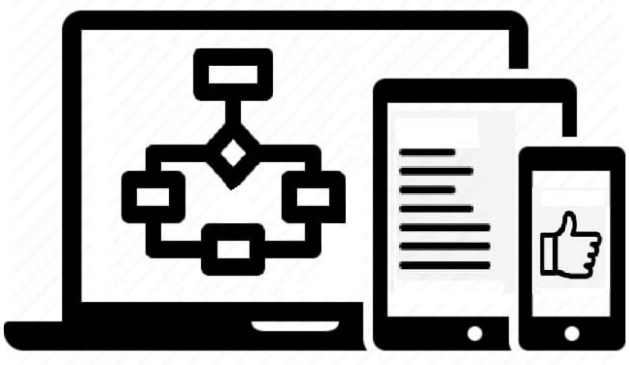
МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Программное обеспечение информационных технологий»

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ И ПРОЕКТИРОВАНИЕ

Методические рекомендации к самостоятельной работе для студентов специальности 1-53 01 02 «Автоматизированные системы обработки информации» заочной формы обучения



Могилев 2022

УДК 621.01
ББК 36.4
О87

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой «Программное обеспечение информационных технологий» «30» сентября 2022 г., протокол № 3

Составители: ст. преподаватель О. В. Сергиенко;
ст. преподаватель Ю. В. Вайнилович;
доц. Н. Н. Горбатенко

Рецензент канд. техн. наук, доц. И. В. Лесковец

Методические рекомендации к самостоятельной работе студентов специальности 1-53 01 02 «Автоматизированные системы обработки информации» заочной формы обучения содержат примеры решения задач, перечень задач для самостоятельного решения, контрольные вопросы для самопроверки.

Учебно-методическое издание

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ И ПРОЕКТИРОВАНИЕ

Ответственный за выпуск	В. В. Кутузов
Корректор	И. В. Голубцова
Компьютерная верстка	Н. П. Полевничая

Подписано в печать . Формат 60x84/16. Бумага офсетная. Гарнитура Таймс.
Печать трафаретная. Усл. печ. л. . Уч.-изд. л. . Тираж 21 экз. Заказ №

Издатель и полиграфическое исполнение:
Межгосударственное образовательное учреждение высшего образования
«Белорусско-Российский университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/156 от 07.03.2019.
Пр-т Мира, 43, 212022, г. Могилев.

Содержание

Введение

Цель изучения дисциплины – освоение методов построения сложных программ и систем с применением объектно-ориентированного программирования.

В методических рекомендациях приводятся пример решения задачи, перечень задач для самостоятельного решения, перечень вопросов для самопроверки глубины усвоения дисциплины.

В результате изучения методических рекомендаций студент приобретает умения применять принципы объектно-ориентированного программирования при разработке программ.

Подразумевается, что студент знаком с теоретическим материалом, изложенным в [1–6].

Методические рекомендации призваны помочь студентам организовать самостоятельное изучение дисциплины «Объектно-ориентированное программирование и проектирование».

Выполненное индивидуальное практическое задание демонстрируется преподавателю на компьютере с необходимыми устными комментариями.

Отчет о выполненном задании содержит условие задачи, программный код, исходные данные для тестирования и результаты тестирования.

Для более глубокой проверки знаний по теме организуется устная беседа или письменные ответы на контрольные вопросы.

Результативность выполнения заданий может быть учтена при выставлении итоговой оценки по дисциплине на усмотрение преподавателя.

1 Пример решения задачи

Условие

Дан объект: цвет в формате RGB, который является элементом для набора. Элемент состоит из компонент: красная [0…255], зеленая [0…255], синяя [0…255], которые хранятся в нем как структура битовых полей.

Описание

Цветовая модель RGB наиболее часто используется при описании цветов, получаемых смешением световых лучей. Она подходит для описания цветов, отображаемых на экране мониторов, получаемых со сканера, смешиваемых цветовыми фильтрами, но не подходит для печатающих устройств. Цвет в модели RGB представляется как сумма трѐх базовых компонент: красная (Red), зеленая (Green) и синяя (Blue). Из первых букв английских названий этих цветов составлено название модели. На рисунке 1 показано, что получается при сложении базовых цветов. На рисунке 2 показано размещение значений компонент цвета внутри структуры данных.

