Міністерство освіти і науки України

Сумський державний університет

Кафедра комп’ютерних наук

**Пояснювальна записка**

**до курсової роботи**

з дисципліни

«Основи об'єктно-орієнтованого програмування»

на тему: «Обчислення швидкості витікання ідеального газу із посудини під тиском з використанням мови програмування С++»

Викладач Тиркусова Н. В.

Студент Безрук В. М.

Група ІН−01/2

Варіант 3

Суми ̶ 2021

**ЗМІСТ**

[**1** **ПОСТАНОВКА ЗАДАЧІ** 3](#_Toc90506530)

[**2** **ТЕОРЕТИЧНИЙ МАТЕРІАЛ ІЗ ТЕМИ** 5](#_Toc90506531)

[**3** **ОПИС СТРУКТУРИ ДАНИХ ТА ВИМОГИ ДО НИХ** 8](#_Toc90506532)

[**4** **АЛГОРИТМ РОБОТИ ПРОГРАМИ** 10](#_Toc90506533)

[**5** **ОПИС ФУНКЦІЙ КОРИСТУВАЧА** 20](#_Toc90506534)

[**6** **ОПИС ФАЙЛІВ ТА ЇХ ПРИЗНАЧЕННЯ** 23](#_Toc90506535)

[**7** **СПИСОК ВИКОРИСТАНИХ БІБЛІОТЕК** 24](#_Toc90506536)

[**8** **ІНСТРУКЦІЯ ДЛЯ РОБОТИ З ПРОГРАМОЮ** 25](#_Toc90506537)

[**9** **ПРИКЛАД ТЕСТУВАННЯ ТА РЕЗУЛЬТАТИ РОБОТИ ПРОГРАМИ** 26](#_Toc90506538)

[**10** **ГРАФІК** 28](#_Toc90506539)

[**11** **КОД ПРОГРАМИ** 29](#_Toc90506540)

[**ВИСНОВОК** 37](#_Toc90506541)

[**СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ** 38](#_Toc90506542)

# **ПОСТАНОВКА ЗАДАЧІ**

*Мета роботи -* закріплення та поглиблення теоретичних знань, оволодіння практичними навичками щодо розроблення алгоритмів розв’язування задач за математичним описом для вирішення конкретних завдань, а також їх реалізація мовою програмування С++, закріплення навичок самостійної роботи шляхом пошуку, аналізу та активного засвоювання інформації із різних джерел.

**Завдання:**

Описати абстрактний клас «Об’єкт\_обчислення» з елементом даних «Обчислюваний\_параметр». Передбачити чисті віртуальні функції:

1) Введення даних.

2) Розрахунок обчислювального параметру.

3) Виведення вхідних даних і результату.

Описати клас «Ідеальний\_газ», похідний від абстрактного класу. В ньому додати елементи даних:

Крім віртуальних функцій передбачити перевантажені операції присвоєння, порівняння (), а також конструктори: пустий, з ініціалізацією і копіювання.

Привести приклад програми, в якій застосовуються усі конструктори, члени-функції і перевантажені оператори.

**Необхідно:**

Обрати один, будь- який параметр, що має вказаний діапазон значень, та протабулювати «Обчислювальний параметр» при зміні параметра від початкового значення до кінцевого з кроком Δ. Результати обчислень вивести у файл. Побудувати графік отриманої залежності.

**Формули для обчислення:**

Швидкість витікання ідеального газу із посудини під тиском обчислюється по формулі:

де: – універсальна газова константа;

– молекулярна маса газу;

*Т* – температура в посудині;

– тиск газу в посудині;

– зовнішній тиск.

де – відношення теплоємності при постійному тиску () і постійному об’ємі ().

**Вхідні дані:**

При виведенні даних слід мати на увазі:

Порядок значень інших параметрів:

Дані читати з файлу, а результат розрахунку вивести на екран.

# **ТЕОРЕТИЧНИЙ МАТЕРІАЛ ІЗ ТЕМИ**

**Короткі відомості про мову С++:**

C++ — мова програмування загального призначення з підтримкою кількох парадигм програмування: об'єктно-орієнтованої, узагальненої, процедурної та інших.

Назва «Сі++» була вигадана Ріком Масситті і вперше була використана в грудні 1983 року. Раніше, на етапі розробки, нова мова називалася «Сі з класами». Ім'я, що вийшло у результаті, походить від оператора Сі «++» (збільшення значення змінної на одиницю) і поширеному способу присвоєння нових імен комп'ютерним програмам, що полягає в додаванні до імені символу «+» для позначення поліпшень.

Переваги мови С++:

* стандартизація C++ визначається міжнародним стандартом, а отже не контролюється якоюсь одною фірмою чи людиною;
* швидкість роботи програм на С++ практично не поступається програмам на С, хоча програмісти отримали в свої руки нові можливості і нові засоби;
* рішення розроблені на С++ можуть використовувати мінімальну необхідну кількість ресурсів таких як пам'ять, ЦПУ, енергія та інші;

Недоліки мови С++:

* наявність безліч можливостей, що порушують принципи типобезпеки приводить до того, що в С++ програми може легко закрастися важковловима помилка;
* погана підтримка модульності. Підключення інтерфейсу зовнішнього модуля через препроцесорну вставку заголовного файлу (#include) серйозно уповільнює компіляцію, при підключенні великої кількості модулів;
* препроцесор С++ (успадкований від C) дуже примітивний. Це приводить з одного боку до того, що з його допомогою не можна (або важко) здійснювати деякі завдання метапрограмування;

**Основні визначення, що були використані в програмі:**

* функція
* оператор
* файл
* клас
* макроозначення
* манупілятор

*Функція* – це окремі незалежні блоки коду, які виконують ряд зумовлених команд.

*Оператори* дозволяють використовувати об'єкти типу класу з операторами, визначеними у класі та маніпулювати ними так само інтуїтивно, як об'єктами вбудованих типів

*Файл* – це сукупність даних, які розміщені на зовнішньому носії, зокрема на жорсткому диску.

*Клас* –це визначений користувачем тип або структура даних, оголошена ключовим словом class , яка містить дані (поля) і функції (методи) як свої члени, доступ до якої регулюється трьома специфікаторами доступу: private, public, protected.

*Макроозначення* – це вираз, який при компіляції файлу з кодом програми підставляється замість імені, що визначає дану макропідстановку. Макровизначення обробляються препроцесором.

*Маніпулятори* використовують для зміни параметрів форматування потоку введення та виведення.

**Стандартні функції, що використовувалися в програмі:**

1. Функції бібліотеки *graphics.h*:

* initwindow() – ініціалізація графічного вікна;
* getmaxx(), getmaxy() – отримання розміру графічного вікна;
* setbkcolor(), setcolor(), setlinestyle() – зміна кольору вікна, лінії та типу лінії;
* cleardevice() – очищення графічого вікна;
* line(), outtextxy() – вивід лінії та тексту.

