**Варіант 20**

Описати клас об’єктів «Пробіг молекули». Як відомо з фізики, молекула газу проходить певну відстань, перш ніж зіткнеться з іншою молекулою. Середню довжину вільного перебігу обчислюють за формулою

де N – кількість молекул;

δ – діаметр молекули.

Кількість молекул у заданому об’ємі обчислюють за формулою

де *p* – тиск газу;

– температура газу.

Під час описання класу передбачити такі елементи даних: середню довжину вільного пробігу, діаметр молекули, кількість молекул, тиск газу, температуру газу.

Також передбачити методи:

1. уведення даних із файлу;
2. розрахунок середньої довжини вільного пробігу;
3. виведення в інший файл вхідних даних і довжини пробігу.

Передбачити перевантажені операції присвоєння, порівняння (==,!=,<,<=,>,>=), а також конструктори: пустий, з ініціалізацією та копіювання.

Навести приклад програми, в якій застосовують усі конструктори, методи та перевантажені операції.

У контрольному прикладі можна використати такі значення параметрів:

Дані зчитувати з файлу. Результат розрахунку вивести в інший файл разом із вхідними даними.

Обрати один, будь-який, **параметр**, що має зазначений діапазон значень, та протабулювати «Обчислювальний параметр» за зміни параметра від **початкового**значення до **кінцевого** з **кроком Δ.** Результати обчислень вивести у файл. Побудувати графік одержаної залежності.