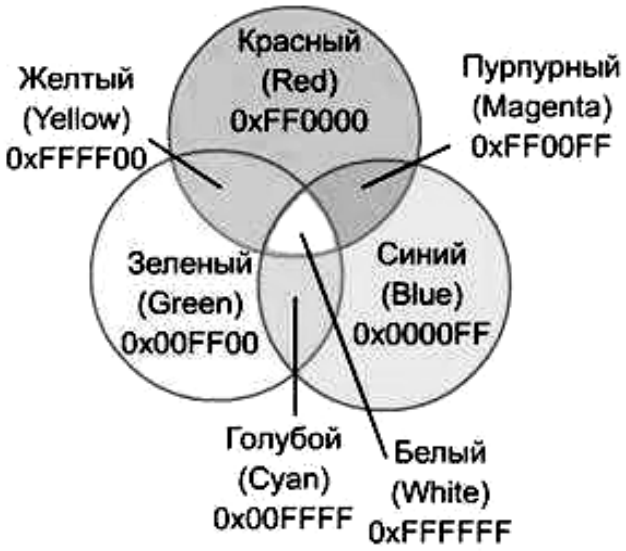


Рисунок 1 – Сложение цветов модели RGB

В модели RGB каждый базовый цвет характеризуется интенсивностью, которая может принимать 256 дискретных значений от 0 до 255, что позволяет смешивать цвета в различных пропорциях, варьируя каждой составляющей. Таким образом, можно получить 256·256·256 = 16 777 216 оттенков цвета. Каждое значение от 0 до 255 можно представить 8-разрядным двоичным числом ($256 = 2^8$), т. е. одним байтом. Таким образом, каждый цвет кодируется тремя байтами (по одному байту на каждый цвет) или 24 битами.

Каждому цвету сопоставляет код, содержащий значения интенсивности трѐх базовых составляющих. Для кода цвета используются десятичное и

шестнадцатеричное представления. Десятичное представление – это тройка десятичных чисел, разделенных запятыми: первое число – красная, второе – зеленая, а третье – синяя составляющая. Код цвета в шестнадцатеричном представлении имеет вид 0xXXXXXX. За префиксом 0x следуют шесть шестнадцатеричных цифр (0, 1, …, 9, A, …, F). Первая и вторая цифры представляют красную, третья и четвертая – зеленую, пятая и шестая – синюю составляющую цвета соответственно. Максимальная интенсивность (255,255,255) – 0xFFFFFF соответствует белому цвету. Минимальная интенсивность (0,0,0) – 0x000000 соответствует черному цвету. Смешение составляющих с различными, но одинаковыми интенсивностями дает шкалу из 256 оттенков серого цвета – от черного до белого.