1. Функції бібліотеки *cassert*:

* assert() – перевірка аргументів функції. Якщо аргументи мають значення false, то виводиться відповідне повідомлення

1. Функції бібліотеки c*math*:

* exp() – експонента ();
* sqrt() – квадратний корінь.

1. Функції бібліотеки iostream:

* cin, cout – потоки введення та виведення;
* precision – маніпулятор для задання точності чисел;
* scientific – маніпулятор для друкування чисел у експоненціальному вигляді.

1. Функції бібліотеки fstream:

* ifstream, ofstream – потоки введення та виведення інформації у файл.

**Можливі помилки:**

* SQRT\_ERROR – неможливо порахувати квадратний корінь;
* DIV\_ERROR – неможливо поділити на нуль;
* OPEN\_ERROR – немоливо відкрити вхідний файл;
* CREATE\_ERROR – неможливо створити файл;
* DATA\_ERROR – некоретні вхідні дані;

**Макровизначення:**

* IN\_FILE "files/input.txt" – вхідний файл;
* OUT\_FILE "files/output.txt" – вихідний файл;
* HEIGHT 720 – висота графічного вікна;
* WIDTH 1280 – ширина графічного вікна;
* STR\_LEN 99 – максимальна довжина рядка;
* p0A 3 – мінімальне значення параметра p;
* p0B 5 – максимальне значення параметра p;
* TA 400 – мінімальне значення параметра T;
* TB 600 – максимальне значення параметра T.

# **ОПИС СТРУКТУРИ ДАНИХ ТА ВИМОГИ ДО НИХ**

**Вхідні дані:**

**Необхідно отримати:**

* – коефіціент відношення теплоємності при постійному тиску () і постійному об’ємі ().
* – швидкість витікання ідеального газу із посудини під тиском.

**Використані змінні:**

Таблица 1. Використані змінні

|  |  |  |  |
| --- | --- | --- | --- |
| Ім’я параметра у формулі | Назва | Тип | Призначення |
|  | a | PerfectGas | об’єкт для обчислення |
| p | dp | double | крок табулювання |
| p | p0 | double | початкове значення p |
| p | p1 | double | кінцеве значення p |
|  | ifs | ifstream | вхідний потік файлу |
|  | ofs | ofstream | вихідний потік файлу |

**Опис класу CompObj:**

Таблица 2. Опис класу CompObj

|  |  |  |  |
| --- | --- | --- | --- |
| Ім’я параметра у формулі | Назва | Тип | Призначення |
|  | calcPar | double | обчислювальний параметр |
|  | set() | virtual void | віртуальна функція для введення значень |
|  | get() | virtual void | віртуальна функція для виведення значень |
|  | calc() | virtual void | віртуальна функція для обрахування обчислювального параметру |
|  | getResult() | virtual void | віртуальна фукнція для виведення обчислювального параметру |

**Опис класу PerfectGas:**

Таблица 3. Опис класу PerfectGas

|  |  |  |  |
| --- | --- | --- | --- |
| Ім’я параметра у формулі | Назва | Тип | Призначення |
| R | R | double | універсальна газова константа |
| T | T | double | температура в посудині |
| Kg | Kg | double | коефіціент відношення |
| M | M | double | молекулярна маса газу |
| p | p | double | зовнішній тиск |
| p0 | p0 | double | тиск газу в посудині |
| K | K | double | відношення теплоємності при постійному тиску |
|  | checkParams() | void | фукнція для перевірки паметрів |
|  | PerfectGas() | - | пустий констуктор |
|  | PerfectGas(const PerfectGas& x) | - | конструктор копіювання |
|  | set() | void | функція введення параметрів |
|  | get() | void | фукнція виведення параметрів |
|  | calc() | void | фукнція для обрахунку обчислювального параметру |
|  | calcKg() | void | функція для обрахунку параметра Kg |
|  | getResult() | void | фукнція для виведення обчислювального параметру |
|  | operator = | PerfectGas& | оператор присвоювання |
|  | operator == | bool | оператор == |
|  | operator != | bool | оператор != |
|  | operator < | bool | оператор < |
|  | operator <= | bool | Оператор <= |
|  | operator > | bool | Оператор > |
|  | operator >= | bool | Оператор >= |
|  | operator >> | ifstream& | оператор для введення даних із файлу |
|  | operator << | ofstream& | оператор для виведення даних у файл |
|  | operator >> | istream& | оператор для введення даних із консолі |
|  | operator << | ostream& | оператор для виведення даних у консоль |
|  | tabulate() | void | функція для табулювання |

# **АЛГОРИТМ РОБОТИ ПРОГРАМИ**

**Блок-схема функції main():**

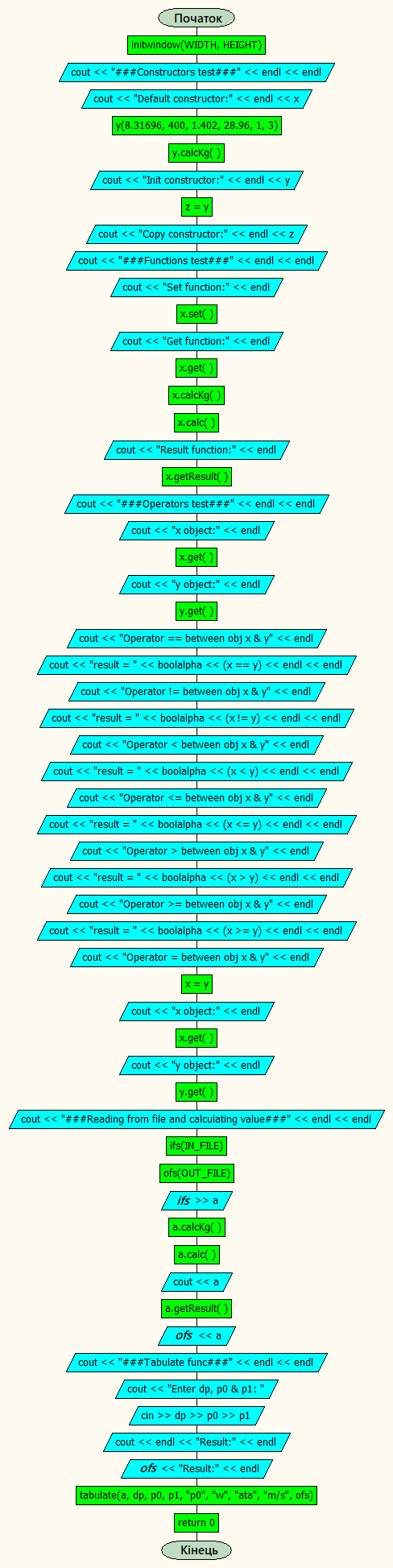


Рисунок 1 Блок-схема функції main()

