Лабораторная работа 2.

Введение в структуры и классы.

Часть 1.

Выполните следующие задачи по ссылке (максимум 5 баллов):

https://informatics.msk.ru/mod/statements/view.php?id=91168#1

Часть 2.

Для каждой задачи заранее подготовьте примеры, аналогичные приведенным в задании.

Задача 1. (1 балл) Реализуйте класс, поддерживающий набор строк в отсортированном порядке. Класс должен содержать два публичных метода:

```
class SortedStrings {
public:
  void AddString(const string& s) {
    // добавить строку s в набор
  }
  vector<string> GetSortedStrings() {
    // получить набор из всех добавленных строк в отсортированном порядке
  }
private:
  // приватные поля
};

```

Пример

Код

```
void PrintSortedStrings(SortedStrings& strings) {
  for (const string& s : strings.GetSortedStrings()) {
    cout << s << " ";
  }
  cout << endl;
}

int main() {
  SortedStrings strings;

  strings.AddString("first");
  strings.AddString("third");
  strings.AddString("second");
  PrintSortedStrings(strings);

  strings.AddString("second");
  PrintSortedStrings(strings);

  return 0;
}

```

Вывод

```
first second third
first second second third

```

Задача 2. (1 балл)

Реализуйте класс для человека, поддерживающий историю изменений человеком своих фамилии и имени.

```cpp
class Person {
public:
  void ChangeFirstName(int year, const string& first_name) {
    // добавить факт изменения имени на first_name в год year
  }
  void ChangeLastName(int year, const string& last_name) {
    // добавить факт изменения фамилии на last_name в год year
  }
  string GetFullName(int year) {
    // получить имя и фамилию по состоянию на конец года year
  }
private:
  // приватные поля
};

```

Считайте, что в каждый год может произойти не более одного изменения фамилии и не более одного изменения имени. При этом с течением времени могут открываться всё новые факты из прошлого человека, поэтому годá в последовательных вызовах методов ChangeLastName и ChangeFirstName не обязаны возрастать.

Гарантируется, что все имена и фамилии непусты.

Строка, возвращаемая методом GetFullName, должна содержать разделённые одним пробелом имя и фамилию человека по состоянию на конец данного года.

- Если к данному году не случилось ни одного изменения фамилии и имени, верните строку **"Incognito"**.
- Если к данному году случилось изменение фамилии, но не было ни одного изменения имени, верните **"*last_name* with unknown first name"**.
- Если к данному году случилось изменение имени, но не было ни одного изменения фамилии, верните **"*first_name* with unknown last name"**.

Пример

Код

```cpp
int main() {
  Person person;

  person.ChangeFirstName(1965, "Polina");
  person.ChangeLastName(1967, "Sergeeva");
  for (int year : {1900, 1965, 1990}) {
    cout << person.GetFullName(year) << endl;
  }

  person.ChangeFirstName(1970, "Appolinaria");
  for (int year : {1969, 1970}) {
    cout << person.GetFullName(year) << endl;
  }

  person.ChangeLastName(1968, "Volkova");
  for (int year : {1969, 1970}) {
    cout << person.GetFullName(year) << endl;
  }

  return 0;
}

```

Вывод

```
Incognito
Polina with unknown last name
Polina Sergeeva
Polina Sergeeva
Appolinaria Sergeeva
Polina Volkova
Appolinaria Volkova

```

Задача 3. (2 балла)

Дополните класс из предыдущей задачи «Имена и фамилии — 1» методом GetFullNameWithHistory:

```cpp
class Person {
public:
  void ChangeFirstName(int year, const string& first_name) {
    // добавить факт изменения имени на first_name в год year
  }
  void ChangeLastName(int year, const string& last_name) {
    // добавить факт изменения фамилии на last_name в год year
  }
  string GetFullName(int year) {
    // получить имя и фамилию по состоянию на конец года year
  }
  string GetFullNameWithHistory(int year) {
    // получить все имена и фамилии по состоянию на конец года year
  }
private:
  // приватные поля
};

```

В отличие от метода GetFullName, метод GetFullNameWithHistory должен вернуть не только последние имя и фамилию к концу данного года, но ещё и все предыдущие имена, и фамилии в обратном хронологическом порядке. Если текущие факты говорят о том, что человек два раза подряд изменил фамилию или имя на одно и то же, второе изменение при формировании истории нужно игнорировать.