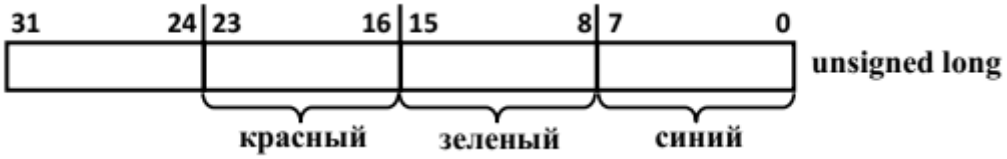


Рисунок 2 – Диаграмма битовых полей

Листинг программы

```
#include <iostream> using namespace std;
/*  Класс реализует элемент «цвет в формате RGB»
*/
class Color { private:
union {
unsigned long number;    // упакованное значение цвета struct {    //
структура компонент цвета:
unsigned red:    8;    // красный [0..255]
unsigned green: 8;    // зеленый [0..255]
unsigned blue:  8; // синий  [0..255] unsigned unused: 8;   //
не используется
} components;         // компоненты
} color;              // внутреннее представление цвета protected:
// Проверка значения компонента цвета int accurateValue(int value)
{ return ((value < 0) ? 0 : (value > 255) ? 255 : value); } public:
// Дефолтный конструктор Color() { color.number = 0; }
// Копирующий конструктор Color(const Color& obj)
{ color.number = obj.color.number; }
// Параметрический конструктор Color(int red, int green, int blue)
{ setRed(red); setGreen(green); setBlue(blue); }
// Установка компонент цвета void setRed(int red)
{ color.components.red = accurateValue(red); } void setBlue(int blue)
{ color.components.blue = accurateValue(blue); } void setGreen(int
green)
{ color.components.green = accurateValue(green); }
// Получение компонент цвета
```

```
      int getRed() const { return int(color.components.red); } int get-
Blue() const { return int(color.components.blue); } int getGreen() const {
return int(color.components.green); }
      // Получение цвета как целого беззнакового числа
      unsigned long getValue() const { return color.number; }
      // Инверсия компонент цвета void Inverse()
      {
      color.components.red = 255 - color.components.red; col-
or.components.blue = 255 - color.components.blue; color.components.green = 255
- color.components.green;
      }
      };
      /* Класс реализует набор «палитра цветов»
      */
      class Palette { private:
      Color* items;     // Указатель на массив элементов (набор)
      int begin;        // Базовый индекс массива (элемента в наборе) int
size;                   // Размер массива (набора)
      protected:
  static Color ground;  // Дефолтный цвет набора public:
      // Параметрический конструктор (и дефолтный одновременно) Palette(int
count = 2)
      {
      begin = 0;
      size = ((count < 2) ? 2 : (count > 99) ? 99 : count); if((items = new
Color[size]) == NULL) size = 0;
      }
      // Копирующий конструктор Palette(const Palette& pal)
      {
      size = pal.size; begin = pal.begin;
      if((items = new Color[size]) == NULL) size = 0; else for(int i = 0 ; i <
size ; i++)
      items[i] = pal.items[i];
      }
      // Деструктор
      ~Palette() { if(size != 0) delete [] items; }
      // Получение размера набора
      int Count() const { return size; }
      // Установка стартового индекса для набора
      void BeginAt(const int start_index) { begin = start_index; }
      // Получение индекса первого элемента в наборе int Start() const {
return begin; }
      // Получение индекса последнего элемента в наборе int End() const {
return (begin + size - 1); }
      // Получение элемента из набора с заданным индексом Color& Item(const
int idx) const
      { return ((idx >= Start() && idx <= End()) ? items[idx - begin] :
ground);
      }
      };
      // Инициализация дефолтного цвета через дефолтный конструктор Color
Palette::ground;
      /* Функция реализует отображение ошибки
```

```
        */
        void ErrorValue()
        {
        cin.clear();  cin.sync();
        cout << "Ошибка: некорректное значение" << endl;
        }
        /*  Функция реализует консольный интерфейс для изменения элемента (цвета)
        */
        void ModifyColor(Color& color)
        {
        char ch = '\0';     // Команда меню
        int value;          // Значение компоненты
do {                        // Обработка команд меню switch(ch)
        {
        case 'P':           // Печать компонент цвета case 'p':
        cout << endl <<"Красный = " << color.getRed()
        << endl <<"Зеленый = " << color.getGreen()
        << endl <<"Синий  = " << color.getBlue()
      << endl; break;
        case 'R':           // Установка красной компоненты case 'r':
        cout << endl << "Введите красную компоненту [0..255]:>"; cin >> value;
        if(!cin.fail()) color.setRed(value); else ErrorValue();
        break;
        case 'G':      // Установка зеленой компоненты case 'g':
        cout << endl << "Введите зеленую компоненту [0..255]:>"; cin >> value;
        if(!cin.fail()) color.setGreen(value); else ErrorValue();
        break;
        case 'B':           // Установка синей компоненты case 'b':
        cout << endl << "Введите синюю компоненту [0..255]:>"; cin >> value;
        if(!cin.fail()) color.setBlue(value); else ErrorValue();
        break;
        case 'I':           // Инвертирование всех компонент case 'i':
        color.Inverse(); break;
        case 'F':           // Установка компонент в случайные значения case
  'f':
        color.setRed(rand()%256);       color.setBlue(rand()%256);       col-
  or.setGreen(rand()%256); break;
case '\0':                  // Пропустить действие break;
        default: cout << "Ошибка: некорректная операция" << endl;
        }
        // Отображение консольного меню cout << endl << "Цвет = "
        << showbase << hex << color.getValue() << dec << noshowbase
        << endl << "(P) Печать всех компонент"
        << endl << "(R) Изменение красной компоненты"
        << endl << "(G) Изменение зеленой компоненты"
        << endl << "(B) Изменение синей компоненты"
        << endl << "(I) Инвертирование всех компонент"
        << endl << "(F) Заполнение случайными значениями"
        << endl << "(O) Выход";
        cout << endl << "Выберите пункт:>"; cin >> ch;
        }
        while(ch != 'O' && ch != 'o');
        }
```

```
        /* Функция реализует консольный интерфейс для изменения набора (палитры)
        */
        void ModifyPalette(Palette& table)
        {
        char ch = 'P';      // Команда меню
        int num;            // Значение для ввода
do {                        // Обработка команд меню switch(ch)
        {
        case 'P':           // Печать содержания набора case 'p':
        cout << endl << "Содержание палитры цветов:"; for(num = table.Start() ;
  num <= table.End() ; num++)
        cout << endl << num
        << ". (" << table.Item(num).getRed()
        << "," << table.Item(num).getGreen()
        << "," << table.Item(num).getBlue()
        << ") = " << table.Item(num).getValue(); cout << endl;
        break;
        case 'M':           // Изменение элемента в наборе case 'm':
        cout << endl << "Введите индекс цвета:>"; cin >> num;
        if(!cin.fail()) ModifyColor(table.Item(num)); else ErrorValue();
        break;
        case 'I':           // Установка начального индекса case 'i':
        cout << endl << "Введите начальный индекс:>"; cin >> num;
        if(!cin.fail()) table.BeginAt(num); else ErrorValue();
        ch = 'P';
        continue;
        case 'F':       // Заполнение набора случайными значениями case 'f':
        for(num = table.Start() ; num <= table.End() ; num++)
        {
        table.Item(num).setRed(rand()%256);
        table.Item(num).setGreen(rand()%256);
        table.Item(num).setBlue(rand()%256);
        }
        ch = 'P';
        continue;
        default: cout << "Ошибка: некорректная операция" << endl;
        }
        // Отображение консольного меню
        cout << endl << "(P) Печать всей палитры"
        << endl << "(M) Изменение цвета в палитре"
        << endl << "(I) Установка начального индекса"
        << endl << "(F) Заполнение случайными значениями"
        << endl << "(Q) Выход";
        cout << endl << "Выберите пункт:>"; cin >> ch;
        }
        while(ch != 'Q' && ch != 'q');
        }
        /* Главная функция */ void main()
        {
        // Переключить вывод сообщений на русском языке setlocale(LC_ALL,
  "Russian");
        cout << "Работа 1";
        int count;     // Размер набора (=0 выход) do {
```

```
      // Ввод размера набора
      cout << endl << "Введите размер палитры [=0 выход]:>"; cin >> count;
      if(!cin.fail())
      if(count == 0) break;              // Выход, если ноль, иначе
else ModifyPalette(Palette(count));      // Создание и изменение набора
      else ErrorValue();                 // Вывод ошибки
      }
      while(1);
      }
```