**Блок-схема функції getMin():**

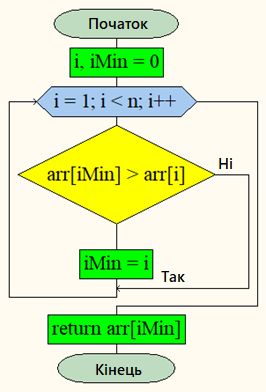
****

Рисунок 2 Блок-схема функції getMin()

**Блок-схема функції getMax():**

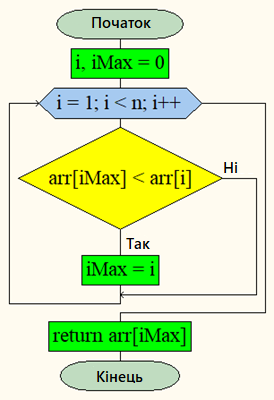
****

Рисунок 3 Блок-схема функції getMax()

**Блок-схема функції convert():**

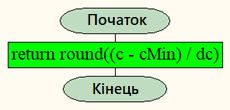
****

Рисунок 4 Блок-схема функції convert()

**Блок-схема функції draw():**

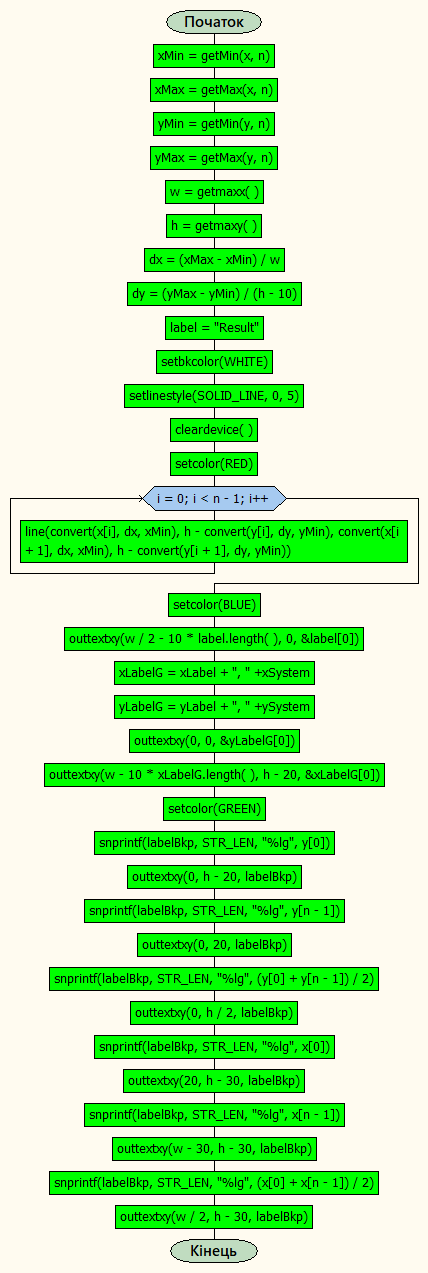


Рисунок 5 Блок-схема функції draw()

**Блок-схема функції tabulate():**

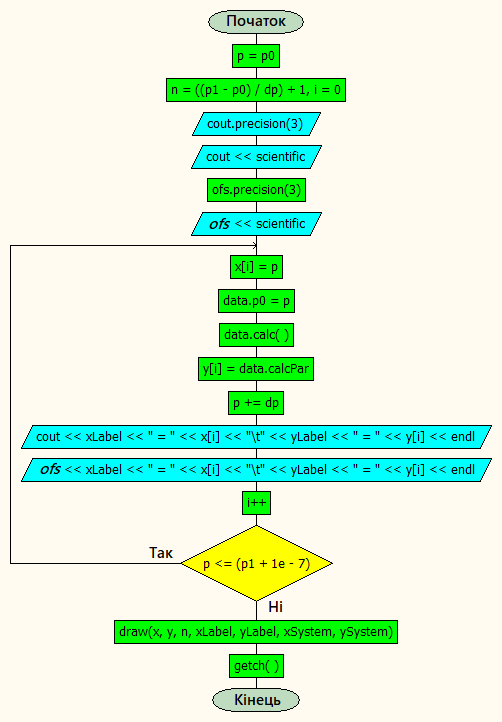
****

Рисунок 6 Блок-схема функції tabulate()

**Блок-схема функції checkParams() класу PerfectGas:**

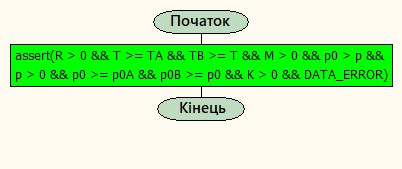


Рисунок 7 Блок-схема функції checkParams ()

**Блок-схема функції set() класу PerfectGas:**

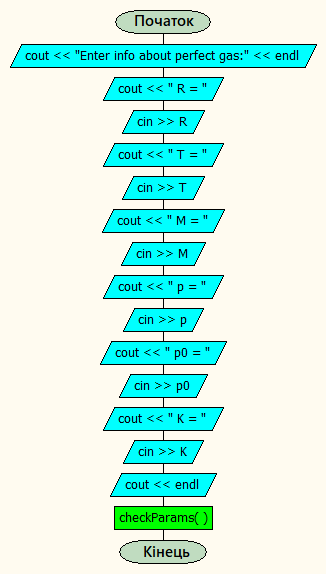


Рисунок 8 Блок-схема функції set ()

**Блок-схема функції get() класу PerfectGas:**

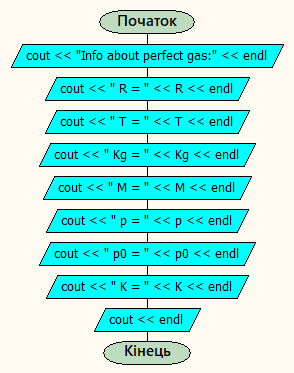


Рисунок 9 Блок-схема функції get ()

**Блок-схема функції calc() класу PerfectGas:**

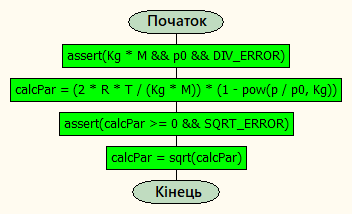


Рисунок 10 Блок-схема функції calc ()

**Блок-схема функції calcKg() класу PerfectGas:**

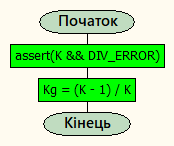


Рисунок 11 Блок-схема функції calcKg ()

**Блок-схема функції getResult() класу PerfectGas:**

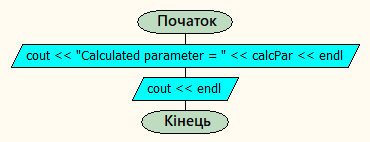


Рисунок 12 Блок-схема функції getResult ()