Для лучшего понимания формата см. примеры.

Пример 1

Код

```
int main() {
  Person person;

  person.ChangeFirstName(1900, "Eugene");
  person.ChangeLastName(1900, "Sokolov");
  person.ChangeLastName(1910, "Sokolov");
  person.ChangeFirstName(1920, "Evgeny");
  person.ChangeLastName(1930, "Sokolov");
  cout << person.GetFullNameWithHistory(1940) << endl;

  return 0;
}

```

Вывод

```
Evgeny (Eugene) Sokolov

```

Пример 2

Код

```
int main() {
  Person person;

  person.ChangeFirstName(1965, "Polina");
  person.ChangeLastName(1967, "Sergeeva");
  for (int year : {1900, 1965, 1990}) {
    cout << person.GetFullNameWithHistory(year) << endl;
  }

  person.ChangeFirstName(1970, "Appolinaria");
  for (int year : {1969, 1970}) {
    cout << person.GetFullNameWithHistory(year) << endl;
  }

  person.ChangeLastName(1968, "Volkova");
  for (int year : {1969, 1970}) {
    cout << person.GetFullNameWithHistory(year) << endl;
  }

  person.ChangeFirstName(1990, "Polina");
  person.ChangeLastName(1990, "Volkova-Sergeeva");
  cout << person.GetFullNameWithHistory(1990) << endl;

  person.ChangeFirstName(1966, "Pauline");
  cout << person.GetFullNameWithHistory(1966) << endl;

  person.ChangeLastName(1960, "Sergeeva");
  for (int year : {1960, 1967}) {
    cout << person.GetFullNameWithHistory(year) << endl;
  }

  person.ChangeLastName(1961, "Ivanova");
  cout << person.GetFullNameWithHistory(1967) << endl;

  return 0;
}

```

Вывод

```
Incognito
Polina with unknown last name
Polina Sergeeva
Polina Sergeeva
Appolinaria (Polina) Sergeeva
Polina Volkova (Sergeeva)
Appolinaria (Polina) Volkova (Sergeeva)
Polina (Appolinaria, Polina) Volkova-Sergeeva (Volkova, Sergeeva)
Pauline (Polina) with unknown last name
Sergeeva with unknown first name
Pauline (Polina) Sergeeva
Pauline (Polina) Sergeeva (Ivanova, Sergeeva)

```

Пример 3

Код

```
int main() {
  Person person;

  person.ChangeFirstName(1965, "Polina");
  person.ChangeFirstName(1965, "Appolinaria");

  person.ChangeLastName(1965, "Sergeeva");
  person.ChangeLastName(1965, "Volkova");
  person.ChangeLastName(1965, "Volkova-Sergeeva");

  for (int year : {1964, 1965, 1966}) {
    cout << person.GetFullNameWithHistory(year) << endl;
  }

  return 0;
}
```

Вывод

```
Incognito
Appolinaria Volkova-Sergeeva
Appolinaria Volkova-Sergeeva

```

Задача 4. (2 балла)

Реализуйте класс ReversibleString, хранящий строку и поддерживающий методы Reverse для переворота строки и ToString для получения строки.

Пример

Код

```
int main() {
  ReversibleString s("live");
  s.Reverse();
  cout << s.ToString() << endl;

  s.Reverse();
  const ReversibleString& s_ref = s;
  string tmp = s_ref.ToString();
  cout << tmp << endl;

  ReversibleString empty;
  cout << '"' << empty.ToString() << '"' << endl;

  return 0;
}

```

Вывод

```
evil
live
""

```

Задача 5. (2 балла)

Дополните класс Person из задачи 2 конструктором, позволяющим задать имя и фамилию человека при рождении, а также сам год рождения. Класс не должен иметь конструктора по умолчанию.