Пример использования

```
Введите размер палитры [=0 выход]:>3 Содержание палитры цветов:
   0.      (0,0,0)= 0
   1.      (0,0,0)= 0
   2.      (0,0,0)= 0
   (P)     Печать всей палитры
(M)    Изменение цвета в палитре
(I)    Установка начального индекса
(F) Заполнение случайными значениями
(Q) Выход Выберите пункт:>F
Содержание палитры цветов:
0. (41,35,190) = 12460841
1. (132,225,108) = 7135620
2. (214,174,82) = 5418710
Выберите пункт:>I
Введите начальный индекс:>6 Содержание палитры цветов:
6. (41,35,190) = 12460841
7. (132,225,108) = 7135620
8. (214,174,82) = 5418710
Выберите пункт:>M Введите индекс цвета:>7
**********************************************************************
Цвет = 0x6ce184
(P) Печать всех компонент
(R) Изменение красной компоненты
(G) Изменение зеленой компоненты
(B)    Изменение синей компоненты
(I)    Инвертирование всех компонент
(F) Заполнение случайными значениями
(O) Выход Выберите пункт:>R
Введите красную компоненту [0..255]:>10 Цвет = 0x6ce10a
Выберите пункт:>G
Введите зеленую компоненту [0..255]:>20 Цвет = 0x6c140a
Выберите пункт:>B
Введите синюю компоненту [0..255]:>30
Цвет = 0x1e140a Выберите пункт:>P Красный = 10
Зеленый = 20
Синий  = 30
Выберите пункт:>I
```

```
Цвет = 0xe1ebf5
Выберите пункт:>P
Красный = 245
Зеленый = 235
Синий   = 225
Цвет = 0xe1ebf5
Выберите пункт:>O
**************************************************************************
Выберите пункт:>P
Содержание палитры цветов:
6. (41,35,190) = 12460841
7. (245,235,225) = 14806005
8. (214,174,82) = 5418710
Выберите пункт:>Q
**************************************************************************
Введите размер палитры [=0 выход]:>-5
Содержание палитры цветов:
0. (0,0,0) = 0
1. (0,0,0) = 0
Выберите пункт:>Q
**************************************************************************
Введите размер палитры [=0 выход]:>0
Завершение программы
```