**Блок-схема** **оператора = класу PerfectGas:**

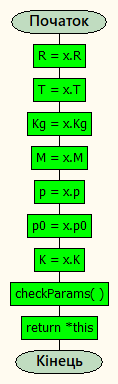


Рисунок 13 Блок-схема оператора =

**Блок-схема оператора == класу PerfectGas:**

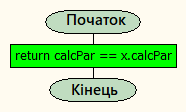


Рисунок 14 Блок-схема оператора ==

**Блок-схема оператора != класу PerfectGas:**

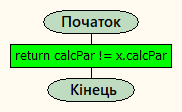


Рисунок 15 Блок-схема оператора !=

**Блок-схема оператора < класу PerfectGas:**

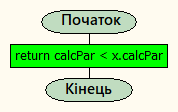


Рисунок 16 Блок-схема оператора <

**Блок-схема оператора <= класу PerfectGas:**

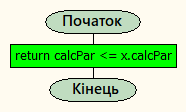


Рисунок 18 Блок-схема оператора <=

**Блок-схема оператора > класу PerfectGas:**

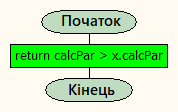


Рисунок 20 Блок-схема оператора >

**Блок-схема оператора >= класу PerfectGas:**

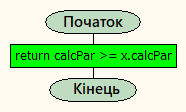


Рисунок 21 Блок-схема оператора >=

**Блок-схема оператора ifstream >> класу PerfectGas:**

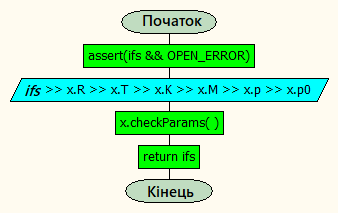


Рисунок 22 Блок-схема оператора ifstream >>

**Блок-схема оператора ofstream << класу PerfectGas:**

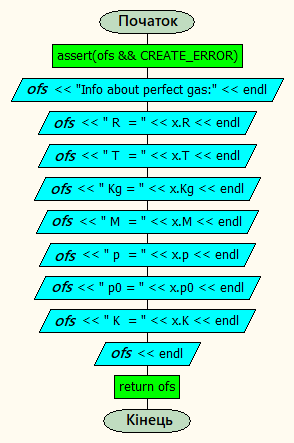


Рисунок 23 Блок-схема оператора ofstream <<

**Блок-схема оператора istream >> класу PerfectGas:**

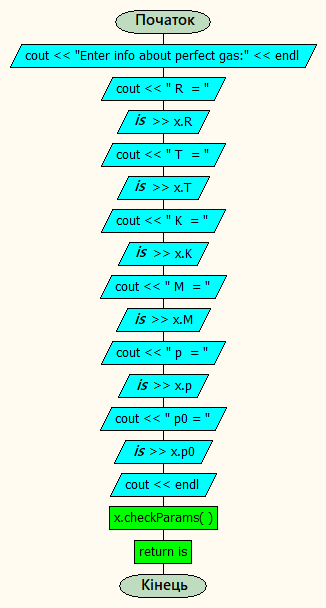


Рисунок 24 Блок-схема оператора istream >>

**Блок-схема оператора ostream << класу PerfectGas:**

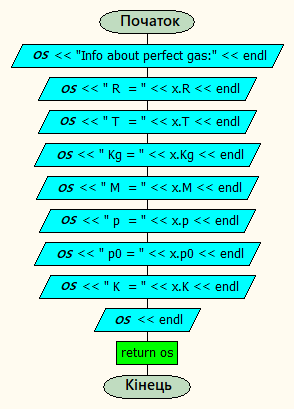


Рисунок 25 Блок-схема оператора ostream <<

# **ОПИС ФУНКЦІЙ КОРИСТУВАЧА**

1. Функція getMin():

Таблица 4. Функція getMin()

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Призначення функції | Назва параметра | Тип параметра | Тип даних | Призначення |
| Знаходить найменше значення  в масиві | arr | вхідний | double[] | масив |
| n | вхідний | int | розмір масиву |
| iMin | вихідний | int | індекс мін.  елемента |

1. Функція getMax():

Таблица 5. Функція getMax()

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Призначення функції | Назва параметра | Тип параметра | Тип даних | Призначення |
| Знаходить найбільше значення  в масиві | arr | вхідний | double[] | масив |
| n | вхідний | int | розмір масиву |
| iMax | вихідний | int | індекс макс.  елемента |

1. Функція convert():

Таблица 6. Функція convert()

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Призначення функції | Назва параметра | Тип параметра | Тип даних | Призначення |
| Перетворює обраховене значення функції в координату на екрані | c | вхідний | double | значення ф. |
| dc | вхідний | double | зміна функції |
| cMin | вхідний | double | мін. значення |
| round((c - cMin) / dc) | вихідний | double | обрахована координата |

1. Функція draw():

Таблица 7. Функція draw()

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Призначення функції | Назва параметра | Тип параметра | Тип даних | Призначення |
| Будує  графік  функції | x | вхідний | double\* | значення x |
| y | вхідний | double\* | значення y |
| n | вхідний | int | кількість точок |
| xLabel | вхідний | string | назва осі Ox |
| yLabel | вхідний | string | назва осі Oy |
| xSystem | вхідний | string | тип осі Ox |
| ySystem | вхідний | string | тип осі Oy |

1. Функція tabulate():

Таблица 8. Функція tabulate()

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Призначення функції | Назва параметра | Тип параметра | Тип даних | Призначення |
| Табулює функцію на проміжку з кроком | data | вхідний | PerfectGas | вхідні дані |
| dp | вхідний | double | крок табуляції |
| p0 | вхідний | double | початкова точка |
| p1 | вхідний | double | кінцева точка |
| xLabel | вхідний | string | назва осі Ox |
| yLabel | вхідний | string | назва осі Oy |
| xSystem | вхідний | string | тип осі Ox |
| ySystem | вхідний | string | тип осі Oy |
| ofs | вхідний | ofstream& | потік виведення |

1. Функція checkParams() класу PerfectGas:

Таблица 9. Функція checkParams()

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Призначення функції | Назва параметра | Тип параметра | Тип даних | Призначення |
| Перевіряє параметри класу  PerfectGas | - | - | - | - |

1. Функція set() класу PerfectGas:

Таблица 10. Функція set()

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Призначення функції | Назва параметра | Тип параметра | Тип даних | Призначення |
| Введення параметрів класу  PerfectGas | - | - | - | - |

1. Функція get() класу PerfectGas:

Таблица 11. Функція get()

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Призначення функції | Назва параметра | Тип параметра | Тип даних | Призначення |
| Виведення параметрів класу | - | - | - | - |

1. Функція calc() класу PerfectGas:

Таблица 12. Функція calc()

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Призначення функції | Назва параметра | Тип параметра | Тип даних | Призначення |
| Обрахування обчислювального параметру | - | - | - | - |

1. Функція calcKg() класу PerfectGas:

Таблица 13. Функція calcKg()

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Призначення функції | Назва параметра | Тип параметра | Тип даних | Призначення |
| Обрахування параметру Kg | - | - | - | - |

