1\n";
  cout<<"side_r="<<r1.Get_r();
  cout<<"\nperimeter="<<r1.L_cir;
  cout<<"  area="<<r1.S_cir<<endl;
  r1.perim_Pel(); // обчислення периметру еліпса
  r1.area_Sel(); // обчислення площі еліпса
//Виведення результатів
  cout<<"\nEllipse_1\n";
  cout<<"perimeter="<<r1.P_el;
  cout <<" area="<<r1.S_el<<endl;
  system("pause");
  return 0;
}
```

Отримані результати на екрані:

```
Rhomb_1
diagonal_1=8 diagonal_2=6
side_x=5
perimeter=20 area=24

CIRCLE_1
side_r=2.4
perimeter=15.0796 area=18.0956

Ellipse_1
perimeter=22.2144 area=50.2655
```

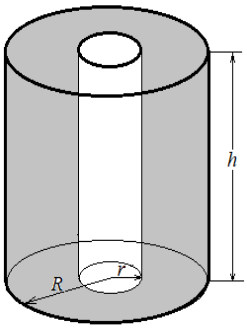
Завдання до програми 4

Варіант 4.1

В циліндрі із радіусом підстави *R* та висотою *h* через центр висвердлюється циліндричний отвір з радіусом *r*. Треба визначити площу і об'єм циліндра і отвору, а також площу і об'єм отриманого геометричного тіла (циліндра з отвором), площа якого повинна включати площу бічної поверхні всередині отвору (див. рисунок).

Розробіть клас **`Cylinder`** (циліндр). Конструктор повинен ініціалізувати радіус циліндра *R* та висоту циліндра *h*. Клас повинен мати наступні методи: методи обчислення загальної площі *S_cyl* і об'єму *V_cyl* циліндра, метод завдання радіуса отвору *r*, методи обчислення загальної площі *S_slot* та об'єму *V_slot* отвору, методи обчислення площі *S_dril* та об'єму циліндра *V_dril* з отвором.

У головній функції **`main()`** визначте два об'єкти класу **`Cylinder`** з різними даними. Результати з коментарями виведіть на екран.



Тест

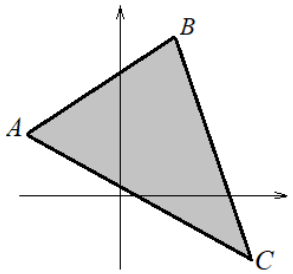
Дано: R = 10.0 см,
h = 20.0 см, r = 2.0 см
Результат:
S_cyl = 1884.96 см^2
V_cyl = 6283.184 см^3
S_slot = 276.46 см^2
V_slot = 251.327 см^3
S_dril = 2111.15 см^2
V_dril = 6031.857 см^3

Варіант 4.2

На площині задані координати трьох точок *A, B, C,* які з'єднуються прямими. При з'єднанні цих точок утворюється трикутник *ABC* або пряма. Для трикутника треба визначити довжини сторін, периметр та площу. Для прямої треба визначити довжини сторін та загальну довжину прямої (див. рисунок).

Розробіть клас **`Points`** (точки). Конструктор класу повинен ініціалізувати координати трьох точок: точка *A* з координатами *Xa, Ya,* точка *B* з координатами *Xb, Yb*, точка *C* з координатами *Xc, Yc*. Клас повинен мати методи обчислення довжину сторін трикутника *Lab, Lbc, Lac,* площі трикутника *S_abc*. А також мати метод визначення, чи є отримана фігура трикутником (для нього обчислити периметр *P_abc*), чи це пряма (для неї обчислити довжину).

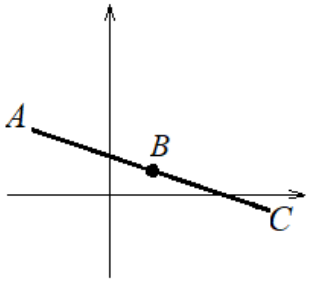
У головній функції **`main()`** визначте три об'єкти класу **`Points`** з різними даними (два об'єкти – трикутники та один – пряма). Результати з коментарями виведіть на екран.