2 Задание для самостоятельного решения № 1

Условие

Согласно вариантам дан объект, который является элементом для набора. Элемент состоит из компонент, которые хранятся в нем. Над элементом определены операции:

- получение значения компоненты элемента;
- установка и инициализация значения компоненты элемента;
- контроль значения компоненты элемента (на допустимый диапазон);
- копирование элемента.

Из элементов строится набор. Элементы в наборе проиндексированы от стартового значения. Размер набора задается при создании. Над набором определены операции:

- установка стартового индекса, получение диапазона индексов;
- заполнение набора случайными значениями;
- получение и изменение элемента набора по индексу;
- сортировка элементов по возрастанию и по убыванию;
- дополнительные операции согласно вариантам.

Необходимо разработать:

- класс для описания элемента и его свойств;

– класс для описания набора и его свойств;

– методы работы с элементом и с набором для перечисленных операций;

– дефолтный, копирующий, параметрический конструкторы для создания экземпляров набора и экземпляров элемента;

– интерфейс для редактирования элемента и интерфейс для редактирования набора, отображения и изменения их свойств.

Требования

Интерфейс и все классы реализуются в одном модуле. Для редактирования элемента разработать функцию *ModifyElement*(), которая должна получать ссылку на экземпляр элемента и предоставлять интерактивный консольный интерфейс для работы с ним. Для редактирования набора разработать функцию *ModifyPalette*(), которая должна получать ссылку на экземпляр набора и предоставлять интерактивный консольный интерфейс для работы с ним. Функция Main() запрашивает количество элементов, создаѐт экземпляр набора параметрическим конструктором и вызывает функцию *ModifyPalette*(), которая использует *ModifyElement*(). Использовать библиотеку ввода-вывода iostream.

Комментарии

Набор хранит элементы как динамический массив (рисунок 3), определяя его размер при создании. При копировании набора копируются все его элементы.

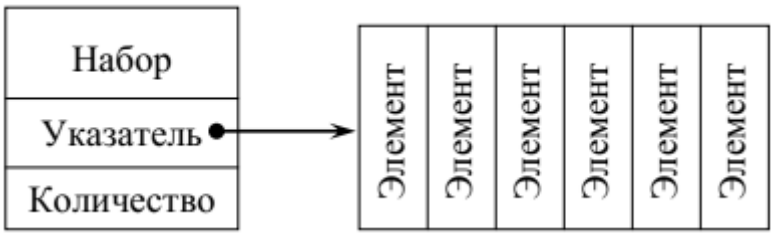


Рисунок 3 – Структура элемента

Варианты заданий

1 Элемент: цвет в формате CYMK. Дополнительно: сортировка по компонентам и целиком, пересчѐт в RGB.

2 Элемент: положение солнца в координатах *a*-азимут, *z*-зенит, *h*-горизонт. Дополнительно: сортировка по компонентам и целиком, пересчет в другую систему координат (*на выбор*).

3 Элемент: цвет в формате YUV. Дополнительно: сортировка по компонентам и целиком, пересчѐт в RGB.

4 Элемент: положение солнца в экваториальных координатах δ – склонение, *p* – полярное расстояние, *t* – часовой угол. Дополнительно: сортировка

по компонентам и целиком, пересчѐт в другую систему координат (*на выбор*).