1. Функція getResult() класу PerfectGas:

Таблица 14. Функція getResult()

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Призначення функції | Назва параметра | Тип параметра | Тип даних | Призначення |
| Друкування обчислювального параметру | - | - | - | - |

# **ОПИС ФАЙЛІВ ТА ЇХ ПРИЗНАЧЕННЯ**

**Структура програми:**

Таблица 15. Структура програми

|  |  |  |
| --- | --- | --- |
| Назва | Призначення | Структура |
| input.txt | містяться вхідні дані | містить набори вхідних даних |
| output.txt | містяться вихідні дані | містить протабульовані функції |
| main.cpp | файл з програмним кодом | містить розроблену програму |
| main.exe | скомпільована програма |  |

# **СПИСОК ВИКОРИСТАНИХ БІБЛІОТЕК**

**Використані бібліотеки:**

* Бібліотека graphics.h – ця бібліотека забезпечує доступ до простої графіки, що дозволяє малювати лінії, прямокутники, овали, дуги, багатокутники, зображення та рядки у графічному вікні;
* Бібліотека cmath – стандартна бібліотеки мови програмування С++, яка надає прототипи функцій, розроблены для виконання простих математичних операцій;
* Бібліотека iostream — бібліотека і відповідний заголовний файл, які використовується для організації введення-виведення в мові програмування C++;
* Бібліотека fstream — заголовковий файл для роботи з файлами, у якому підключено такі заголовкові файли як ifstream — бібліотека для файлового введеня, і ofstream — бібліотека для файлового виведення;
* Бібліотека cassert — містить існтрументи для відслідковування помилок.

# **ІНСТРУКЦІЯ ДЛЯ РОБОТИ З ПРОГРАМОЮ**

**Ресурси необхідні для роботи програми:**

* Компілятор C++ для компіляції коду. Наприклад, Dev C++ або Microsoft Visual Studio;
* Встановити бібліотеку graphics.h із мережі інтернет.

**Необхідно зробити для запуску програми:**

* Занести вхідні дані у файл input.txt;
* Скомпілювати програмний код за допомогою C++ компілятора;
* Запустити на виконання скомпільований додаток.

**Варіанти можливих повідомлень при роботі з програмою:**

* Некоректні вхідні параметри:

**Изображение выглядит как текст, экран, снимок экрана, закрыть

Автоматически созданное описание**

Рисунок 26. Некоректні вхідні параметри

* Ділення на нуль:

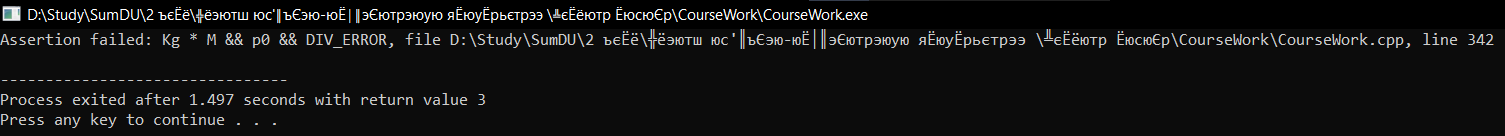
****

Рисунок 27. Ділення на нуль

* Неможливо порахувати квадрантий корінь

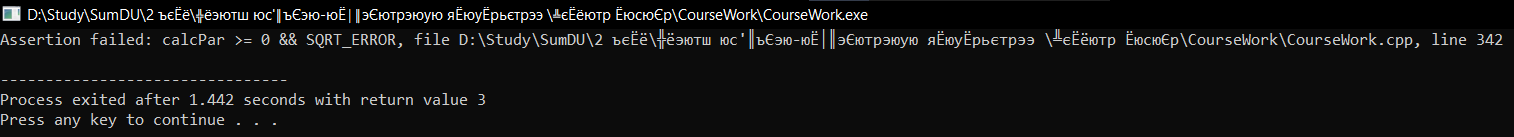
****

Рисунок 28. Неможливо порахувати квадратний корінь

* Неможливо відкрити вхідний файл

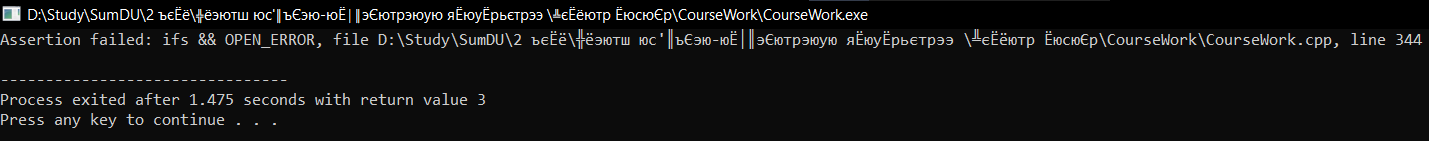
****

Рисунок 29. Неможливо відкрити вхідний файл

* Неможливо створити вихідний файл

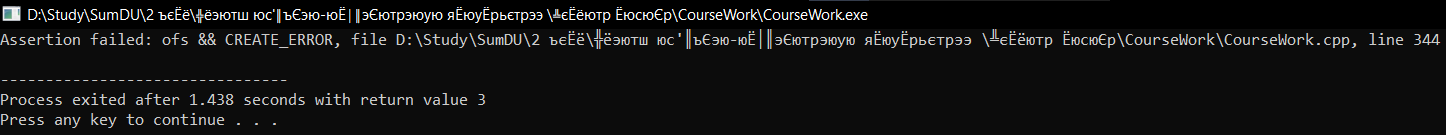


Рисунок 30. Неможливо створити вихідний файл

# **ПРИКЛАД ТЕСТУВАННЯ ТА РЕЗУЛЬТАТИ РОБОТИ ПРОГРАМИ**

**Вхідні дані:**

**Вміст файлу input.txt:**

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 31. Вхідний файл

Для запуску програми потрібно запустити на виконання main.exe

**Переглянемо результат роботи програми при відсутності помилок:**

**Вигляд командного рядка:**

Изображение выглядит как текст

Автоматически созданное описание Изображение выглядит как текст

Автоматически созданное описание Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание Изображение выглядит как текст, электроника, клавиатура