Тест 1

Дано: *Xa* = –5.0, *Ya* = 3.0;
Xb = 3.0, *Yb* = 8.0;
Xc = 6.*0*, *Yc* = –4.0

Результат:
This is a triangle:
Lab=9.434, Lbc=12.369, Lac=13.038,
P_abc=34.842
S_abc = 55.5 см^2



Тест 2

Дано: *Xa* = –4.0, *Ya* = 3.0;
Xb = 1.0, *Yb* = 1.0;
Xc= 6.*0*, *Yc* = –1.0

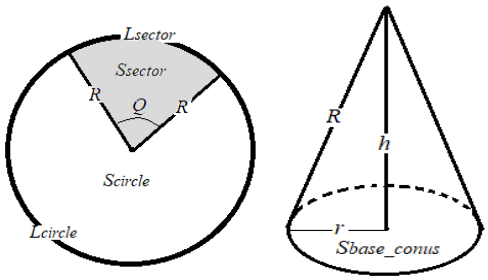
Результат:
This is a line:
Lab=5.385, Lbc=5.385, L_ac=10.77
Length_line Lac=10.77, *S_abc* = 0

Варіант 4.3

З кола радіусом *R* сантиметрів вирізується сектор із кутом *Q* градусів. З вирізаного сектора виходить конус. Треба визначити площу підстави і об’єм отриманого конуса (див. рисунок).

Розробіть клас **`Conus`** (конус). Конструктор класу повинен ініціалізувати радіус кола *R* та кут сектора *Q* , а також обчислювати коефіцієнт *k_sector*=*Q*/360. Клас повинен мати наступні методи: методи обчислення довжини *Lcircle* і площі кола *Scircle* з радіусом *R*, методи обчислення довжини дуги *Lsector* і площі сектора *Ssector* з кутом *Q*, методи обчислення площі підстави *Sbase_conus* і об’єму *Vconus* отриманого конуса.

В головній функції **`main()`** визначте три об’єкти класу **`Conus`** з різними даними. Результати з коментарями виведіть на екран.



Тест

Дано: *R*=20.0 *Q*=60 град

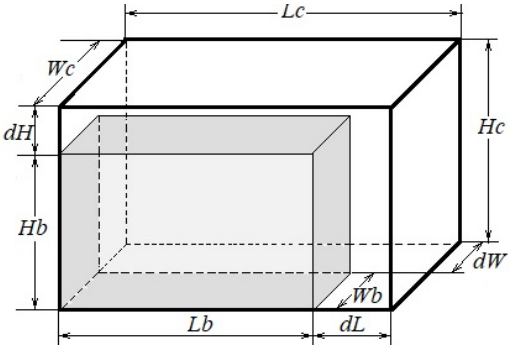
Результат:

Lcircle=125.664
Scircle=1256.64
Lsector=20.944
Ssector=209.44
Sbase_conus=34.9066
Vconus=229.456

Варіант 4.4

В контейнер з розмірами $x \times y \times z$ треба вкласти коробку з розмірами $a \times b \times c$. Треба визначити, чи можна вкласти коробку в контейнер, і на скільки розміри коробки більші або менші розмірів контейнеру (див. рисунок).

Розробіть клас **`BOX`** (коробка). Конструктор класу повинен ініціалізувати розміри контейнера та визначати довжину, ширину та висоту контейнера (*Lc*, *Wc*, *Hc*, при чому $Lc \geq Wc \geq Hc$), Клас повинен мати метод завдання розмірів коробки та визначення довжини, ширини та висоти коробки (*Lb*, *Wb*, *Hb*, при чому $Lb \geq Wb \geq Hb$), метод визначення, чи можна вкласти коробку в контейнер, метод визначення, на скільки розміри коробки більші або менші розмірів контейнеру (*dL*, *dW*, *dH*). В головній функції **`main()`** визначте три об'єкти класу **`BOX`** з різними даними. Результати з коментарями виведіть на екран.