5 Элемент: цвет в формате AHSL. Дополнительно: сортировка по компонентам и целиком, пересчет в RGB.

6 Элемент: цвет в формате RYB. Дополнительно: сортировка по компонентам и целиком, пересчет в RGB.

7 Элемент: декартовы координаты в пространстве (*x*, *y*, *z*)-координаты. Дополнительно: сортировка по компонентам и целиком, пересчѐт в цилиндрическую систему координат.

8 Элемент: цвет в формате YIQ. Дополнительно: сортировка по компонентам и целиком, пересчет в RGB.

9 Элемент: время в формате *h* – часы, *m* – минуты, *s* – секунды. Дополнительно: сортировка по компонентам и целиком, пересчет в 12-часовой формат времени (a. m. и p. m.).

10 Элемент: цвет в формате HSV. Дополнительно: сортировка по компонентам и целиком, пересчет в RGB.

Используйте ru.wikipedia.org для получения информации о форматах.

3 Задание для самостоятельного решения № 2

Условие

Дан объект данных, над которым определены операции **согласно вариантам**. Реализовать набор операций для работы с объектом так, чтобы его можно было использовать в выражениях, не прибегая к вызову функций.

Необходимо разработать:

- класс объекта и определить правила выполнения операций над ним;
- набор перегруженных операторов, реализующих операции с объектом;
- интерфейс для редактирования объекта с помощью операторов;
- интерфейс для тестирования использования объекта в выражениях.

Требования

Интерфейс и класс объекта реализуются в одном модуле. Для редактирования объекта разработать функцию Modify*Object*(), которая должна получать ссылку на экземпляр объекта и предоставлять интерактивный консольный интерфейс для работы с ним. Для тестирования использования объекта в выражениях разработать функцию Main(), которая должна предоставлять интерактивный консольный интерфейс, демонстрирующий применение объекта в выражениях.

Обязательной реализации подлежат следующие операции: сложение (+ и +=), вычитание (– и –=), умножение (* и *=), сравнение на равенство (== и !=), унарные (+, –), инверсия (~), присваивание (=), проверка на ноль (!), преобразование к типу (type), ввод из потока (cin >>) и вывод в поток (cout <<).

Разработать набор тестовых примеров, демонстрирующих использование перегруженных операторов в арифметических и логических выражениях. Использовать библиотеку ввода-вывода iostream.

Варианты заданий

1 Объект: комплексное число (вещественная и мнимая части). Принять: (!) – проверка на ноль, (+ и +=) – сложение, (– и –=) – вычитание, (* и *=) – умножение, (== и !=) – сравнение, (double) – вычисление модуля, (float) – вычисление аргумента, (~) – сопряженное число.

2 Объект: интервал времени (часы, минуты, секунды). Реализовать операции с учетом ограничений на часы (0 до 23), минуты и секунды (0 до 59), т. е. результат всегда от 0:0:0 до 23:59:59. Принять: (+ и +=) – сложение, (– и –=) – вычитание, (* и *=) – удлинение или сокращение, (!) – проверка на ноль, (== и !=) – сравнение, (long) – преобразование в секунды, (float) – преобразование в часы (3600 сек), (~) – дополнение до конца суток.

3 Объект: денежная сумма (признак валюты [р., $], сумма в номинале [рубли, доллары], сумма в размене [копейки, центы]). Реализовать операции с учетом конвертации, если валюты не совпадают. Принять: (+ и +=) – сложение, (– и –=) – вычитание, (* и *=) – умножение, (!) – проверка на ноль, (== и !=) – сравнение, (float) – в номинал, (int) – в размен, (~) – изменение признака валюты с конвертацией, (%) – процент от суммы.

4 Объект: интервал даты (часов, дней, лет). Реализовать операции с учетом столетия (0 до 99) и ограничений на дни (0 до 364) и часы (0 до 23), т. е. результат всегда от 0-0-0 до 23-364-99. Принять: (* и *=) – удлинение или сокращение, (+ и +=) – сложение, (– и –=) – вычитание, (== и !=) – сравнение, (!) – проверка на ноль, (long) – преобразование в часы, (float) – преобразование в года (365 дней), (~) – дополнение до конца столетия.