Автоматически созданное описание

Рисунок 32. Командний рядок

**Вміст файлу output.txt після виконання програми:**

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 33. Вихідний файл

# **ГРАФІК**

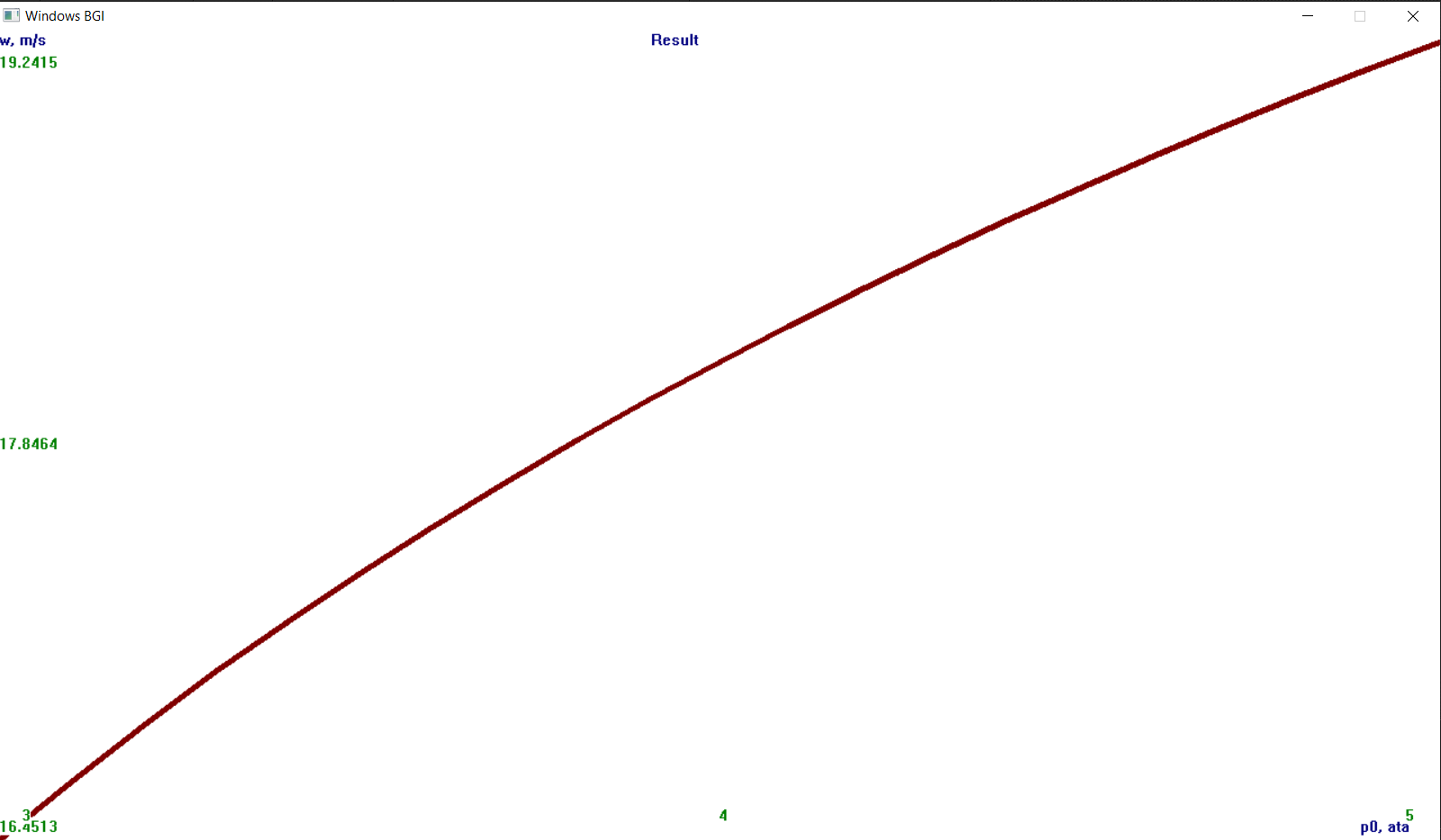


Рисунок 34. Графік

# **КОД ПРОГРАМИ**

1. #include <iostream>
2. #include <cmath>
3. #include <fstream>
4. #include <cassert>
5. #include <graphics.h>
7. */\*вхідний та вихідний файли\*/*
8. #define IN\_FILE         "files/input.txt"
9. #define OUT\_FILE        "files/output.txt"
11. */\*висота та ширина графічного вікна\*/*
12. #define HEIGHT          720
13. #define WIDTH           1280
15. */\*довжина рядка\*/*
16. #define STR\_LEN         99
18. */\*можливі помилки\*/*
19. #define DIV\_ERROR       "Cannot be divided by zero!"
20. #define SQRT\_ERROR      "Cannot take the root of a negative number!"
21. #define OPEN\_ERROR      "The file cannot be opened!"
22. #define CREATE\_ERROR    "The file cannot be created!"
23. #define DATA\_ERROR      "Invalid data!"
25. */\*межі параметрів\*/*
26. #define p0A 3
27. #define p0B 5
28. #define TA  400
29. #define TB  600
31. using namespace std;
33. */\*клас об’єкт обчислення\*/*
34. class CompObj {
35. protected:
36. */\*обчислюваний параметр\*/*
37. double calcPar;
39. public:
40. */\*віртуальна функція введення даних\*/*
41. virtual void set() = 0;
43. */\*віртуальна функція виведення даних\*/*
44. virtual void get() = 0;
46. */\*віртуальна функція обчислення\*/*
47. virtual void calc() = 0;
49. */\*віртуальна функція друкування результату обчислення\*/*
50. virtual void getResult() = 0;
51. };
53. */\*клас ідеальний газ похідний від абстрактного класу об’єкт обчислення\*/*
54. class PerfectGas : virtual public CompObj {
55. private:
56. */\*параметри для обчислення\*/*
57. double R, T, Kg, M, p, p0, K;
59. */\*функція для перевірки параметрів\*/*
60. void checkParams() {
61. assert(R > 0 && T >= TA && TB >= T && M > 0
62. && p0 > p && p > 0 && p0 >= p0A
63. && p0B >= p0 && K > 0 && DATA\_ERROR);
64. }
66. public:
67. */\*конструктор за замовчуванням\*/*
68. PerfectGas() { R = T = Kg = M = p = p0 = K = 0; }
70. */\*констуктор з ініціалізацією списком\*/*
71. PerfectGas(double \_R, double \_T, double \_K, double \_M, double \_p, double \_p0) : R(\_R),
72. T(\_T), K(\_K), M(\_M), p(\_p), p0(\_p0) {
73. checkParams();
74. }
76. */\*констуктор копіювання\*/*
77. PerfectGas(const PerfectGas& x) {
78. R = x.R;
79. T = x.T;
80. Kg = x.Kg;
81. M = x.M;
82. p = x.p;
83. p0 = x.p0;
84. K = x.K;
85. checkParams();
86. }
88. */\*перевантажена фукнція введення даних\*/*
89. void set() {
90. cout << "Enter info about perfect gas:" << endl;
91. cout << " R = ";    cin >> R;
92. cout << " T = ";    cin >> T;
93. cout << " M = ";    cin >> M;
94. cout << " p = ";    cin >> p;
95. cout << " p0 = ";   cin >> p0;
96. cout << " K = ";    cin >> K;
97. cout << endl;
98. checkParams();
99. }
101. */\*перевантажена функція виведення даних\*/*
102. void get() {
103. cout << "Info about perfect gas:" << endl;
104. cout << " R = " << R << endl;
105. cout << " T = " << T << endl;
106. cout << " Kg = " << Kg << endl;
107. cout << " M = " << M << endl;
108. cout << " p = " << p << endl;
109. cout << " p0 = " << p0 << endl;
110. cout << " K = " << K << endl;
111. cout << endl;
112. }
114. */\*перевантажена фукнція для обчислення\*/*
115. void calc() {
116. assert(Kg \* M && p0 && DIV\_ERROR);
118. calcPar = (2 \* R \* T / (Kg \* M)) \* (1 - pow(p / p0, Kg));
120. assert(calcPar >= 0 && SQRT\_ERROR);
122. calcPar = sqrt(calcPar);
123. }
125. */\*функція для обчислення параметра Kg\*/*
126. void calcKg() {
127. assert(K && DIV\_ERROR);
129. Kg = (K - 1) / K;
130. }
132. */\*функція для виведення обчислювального параметру\*/*
133. void getResult() {
134. cout << "Calculated parameter = " << calcPar << endl;
135. cout << endl;
136. }
138. */\*оператор присвоювання\*/*
139. PerfectGas& operator = (const PerfectGas& x) {
140. R = x.R;
141. T = x.T;
142. Kg = x.Kg;
143. M = x.M;
144. p = x.p;
145. p0 = x.p0;
146. K = x.K;
147. checkParams();
148. return \*this;
149. }
151. */\*оператор ==\*/*
152. bool operator == (const PerfectGas& x) { return calcPar == x.calcPar; }
154. */\*оператор !=\*/*
155. bool operator != (const PerfectGas& x) { return calcPar != x.calcPar; }
157. */\*оператор <\*/*
158. bool operator < (const PerfectGas& x) { return calcPar < x.calcPar; }
160. */\*оператор <=\*/*
161. bool operator <= (const PerfectGas& x) { return calcPar <= x.calcPar; }
163. */\*оператор >\*/*
164. bool operator > (const PerfectGas& x) { return calcPar > x.calcPar; }
166. */\*оператор >=\*/*
167. bool operator >= (const PerfectGas& x) { return calcPar >= x.calcPar; }
169. */\*оператор для введення даних із файлу\*/*
170. friend ifstream& operator >> (ifstream& ifs, PerfectGas& x);
172. */\*оператор для виведення даних у файл\*/*
173. friend ofstream& operator << (ofstream& ofs, const PerfectGas& x);
175. */\*оператор для введення даних із консолі\*/*
176. friend istream& operator >> (istream& is, PerfectGas& x);
178. */\*оператор для виведення даних у консоль\*/*
179. friend ostream& operator << (ostream& os, const PerfectGas& x);
181. */\*дружня функція для табулювання\*/*
182. friend void tabulate(PerfectGas data, double dp, double p0, double p1,
183. string xLabel, string yLabel, string xSystem, string ySystem, ofstream& ofs);
184. };
186. */\*оператор для введення даних із файлу\*/*
187. ifstream& operator >> (ifstream& ifs, PerfectGas& x) {
188. assert(ifs && OPEN\_ERROR);
190. ifs >> x.R >> x.T >> x.K >> x.M >> x.p >> x.p0;
191. x.checkParams();
192. return ifs;
193. }
195. */\*оператор для виведення даних у файл\*/*
196. ofstream& operator << (ofstream& ofs, const PerfectGas& x) {
197. assert(ofs && CREATE\_ERROR);
199. ofs << "Info about perfect gas:" << endl;
200. ofs << " R  = " << x.R << endl;
201. ofs << " T  = " << x.T << endl;
202. ofs << " Kg = " << x.Kg << endl;
203. ofs << " M  = " << x.M << endl;
204. ofs << " p  = " << x.p << endl;
205. ofs << " p0 = " << x.p0 << endl;
206. ofs << " K  = " << x.K << endl;
207. ofs << endl;
209. return ofs;
210. }