При получении на вход года, который меньше года рождения:

- методы GetFullName и GetFullNameWithHistory должны отдавать **"No person"**;
- методы ChangeFirstName и ChangeLastName должны игнорировать запрос.

Кроме того, необходимо объявить константными все методы, которые по сути ими являются.

Пример

Код

```
int main() {
  Person person("Polina", "Sergeeva", 1960);
  for (int year : {1959, 1960}) {
    cout << person.GetFullNameWithHistory(year) << endl;
  }

  person.ChangeFirstName(1965, "Appolinaria");
  person.ChangeLastName(1967, "Ivanova");
  for (int year : {1965, 1967}) {
    cout << person.GetFullNameWithHistory(year) << endl;
  }

  return 0;
}
```

Вывод

```
No person
Polina Sergeeva
Appolinaria (Polina) Sergeeva
Appolinaria (Polina) Ivanova (Sergeeva)
```

Задача 6. (максимум 5 баллов)

Во всех заданиях предусмотрите конструкторы с аргументами по умолчанию, а также дружественную перегруженную операцию вывода в поток и чтения из потока. Помните, что в каждом классе должны быть предусмотрены константные функции get.

Варианты

1. Создайте класс с именем Complex для выполнения арифметических действий с комплексными числами. Напишите программу для проверки вашего класса. Комплексные числа должны быть представлены в форме *Re+Im*i*, где i*i = –1. Используйте переменные с плавающей точкой

для представления закрытых данных класса.

Создайте конструктор, деструктор и открытые методы для следующих действий:

a) сложение 2 комплексных чисел;

b) вычитание 2 комплексных чисел;

c) умножение 2 комплексных чисел.

Перегрузите операцию вывода в поток для печати комплексного числа в виде $z = r * e^{i\varphi}$ (В показательной форме)

2.Создайте класс с именем Rational для выполнения арифметических действий с дробями. Напишите программу для проверки вашего класса. Используйте целые переменные для представления закрытых данных класса – числителя и знаменателя. Создайте функцию-конструктор класса, которая позволяет объекту этого класса принимать начальные значения при его объявлении. Конструктор должен содержать значения по умолчанию на случай отсутствия заданных начальных и должен хранить дроби в сокращенном виде. Создайте открытые методы для случаев:

a) сложение 2 дробей (здесь и далее результат должен храниться в сокращенном виде);

b) вычитание 2 дробей;

c) перемножение 2 дробей;

d) деление 2 дробей;

e) печать дроби в десятичном виде.

Перегрузите операцию вывода в поток для печати дроби в виде a/b.

3. То же, что и в варианте 2, но для дроби, числитель и знаменатель которой – комплексные числа.

Перегруженная операция печати должна выглядеть следующим образом: на экран должна выводиться дробь в виде (ReA+i*ImA)/(ReB+i*ImB).

4. Напишите класс MyTime, который позволяет работать со временем. Включите метод tick, который дает приращение времени, хранящегося в объекте myTime, равное одной секунде. Объект Time должен всегда находиться в непротиворечивом состоянии. Напишите программу для проверки метода tick в цикле, которая печатала бы время в каком-либо стандартном формате на каждой итерации цикла и иллюстрировала правильную работу метода tick. Удостоверьтесь в правильности работы в следующих случаях:

a) приращение с переходом в следующую минуту;

b) приращение с переходом в другой час;

c) приращение с переходом в другой день.

Предусмотрите перегруженную операцию вывода в поток, а также перегруженную операцию инкремента в префиксной и постфиксной формах.

5. Напишите класс MyDate, который позволяет работать со временем. Резализуйте метод nextDay, который будет увеличивать дату на 1 день. Объект myDate должен всегда находиться в непротиворечивом состоянии. Напишите программу, проверяющую функцию nextDay в цикле и печатающую время в стандартном формате на каждой итерации цикла. Проверьте правильность работы функции в следующих случаях:

a) приращение с переходом в следующий месяц;

b) приращение с переходом в следующий год.