5 Объект: расстояние (сажень, аршин, вершок). 1 сажень = 3 аршинам, 1 аршин = 16 вершкам, 1 вершок = 44,5 мм. Результат всегда от 0 до 500 саженей (1 верста). Принять: (+ и +=) – сложение, (– и –=) – разность, (* и *=) – удлинение или сокращение, (== и !=) – сравнение, (!) – проверка на ноль, (double) – преобразование в миллиметры, (int) – преобразование в вершки, (~) – дополнение до версты (500 саженей).

6 Объект: строка символов (0 до 128). Принять: (+ и +=) – соединение строк, повторение символа, (– и –=) – отсечение строки, (*) – поиск подстроки, (*=) – заполнение подстрокой или символом, (== и !=) – сравнение, (!) – проверка на пусто, (~) – переворот наоборот, (int) – длина строки.

7 Объект: натуральная дробь (целое, числитель, знаменатель). Реализовать операции с учѐтом приведения к общему знаменателю. Принять: (+ и +=) – сложение, (– и –=) – вычитание, (* и *=) – умножение, (!) – проверка на ноль, (== и !=) – сравнение, (double) – преобразование в рациональную дробь, (~) – взаимообратная натуральная дробь.

8 Объект: угол (градусы, минуты, секунды). Реализовать операции с учѐтом целых оборотов и ограничений на градусы (0 до 359), минуты и секунды

(0 до 59), т. е. результат всегда от 0·0·0· до $359 \cdot 59 \cdot 59^0$. Принять: (+ и +=) – сложение, (– и –=) – вычитание, (* и *=) – умножение, (!) – проверка на ноль, (== и !=) – сравнение, (double) – преобразование в радианы, (int) – преобразование в секунды, (~) – обратный угол до 360^0.

9 Объект: квадратная матрица [3x3]. Реализовать операции над матрицами. Принять: (+ и +=) – сложение, (– и –=) – вычитание, (* и *=) – умножение, (!) – проверка на ноль, (== и !=) – сравнение, (double) – вычисление детерминанта, (int) – количество ячеек, (~) – транспонирование.

10 Объект: алфавит (только прописные от A до Z). Реализовать операции над алфавитами как над множествами. Принять: (+ и +=) – объединение, (!) – проверка на пусто, (– и –=) – разность, (* и *=) – пересечение, (== и !=) – сравнение, (int) – количество букв, (~) – отрицание как замена на буквы, которых нет.

4 Задание для самостоятельного решения № 3

Условие

Дана фигура на плоскости **согласно вариантам**. Фигура описывается индивидуальными геометрическими свойствами и общими оформительскими свойствами: цвет, видимость. У фигуры имеются характеристики: периметр, площадь, ограничивающая область. Область размещения фигур в плоскости ограничена экстентами, за которые фигура не должна выходить.

Необходимо разработать:

- классы для описания положения Location и ограничивающей области Clip в плоскости;
- статический класс Geometry для хранения общих констант и методов проверки различных ограничений на размещение фигур в плоскости;
- класс геометрического примитива Primitive для хранения и редактирования оформительских свойств фигуры как наследника от статического класса Geometry;
- класс примитивной фигуры – точки Point как наследника от классов Location и Primitive;
- класс фигуры согласно варианту Figure как наследника от класса Point с описанием специфических свойств и методов фигуры;
- наборы конструкторов для создания экземпляров каждого класса различными способами (дефолтный, копирующий, параметрический);
- методы для изменения свойств и вычисления характеристик фигуры;
- интерфейс для отображения и изменения всех свойств фигуры.

Требования

Интерфейс и классы реализуются в одном модуле. Для редактирования фигуры разработать функцию *ModifyFigure*(), которая должна получать ссылку

на экземпляр фигуры и предоставлять интерактивный консольный интерфейс для работы с ним.

Варианты заданий

1 Фигура: сектор окружности.
2 Фигура: треугольник Рело.
3 Фигура: правильный шестиугольник.
4 Фигура: эллипс.
5 Фигура: параллелограмм.
6 Фигура: сегмент окружности.
7 Фигура: ромб.
8 Фигура: кольцо (бублик).
9 Фигура: правильная трапеция.
10 Фигура: дельтоид.