|  |
| --- |
| #include <iostream>  #include<cmath>  #include <iomanip>  #include<Windows.h>  #include<fstream>  #define pi 3,14  using namespace std;  //абстрактний клас  class Object {  protected:  //oбчислювальний параметр  double l;  public:  //чисті віртуальні функції  virtual void enter\_file(const char\* S) = 0;  virtual void calc\_l() = 0;  virtual void output\_file(const char\* S) = 0;  virtual const double output\_l() = 0;  virtual ~Object() {}  };  //похідний клас  class Mileage\_molecule : public Object {  double N = 0;  double d = 0;  double p = 0;  double Tg = 0;  public:  //конструктори  Mileage\_molecule() {};  Mileage\_molecule(double d, double p, double Tg);  Mileage\_molecule(const Mileage\_molecule& Mm);  void enter\_file(const char\* S);  void calc\_l();  const double output\_l();  double calc\_N();  void output\_file(const char\* S);  //перевантажені операції  void operator=(Mileage\_molecule const& Mm);  friend bool operator==(Mileage\_molecule const& A, Mileage\_molecule const& B);  friend bool operator!=(Mileage\_molecule const& A, Mileage\_molecule const& B);  friend bool operator<(Mileage\_molecule const& A, Mileage\_molecule const& B);  friend bool operator<=(Mileage\_molecule const& A, Mileage\_molecule const& B);  friend bool operator>(Mileage\_molecule const& A, Mileage\_molecule const& B);  friend bool operator>=(Mileage\_molecule const& A, Mileage\_molecule const& B);  ~Mileage\_molecule() {}  };  int main()  {  //для використання української мови  SetConsoleCP(1251);  SetConsoleOutputCP(1251);  cout << "No I" << endl;  cout << "Покажчик на батьківський клас. Пустий конструктор. Введення з файлу" << endl;  Object\* Obj\_1 = new Mileage\_molecule();  Obj\_1->enter\_file("Obj\_1.txt");  Obj\_1->calc\_l();  Obj\_1->output\_file("Obj\_1\_res.txt");  cout << endl;  cout << "No II" << endl;  cout << "Покажчик на батьківський клас. Конструктор з ініціалізацією:" << endl << "N = 5 d = 3, 656 \* 10 ^ -10 p = 1, 5 Tg = 500" << endl;  Object\* Obj\_2 = new Mileage\_molecule(3.656 \* pow(10, -10), 1.5, 500);  Obj\_2->calc\_l();  cout << "l = " << Obj\_2->output\_l() << endl;  cout << endl;  ofstream ofs("comparison\_res.txt");  cout << "No III" << endl;  cout << "Пустий конструктор. Введення з файлу " << endl;  Mileage\_molecule Obj\_3;  Obj\_3.enter\_file("Obj\_3.txt");  Obj\_3.calc\_l();  Obj\_3.output\_file("Obj\_3\_res.txt");  ofs << "l = " << Obj\_3.output\_l() << endl;  cout << endl;  cout << "No IV" << endl;  cout << "Конструктор з ініціалізацією" << endl << "N = 1 d = 3,64 \* 10^-10 p = 1,2 Tg = 300 " << endl;  Mileage\_molecule Obj\_4(3.64 \* pow(10, -10), 1.2, 300);  Obj\_4.calc\_l();  cout << "l = " << Obj\_4.output\_l() << endl;  ofs << "l = " << Obj\_4.output\_l() << endl;  cout << endl;  cout << "No V" << endl;  cout << "Конструктор копії" << endl << "No V = No IV" << endl;  Mileage\_molecule Obj\_5(Obj\_4);  Obj\_5.calc\_l();  cout << "l = " << Obj\_5.output\_l() << endl;  ofs << "l = " << Obj\_5.output\_l() << endl;  cout << endl << endl;  //Порівняння  ofs << "Порівняння" << endl;  ofs << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;  cout << "Порівняння" << endl;  cout << "Obj\_3 == Obj\_4 ?" << endl;  if (Obj\_3 == Obj\_4) {  cout << "ТАК" << endl;  ofs << "Obj\_3 == Obj\_4" << endl;  }  else {  cout << "НІ" << endl;  ofs << "Obj\_3 != Obj\_4" << endl;  }  cout << "Obj\_4 == Obj\_5 ?" << endl;  if (Obj\_4 == Obj\_5) {  cout << "ТАК" << endl;  ofs << "Obj\_4 == Obj\_5" << endl;  }  else {  cout << "НІ" << endl;  ofs << "Obj\_4 != Obj\_5" << endl;  }  cout << "Obj\_3 == Obj\_5 ?" << endl;  if (Obj\_3 == Obj\_5) {  cout << "ТАК" << endl;  ofs << "Obj\_3 == Obj\_5" << endl;  }  else {  cout << "НІ" << endl;  ofs << "Obj\_3 != Obj\_5" << endl;  }  ofs << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;  cout << "Obj\_3 != Obj\_4 ?" << endl;  if (Obj\_3 != Obj\_4) {  cout << "ТАК" << endl;  ofs << "Obj\_3 != Obj\_4" << endl;  }  else {  cout << "НІ" << endl;  ofs << "Obj\_3 == Obj\_4" << endl;  }  cout << "Obj\_4 != Obj\_5 ?" << endl;  if (Obj\_4 != Obj\_5) {  cout << "ТАК" << endl;  ofs << "Obj\_4 != Obj\_5" << endl;  }  else {  cout << "НІ" << endl;  ofs << "Obj\_4 == Obj\_5" << endl;  }  cout << "Obj\_3 != Obj\_5 ?" << endl;  if (Obj\_3 != Obj\_5) {  cout << "ТАК" << endl;  ofs << "Obj\_3 != Obj\_5" << endl;  }  else {  cout << "НІ" << endl;  ofs << "Obj\_3 == Obj\_5" << endl;  }  ofs << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;  cout << "Obj\_3 > Obj\_4 ?" << endl;  if (Obj\_3 > Obj\_4) {  cout << "ТАК" << endl;  ofs << "Obj\_3 > Obj\_4" << endl;  }  else {  cout << "НІ" << endl;  ofs << "Obj\_3 <= Obj\_4" << endl;  }  cout << "Obj\_4 >= Obj\_5 ?" << endl;  if (Obj\_4 >= Obj\_5) {  cout << "ТАК" << endl;  ofs << "Obj\_4 >= Obj\_5" << endl;  }  else {  cout << "НІ" << endl;  ofs << "Obj\_4 < Obj\_5" << endl;  }  cout << "Obj\_3 < Obj\_5 ?" << endl;  if (Obj\_3 < Obj\_5) {  cout << "ТАК" << endl;  ofs << "Obj\_3 < Obj\_5" << endl;  }  else {  cout << "НІ" << endl;  ofs << "Obj\_3 >= Obj\_5" << endl;  }  cout << "Obj\_3 <= Obj\_4 ?" << endl;  if (Obj\_3 <= Obj\_4) {  cout << "ТАК" << endl;  ofs << "Obj\_3 <= Obj\_4" << endl;  }  else {  cout << "НІ" << endl;  ofs << "Obj\_3 > Obj\_4" << endl;  }  cout << endl << endl;  //табуляція  ofstream ofs1("tabulate\_res.txt");  ofs1 << "Табуляція за параметром Tg" << endl;  ofs1 << "Tg \t l" << endl;  cout << "Табуляція за параметром Tg" << endl;  cout << "Tg \t l" << endl;  for (int delta = 300; delta <= 1300; delta += 100) {  Mileage\_molecule l\_tabulate(3.6 \* pow(10, -10), 1.25, delta);  l\_tabulate.calc\_l();  ofs1 << delta << "\t" << setprecision(5) << l\_tabulate.output\_l() << endl;  cout << delta << "\t" << setprecision(5) << l\_tabulate.output\_l() << endl;  }  cout << endl;  cout << "Результати файлi 'tabulate\_res.txt' " << endl;  ofs.close();  ofs1.close();  delete Obj\_1;  delete Obj\_2;  system("pause");  return 0;  }  Mileage\_molecule::Mileage\_molecule(double d, double p, double Tg) {  this->d = d;  this->p = p;  this->Tg = Tg;  }  Mileage\_molecule::Mileage\_molecule(const Mileage\_molecule& Mm) {  N = Mm.N;  d = Mm.d;  p = Mm.p;  Tg = Mm.Tg;  }  void Mileage\_molecule::enter\_file(const char\* S) {  ifstream ifs(S);  if (!ifs)  {  cout << "Файл не відкритий " << endl;  exit(1);  }  cout << "Файл відкритий " << endl;  ifs >> N >> d >> p >> Tg;  //cout << N << endl << d << endl << p << endl << Tg << endl;  }  void Mileage\_molecule::calc\_l() {  N = calc\_N();  if (d == 0 || N == 0) {  cout << "Помилка" << endl;  exit(2);  }  else {  l = 1.0 / (pi \* d \* d \* N \* sqrt(2));  }  }  double Mileage\_molecule::calc\_N() {  if (Tg == 0) {  cout << "Помилка" << endl;  exit(3);  }  else {  N = (6.02497 \* pow(10, 23) \* p) / (Tg \* 8.2056 \* pow(10, -5));  }  }  void Mileage\_molecule::output\_file(const char\* S) {  ofstream ofs(S);  ofs << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;  ofs << "N = " << N << endl;  ofs << "d = " << d << endl;  ofs << "p = " << p << endl;  ofs << "Tg = " << Tg << endl;  ofs << "l = " << l << endl;  ofs << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;  cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;  cout << "N = " << N << endl;  cout << "d = " << d << endl;  cout << "p = " << p << endl;  cout << "Tg = " << Tg << endl;  cout << "l = " << l << endl;  cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;  cout << "Результати y файлі " << S << endl;  }  const double Mileage\_molecule::output\_l() {  return l;  }  void Mileage\_molecule::operator=(Mileage\_molecule const& Mm) {  N = Mm.N;  d = Mm.d;  p = Mm.p;  Tg = Mm.Tg;  }  bool operator==(Mileage\_molecule const& A, Mileage\_molecule const& B) {  return A.l == B.l ? true : false;  }  bool operator!=(Mileage\_molecule const& A, Mileage\_molecule const& B) {  return A.l != B.l ? true : false;  }  bool operator<(Mileage\_molecule const& A, Mileage\_molecule const& B) {  return A.l < B.l ? true : false;  }  bool operator<=(Mileage\_molecule const& A, Mileage\_molecule const& B) {  return A.l <= B.l ? true : false;  }  bool operator>(Mileage\_molecule const& A, Mileage\_molecule const& B) {  return A.l > B.l ? true : false;  }  bool operator>=(Mileage\_molecule const& A, Mileage\_molecule const& B) {  return A.l >= B.l ? true : false;  } |