212. */\*оператор для введення даних із консолі\*/*
213. istream& operator >> (istream& is, PerfectGas& x) {
214. cout << "Enter info about perfect gas:" << endl;
215. cout << " R  = ";   is >> x.R;
216. cout << " T  = ";   is >> x.T;
217. cout << " K  = ";   is >> x.K;
218. cout << " M  = ";   is >> x.M;
219. cout << " p  = ";   is >> x.p;
220. cout << " p0 = ";   is >> x.p0;
221. cout << endl;
222. x.checkParams();
223. return is;
224. }
226. */\*оператор для виведення даних у консоль\*/*
227. ostream& operator << (ostream& os, const PerfectGas& x) {
228. os << "Info about perfect gas:" << endl;
229. os << " R  = " << x.R << endl;
230. os << " T  = " << x.T << endl;
231. os << " Kg = " << x.Kg << endl;
232. os << " M  = " << x.M << endl;
233. os << " p  = " << x.p << endl;
234. os << " p0 = " << x.p0 << endl;
235. os << " K  = " << x.K << endl;
236. os << endl;
238. return os;
239. }
241. */\*допоміжна функція для побудови графіка (знаходження мінімуму у масиві)\*/*
242. double getMin(double arr[], int n) {
243. int i, iMin = 0;
245. for (i = 1; i < n; i++) {
246. if (arr[iMin] > arr[i]) {
247. iMin = i;
248. }
249. }
251. return arr[iMin];
252. }
254. */\*допоміжна функція для побудови графіка (знаходження максимуму у масиві)\*/*
255. double getMax(double arr[], int n) {
256. int i, iMax = 0;
258. for (i = 1; i < n; i++) {
259. if (arr[iMax] < arr[i]) {
260. iMax = i;
261. }
262. }
264. return arr[iMax];
265. }
267. */\*допоміжна функція для побудови графіка\*/*
268. int convert(double c, double dc, double cMin) {return round((c - cMin) / dc);}
270. */\*функція для побудови графіка\*/*
271. void draw(double\* x, double\* y, int n, string xLabel, string yLabel, string xSystem, string ySystem) {
272. string xLabelG;
273. string yLabelG;
275. double xMin = getMin(x, n);
276. double xMax = getMax(x, n);
277. double yMin = getMin(y, n);
278. double yMax = getMax(y, n);
280. int w = getmaxx();
281. int h = getmaxy();
283. double dx = (xMax - xMin) / w;
284. double dy = (yMax - yMin) / (h - 10);
286. string label = "Result";
287. char labelBkp[STR\_LEN];
289. int i;
291. setbkcolor(WHITE);
292. setlinestyle(SOLID\_LINE, 0, 5);
293. cleardevice();
294. setcolor(RED);
296. for (i = 0; i < n - 1; i++) {
297. line(convert(x[i], dx, xMin),
298. h - convert(y[i], dy, yMin),
299. convert(x[i + 1], dx, xMin),
300. h - convert(y[i + 1], dy, yMin));
301. }
303. setcolor(BLUE);
305. outtextxy(w / 2 - 10 \* label.length(), 0, &label[0]);
307. xLabelG = xLabel + ", " + xSystem;
308. yLabelG = yLabel + ", " + ySystem;
310. outtextxy(0, 0, &yLabelG[0]);
311. outtextxy(w - 10 \* xLabelG.length(), h - 20, &xLabelG[0]);
313. setcolor(GREEN);
315. snprintf(labelBkp, STR\_LEN, "%lg", y[0]);
317. outtextxy(0, h - 20, labelBkp);
319. snprintf(labelBkp, STR\_LEN, "%lg", y[n - 1]);
321. outtextxy(0, 20, labelBkp);
323. snprintf(labelBkp, STR\_LEN, "%lg", (y[0] + y[n - 1]) / 2);
325. outtextxy(0, h / 2, labelBkp);
327. snprintf(labelBkp, STR\_LEN, "%lg", x[0]);
329. outtextxy(20, h - 30, labelBkp);
331. snprintf(labelBkp, STR\_LEN, "%lg", x[n - 1]);
333. outtextxy(w - 30, h - 30, labelBkp);
335. snprintf(labelBkp, STR\_LEN, "%lg", (x[0] + x[n - 1]) / 2);
337. outtextxy(w / 2, h - 30, labelBkp);
338. }
340. */\*функція для табулювання\*/*
341. void tabulate(PerfectGas data, double dp, double p0, double p1,
342. string xLabel, string yLabel, string xSystem, string ySystem, ofstream& ofs) {
343. double p = p0;
344. int n = ((p1 - p0) / dp) + 1, i = 0;
345. double x[n], y[n];
347. cout.precision(3);
348. cout << scientific;
350. ofs.precision(3);
351. ofs << scientific;
353. do {
354. x[i] = p;
356. data.p0 = p;
357. data.calc();
359. y[i] = data.calcPar;
361. p += dp;
363. cout << xLabel << " = " << x[i] << "**\t**" << yLabel << " = " << y[i] << endl;
364. ofs << xLabel << " = " << x[i] << "**\t**" << yLabel << " = " << y[i] << endl;
366. i++;
367. } while (p <= (p1 + 1e-7));
369. draw(x, y, n, xLabel, yLabel, xSystem, ySystem);
370. getch();
371. }
373. int main() {
374. initwindow(WIDTH, HEIGHT);
376. cout << "###Constructors test###" << endl << endl;
378. PerfectGas x;
380. cout << "Default constructor:" << endl << x;
382. PerfectGas y(8.31696, 400, 1.402, 28.96, 1, 3);
384. y.calcKg();
386. cout << "Init constructor:" << endl << y;
388. PerfectGas z = y;
390. cout << "Copy constructor:" << endl << z;
392. cout << "###Functions test###" << endl << endl;
394. cout << "Set function:" << endl;
395. x.set();
397. cout << "Get function:" << endl;
398. x.get();
400. x.calcKg(); x.calc();
402. cout << "Result function:" << endl;
403. x.getResult();
405. cout << "###Operators test###" << endl << endl;
407. cout << "x object:" << endl;
408. x.get();
410. cout << "y object:" << endl;
411. y.get();
413. cout << "Operator == between obj x & y" << endl;
414. cout << "result = " << boolalpha << (x == y) << endl << endl;
416. cout << "Operator != between obj x & y" << endl;
417. cout << "result = " << boolalpha << (x != y) << endl << endl;
419. cout << "Operator < between obj x & y" << endl;
420. cout << "result = " << boolalpha << (x < y) << endl << endl;
422. cout << "Operator <= between obj x & y" << endl;
423. cout << "result = " << boolalpha << (x <= y) << endl << endl;
425. cout << "Operator > between obj x & y" << endl;
426. cout << "result = " << boolalpha << (x > y) << endl << endl;
428. cout << "Operator >= between obj x & y" << endl;
429. cout << "result = " << boolalpha << (x >= y) << endl << endl;
431. cout << "Operator = between obj x & y" << endl;
432. x = y;
434. cout << "x object:" << endl;
435. x.get();
437. cout << "y object:" << endl;
438. y.get();
440. cout << "###Reading from file and calculating value###" << endl << endl;
442. PerfectGas a; ifstream ifs(IN\_FILE); ofstream ofs(OUT\_FILE);
444. ifs >> a;
446. a.calcKg(); a.calc();
448. cout << a;
450. a.getResult();
452. ofs << a;
454. cout << "###Tabulate func###" << endl << endl;
456. double dp, p0, p1;
458. cout << "Enter dp, p0 & p1: ";
459. cin >> dp >> p0 >> p1;
461. cout << endl << "Result:" << endl;
462. ofs << "Result:" << endl;
464. tabulate(a, dp, p0, p1, "p0", "w", "ata", "m/s", ofs);
466. return 0;
467. }