Используйте ru.wikipedia.org для получения информации о фигурах.

5 Контрольные вопросы для самопроверки

1 Что такое парадигма программирования?

2 В чем заключается парадигма объектно-ориентированного программирования? Расскажите подробнее о ней и еѐ преимуществах.

3 Какие базовые концепции положены в основу объектно-ориентированного программирования? В чѐм они заключаются?

4 Что такое инкапсуляция? В чѐм она заключается и как используется при работе с классами?

5 Что такое наследование? В чѐм оно заключается и как используется при работе с классами?

6 Что такое полиморфизм? В чѐм он заключается и как используется при работе с классами?

7 Что такое абстракция? В чѐм он заключается и как используется при работе с классами?

8 Что такое класс и как он объявляется в программе?

9 Что такое свойство и метод класса? Как они объявляются в программе? Как создаются статические и динамические экземпляры и их массивы? Приведите примеры.

10 Что такое оператор? Какие операторы Вы знаете?

11 Что такое перегрузка операторов при работе с классами? Как перегрузить операторы? Приведите примеры.

12 Какие имеются уровни доступа к свойствам и методам класса? В чѐм они заключаются?

13 Как обратиться к свойству или методу класса по ссылке и по указателю? Что такое указатель на себя? Приведите примеры.

14 Что такое конструктор и деструктор класса? Как они объявляются и

для чего используются в программе?

15 Что такое конструктор копирования и конструктор по умолчанию? Приведите примеры.

16 В чем заключается порождение одного класса от другого класса? Что такое иерархия классов и как она представляется графически? Порядок вызова конструкторов и деструкторов классов при наследовании. Приведите примеры построения иерархии.

17 Что такое множественное наследование? Как породить класс от нескольких классов? Приведите примеры множественного наследования.

18 Как вызываются конструкторы и деструкторы при множественном наследовании?

19 В чем заключаются механизмы раннего и позднего связывания при вызове методов класса? Приведите примеры.

20 Что такое виртуальный метод класса и как его объявить? В чем преимущество виртуальных методов? Приведите примеры.

21 Что такое константные свойства, методы, аргументы? В чем их особенности? Как описываются константные свойства и методы в тексте программы? Сравнение вызова константного и обычного метода. Приведите примеры.

22 Что такое статические свойства, методы, классы? В чем их особенности? Какова концепция статических свойств и методов, особенности их описания и использования? Контекст вызова статического метода. Приведите примеры.

23 Что такое дружественные функции? Что такое дружественные классы? Как реализовать доступ к личным и защищенным элементам из внешней функции или класса? Приведите примеры.

24 Как объявляется шаблон класса со свойствами и методами? Ссылки и указатели на экземпляры класса. Контекст обращения к элементам класса по ссылке, по указателю, по области действия. Одноименные элементы. Приведите примеры.

25 Что такое контекст вызова метода? В чем различие механизмов вызова обычного, статического и виртуального методов? Как используется указатель на себя? Приведите примеры.

26 Что такое структура и объединение? Как они описываются и используются в программе? Как определяются поля битов? Обращение к элементам структур и объединений. Приведите примеры.

Список литературы

1 **Павловская, Т. А.** C/C++. Программирование на языке высокого уровня: учебник для вузов / Т. А. Павловская. – Санкт-Петербург : Питер, 2016. – 461 с.

2 **Павловская, Т. А.** C/C++. Структурное и объектно-ориентированное программирование: Практикум / Т. А. Павловская, Ю. А. Щупак. – Санкт-Петербург : Питер, 2011. – 352 с.: ил.

3 **Васильев, А. Н.** Программирование на C++ в примерах и задачах / А. Н. Васильев. – Москва : Эксмо, 2016. – 368 с.

4 **Шилдт, Г.** C++. Базовый курс / Г. Шилдт. – Москва: Вильямс, 2015. – 624 с.

5 **Полубенцева, М.** C/C++. Процедурное программирование / М. Полубенцева. – Санкт-Петербург : BHV, 2017. – 432 с.

6 **Тепляков, С.** Паттерны проектирования на платформе .NET / С. Тепляков. – Санкт-Петербург : Питер, 2015. – 320 с.