Тест 1

Дано: *x*=0.3м *y*=0.8м *z*=0.4м
a=0.2м *b*=0.15м *c*=0.4м

Результат:

Container:
Lc=0.8м *Wc*=0.4м *Hc*=0.3м
Box:
Lb=0.4м *Wb*=0.2м *Hb*=0.15м
container fit for box
dL=0.4м *dW*=0.2м *dH*=0.15м

Тест 2

Дано: *x*=0.3м *y*=0.8м *z*=0.4м
a =0.4м *b* =0.5м *c* =0.6м
Результат:
Container:
Lc=0.8м *Wc*=0.4м *Hc*=0.3м
Box:
Lb=0.6м *Wb*=0.5м *Hb*=0.4м
container small for box
dL=0.2м *dW*= – 0.1м *dH*= – 0.1м

Варіант 4.5

Навколо газону в вигляді рівностороннього трикутника прокладена доріжка з бордюром з обох сторін. Треба визначити площу газону, площу доріжки, загальну довжину бордюру (см.рис. 7).

Розробіть клас **`Lawn`** (газон). Конструктор класу повинен ініціалізувати сторону *a* трикутника (газону) та ширину доріжки *w*, а також обчислювати сторону *b* трикутника (ділянки). Клас повинен мати методи обчислення периметру *P_lawn* та площі *S_lawn* газону, методи обчислення периметру *P_land* та площі *P_land* ділянки, методи обчислення площі доріжки *S_w* та довжини бордюру *P_border*. В головній функції **`main()`** визначте три об'єкти класу **`Lawn`** з різними даними. Результати з коментарями виведіть на екран.

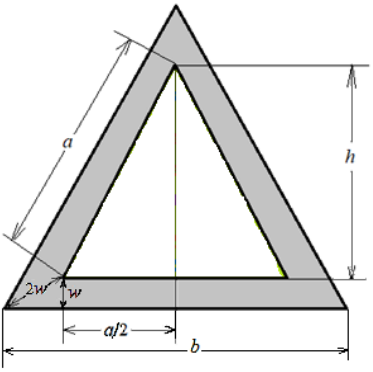


Рисунок 7

Тест

Дано:
a = 3.0 м *w* = 0.60 м
Результат:
b =5.0785 м
S_lawn = 3.897 м^2
P_lawn = 9.0 м
S_land = 11.1677 м^2
P_land = 15.2354 м
S_w = 7.271 м^2
P_ border = 24.2354 м

КОНТРОЛЬНІ ПИТАННЯ ДО ЛАБОРАТОРНОЇ РОБОТИ

1. Що таке клас в мові С ++?
2. Які є рівні доступу до даних класу?
3. Чим визначається розмір пам'яті, що займає об'єкт класу?
4. Як отримати доступ до відкритого елемента класу? Приклад
5. Як отримати доступ до закритого елементу класу? Приклад
6. Дайте визначення і призначення конструктора класу.
7. Які конструктори бувають? Чим відрізняються?
8. Дайте визначення і призначення деструктора класу.
9. Знайдіть помилки:

```
class ABC
{int a=3;
 int b, c;
 public:
 int ABC(int n=1) {b=n;}
 mul() {c=1+a*b;}
 ~ABC(int c){cout<<end\n";}
};
```

10. В якому випадку значення одного об'єкта можна привласнити іншому об'єкту?
11. Чим відрізняються статичні члени класу від звичайних?
12. Як визначити статичний член класу і отримати доступ до нього? Приклад.

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. Подбельский В.В. Язык Си++: Учеб. пособие. – М.: Финансы и статистика, 2007. – 560 с.
2. Павловская Т.А. С/С++. Программирование на языке высокого уровня. – СПб: Питер, 2009. – 464 с
3. Стивен Прата. Язык программирования С++. Лекции и упражнения. – 6-е изд. / Пер. с англ. – М.: Вильямс, 2012. – 478 с.
4. Х.М. Дейтел, П.Дж. Дейтел. Как программировать на С++. – 5-е изд. / Пер. с англ. – М.: ООО "Бином-Пресс", 2008.– 1456 с.
5. Шилдт, Герберт. Полный справочник по С++. – 4-е изд. / Пер. с англ. – М.: Вильямс, 2008. – 800 с.
6. Васильев А.К. Программирование на С++ в примерах и задачах. – СПб: Питер, 2017. –369 с.
7. Страуструп Б. Программирование. Принципы и практика с использованием С++.– 2 изд. / Пер. с англ. – М.: Вильямс, 2016. – 1329с.