Предусмотрите перегруженную операцию вывода в поток, а также перегруженную операцию инкремента в префиксной и постфиксной формах.

6. Создайте класс параллелограмм, который хранит только декартовы координаты его четырех углов. Конструктор вызывает набор методов, которые принимают 4 группы координат и проверяют, чтобы каждая из координат x и y находилась в первом квадранте в диапазоне от 0,0 до 50,0. Это множество функций также должно проверять, что переданные координаты определяют параллелограмм. Должны быть предусмотрены методы, вычисляющие длины сторон параллелограмма, периметр и площадь. Включите функцию, которая определяла бы, не является ли параллелограмм прямоугольником.

Перегрузите операцию вывода в поток для печати всей информации об объекте.

7. Создайте класс Rectangle (прямоугольник). Класс имеет атрибуты length (длина) и width (ширина), каждый из которых по умолчанию равен 1. Он имеет методы, которые вычисляют периметр и площадь прямоугольника. Он имеет функции записи и чтения для длины и ширины.

Функции записи должны проверять, что длина и ширина – числа с плавающей запятой, находя щиеся в пределах от 0,0 до 20,0.

Перегрузите операцию вывод в поток для печати всей информации о прямоугольнике.

8. Создайте класс прямоугольник, который хранит только декартовы координаты четырех углов прямоугольника. Конструктор вызывает набор функций, которые принимают 4 группы координат и проверяют, чтобы каждая из координат x и y находилась в первом квадранте в диапазоне от 0,0 до 20,0. Это множество методов также должно проверять, что переданные координаты определяют прямоугольник. Должны быть предусмотрены методы, вычисляющие длину и ширину прямоугольника, периметр и площадь. Включите функцию, которая определяла бы, не является ли прямоугольник квадратом.

Перегрузите операцию вывод в поток для печати всей информации о прямоугольнике.

9. Модифицируйте класс прямоугольник из варианта №8 так, чтобы включить в него функцию draw, которая изображает прямоугольник внутри окна 25 на 25. Включите функцию setFillCharacter, чтобы задавать символ, которым будет заполняться прямоугольник внутри. Включите функцию setPerimeterCharacter, чтобы задавать символ, которым будут печататься границы прямоугольника. Включите функцию поворота прямоугольника на 90 градусов вокруг одной из его вершин против и по часовой стрелке.

Перегрузите операцию вывод в поток для печати всей информации о прямоугольнике и рисования прямоугольника.

10. Создайте класс треугольник, хранящий только декартовы координаты вершин. Конструктор вызывает набор методов, которые принимают 3 группы координат и проверяют, чтобы каждая из координат x и y находилась в первом квадранте в диапазоне от 0,0 до 50,0. Методы также должны проверять, чтобы треугольник не «схлопывался» в прямую линию. Должны быть предусмотрены методы, вычисляющие длину сторон, периметр и площадь треугольника. Включите функцию, которая определяла бы, не является ли треугольник равнобедренным, равносторонним или прямоугольным.

Перегрузите операцию вывода в поток для печати всей информации о треугольнике.

11. Создайте класс треугольник, хранящий длины двух сторон и значение угла между ними. Должны быть предусмотрены методы, вычисляющие длину третьей стороны, значения 2 оставшихся углов, периметр и площадь треугольника. Включите функцию, которая определяла бы, не является ли треугольник равнобедренным, равносторонним или прямоугольным.

Перегрузите операцию вывода в поток для печати всей информации о треугольнике.

12. Создайте класс прямая призма, хранящий только декартовы координаты вершин основания и высоту призмы. Конструктор вызывает набор функций, которые принимают группы координат и проверяют, чтобы каждая из координат x и y находилась в первом квадранте в диапазоне от 0,0 до 250,0. Должны быть предусмотрены методы, вычисляющие длину ребер, периметр и площадь основания, а также площадь боковой поверхности, площадь поверхности и объем призмы.