# **ВИСНОВОК**

Головною метою написання курсової роботи було закріплення та поглиблення теоретичних знань, оволодіння практичними навичками щодо розроблення алгоритмів, розв’язування задач за математичним описом для вирішення конкретних завдань, а також їх реалізація мовою програмування С++, закріплення навичок самостійної роботи шляхом пошуку, аналізу та активного засвоювання інформації із різних джерел.

У результаті виконання курсової роботи було розроблено прогаму із використанням алгоритмічної мови програмування C++. Ця програма будує графік зміни швидкісті витікання ідеального газу із посудини під тиском. Під час розробки було використано багато допоміжних бібліотек для роботи з графікою, з введенням та виведенням інформації в командний рядок та файл, з математичними функціями.

Розробивши програму, ми отримали графік зміни швидкісті витікання ідеального газу із посудини під тиском та протабульовану функцію. Результат роботи програми міститься у вихідному файлі output.txt.

# **СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ**

1. Справочник C/C++ - graphics.h. С++ – Программируем. URL: <http://mycpp.ru/cpp/scpp/cppd_graphics.h.htm> (дата звернення: 04.05.2021).
2. Методичні вказівки до курсової роботи з дисципліни «Програмування» / укладачі: В. В. Авраменко, В. О. Боровик, Н. В. Тиркусова. – Суми : Сумський державний університет, 2021. – 43 с.
3. Использование манипуляторов | Программирование на C и C++. *Программирование на C и C++ | Онлайн справочник программиста на C и C++*. URL: <http://www.c-cpp.ru/books/ispolzovanie-manipulyatorov> (дата звернення: 15.12.2021).
4. Учасники проектів Вікімедіа. C++ – Вікіпедія. *Вікіпедія*. URL: <https://uk.wikipedia.org/wiki/C++> (дата звернення: 15.12.2021).
5. C++ для начинающих. 15. Перегруженные операторы и определенные пользователем преобразования. *С++ – Программируем*. URL: <http://mycpp.ru/cpp/book/c15.html> (дата звернення: 15.12.2021).
6. *eLearning@SumyStateUniversity*. URL: <https://mix.sumdu.edu.ua/textbooks/21664/1160834/index.html> (дата звернення: 15.12.2021).