Перегрузите операцию вывода в поток так, чтобы она печатала, какая фигура лежит в основании, и ее основные характеристики.

13. Создайте класс пирамида, хранящий только декартовы координаты вершин основания и вершины пирамиды. Конструктор вызывает набор функций, которые принимают группы координаты. Должны быть предусмотрены методы, вычисляющие длину ребер, периметр и площадь основания, а также площадь боковой поверхности, площадь поверхности и объем пирамиды.

Перегрузите операцию вывода в поток так, чтобы она печатала, какая фигура лежит в основании, и ее основные характеристики.

14. Создайте класс конус, хранящий только декартовы координаты центра основания, радиус основания и высоту конуса. Должны быть предусмотрены методы, рассчитывающие периметр и площадь основания, а также площадь боковой поверхности, площадь поверхности и объем конуса.

Перегрузите операцию вывода в поток для печати всей информации об объекте.

15. Создайте класс усеченный конус, хранящий только декартовы координаты центра основания, радиусы оснований и высоту конуса. Должны быть предусмотрены методы, рассчитывающие периметр и площадь оснований, а также площадь боковой поверхности, площадь поверхности и объем конуса.

Перегрузите операцию вывода в поток для печати всей информации об объекте.

16. Создайте класс усеченная пирамида, хранящий только декартовы координаты вершин оснований. Конструктор вызывает набор функций, которые принимают группы координат одного основания. Высота пирамиды задается случайным образом, координаты второго основания вычисляются в соответствии с высотой. Должны быть предусмотрены методы, вычисляющие периметр и площадь основания, а также площадь боковой поверхности, площадь поверхности и объем пирамиды.

Перегрузите операцию вывода в поток для печати всей информации об объекте (какая фигура лежит в основании, ее основные характеристики).

17. Создайте класс "Спортсмен", в котором будут храниться данные о ФИО атлета, представляемой стране, виде спорта, установленных рекордах, завоеванных местах на первенствах страны, континента, мира и олимпиадные достижения. Предусмотрите методы "Установить рекорд", "Получить медаль"

(помните, что медали бывают разные...). Предусмотрите также все необходимые методы установки и чтения данных-элементов.

Перегрузите операцию вывода в поток для вывода информации о спортсмене.

18. Создайте класс «Запись в адресной книге». В нем хранятся фамилия и имя человека, номера телефонов (нескольких, в т.ч. домашних, рабочих и сотовых), район и адрес проживания, e-mail (несколько). Конструктор должен вызывать функцию, считывающую эти данные из текстового файла.

Напишите методы для установки и чтения данных. Предусмотрите метод поиска номеров сотовых телефонов, метод формирования текстового файла с заголовком и подписью (как заготовки письма на электронный адрес).

Перегрузите операцию вывода в поток для печати информации об объекте вида «Запись в адресной книге».

19. Создайте класс матрица размерностью n на n , который хранит только размерность матрицы и максимальное по модулю значение элемента матрицы (и указатель на целое). Конструктор должен вызывать функцию заполнения матрицы случайными числами в заданном диапазоне. Напишите методы для:

a) транспонирования матрицы;

b) умножения матрицы на число;

c) сложения матриц;

d) умножения двух матриц.

Перегрузите операцию вывода в поток для печати всей информации об объекте. Перегрузите также оператор вычитания матриц.

20. Создайте класс правильный многоугольник, который хранит число вершин и их координаты. Конструктор вызывает набор функций, которые проверяют, чтобы число вершин было не менее 3, чтобы многоугольник был правильным, и в случае ошибки устанавливают значения всех вершин в 0.

Должны быть предусмотрены методы, вычисляющие периметр, площадь многоугольника. Должна быть предусмотрен метод вывода на печать информации о числе сторон многоугольника. Напишите программу-драйвер, иллюстрирующую применение вашего класса.

Перегрузите операцию вывода в поток для печати всей информации об объекте.