МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

(СибГУТИ)

ХАБАРОВСКИЙ ИНСТИТУТ ИНФОКОММУНИКАЦИЙ (ФИЛИАЛ)

(ХИИК СибГУТИ)

СРЕДНЕЕ ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАНИЕ

**Методические указания к лабораторным работам по дисциплине**

**МДК 01.01 Разработка программных модулей**

**2 семестр**

СОДЕРЖАНИЕ

|  |  |
| --- | --- |
| Лабораторная работа №1 |  |
| Лабораторная работа №2 |  |
| Лабораторная работа №3 |  |
| Лабораторная работа №4 |  |
| Лабораторная работа №5 |  |
| Лабораторная работа №6 |  |
| Лабораторная работа №7, 8 |  |
| Лабораторная работа №9 |  |
| Лабораторная работа №10, 11 |  |
| Лабораторная работа №12, 13 |  |
| Лабораторная работа №14, 15 |  |
| Лабораторная работа №16 |  |
| Лабораторная работа №17 |  |
| Лабораторная работа №18 |  |
| Лабораторная работа №19 |  |
| Лабораторная работа №20 |  |
| Лабораторная работа №21 |  |
| Лабораторная работа №22, 23 |  |

Лабораторная работа №1

Тема: " Разработка приложения с использованием текстовых компонентов"

Цель: закрепить навыки создания приложений с использованием текстовых компонентов.

**Задание**: Разработать приложение на форме с использованием текстовых компонентов поясняющих вводимые значения и результаты вычислений. (Предусмотреть изменение текстовых компонентов - шрифт, цвет, жирное, курсив).

**Варианты индивидуального задания**

Часть 1

1. Вводится число в 10-ричной системе счисления. Вывести значение этого числа в 2-ичной, 8-ричной, 16-ричной системах счисления (таблица 1).

Таблица 1 – Задание на перевод десятичных чисел по вариантам

|  |  |  |  |
| --- | --- | --- | --- |
| Вариант | Число 1 | Число 2 | Число 3 |
| 1 | 25 | 111 | 3 |
| 2 | 50 | 1111 | 7 |
| 3 | 75 | 222 | 15 |
| 4 | 100 | 2222 | 31 |
| 5 | 125 | 333 | 63 |
| 6 | 150 | 3333 | 127 |
| 7 | 175 | 444 | 255 |
| 8 | 200 | 4444 | 511 |
| 9 | 225 | 555 | 1023 |
| 10 | 250 | 5555 | 2045 |
| 11 | 275 | 666 | 1 |
| 12 | 200 | 6666 | 5 |
| 13 | 325 | 777 | 9 |
| 14 | 350 | 7777 | 17 |
| 15 | 375 | 888 | 33 |
| 16 | 400 | 8888 | 133 |
| 17 | 425 | 999 | 257 |
| 18 | 450 | 9999 | 513 |
| 19 | 475 | 1010 | 1025 |
| 20 | 500 | 10101 | 2047 |
| 21 | 525 | 1000 | 4096 |

2. Вводится число в двоичной системе счисления. Вывести 8-ричные, десятичные и 16-ричные эквиваленты (таблица 2).

Таблица 2 – Задание на перевод двоичных чисел по вариантам

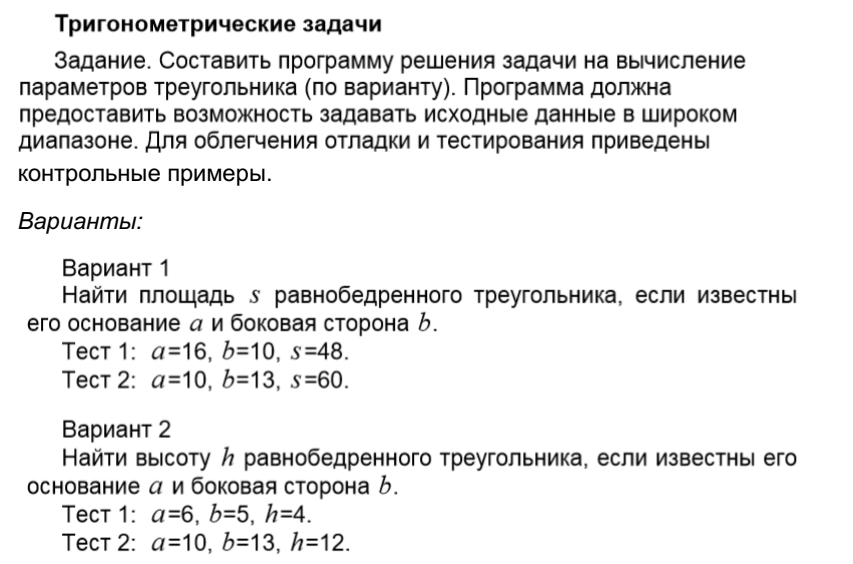
|  |  |  |  |
| --- | --- | --- | --- |
| Вариант | Число 1 | Число 2 | Число 3 |
| 1 | 1000 | -11110 | 11110000 |
| 2 | 10001 | -10111 | 11110001 |
| 3 | 110011 | -110111 | 11110010 |
| 4 | 1100011 | -111011 | 11110011 |
| 5 | 10101 | -111101 | 11110100 |
| 6 | 1001001 | -111100 | 11110101 |
| 7 | 11100 | -111011 | 111101110 |
| 8 | 111001 | -111010 | 101010 |
| 9 | 110001 | -111001 | 101011 |
| 10 | 1010 | -1010001 | 111010 |
| 11 | 101011 | -1010000 | 111011 |
| 12 | 101110 | -1010100 | 110010 |
| 13 | 100100 | -100011 | 11001110 |
| 14 | 100111 | -11100111 | 1100011 |
| 15 | 100101 | -11101000 | 11000011 |
| 16 | 100011 | -1110010 | 110000011 |
| 17 | 100010 | -101011 | 10000 |
| 18 | 10101010 | -101010 | 100001 |
| 19 | 11011 | -101111 | 1000010 |
| 20 | 101110 | -101110 | 110010011 |
| 21 | 11111 | -111111 | 101010 |

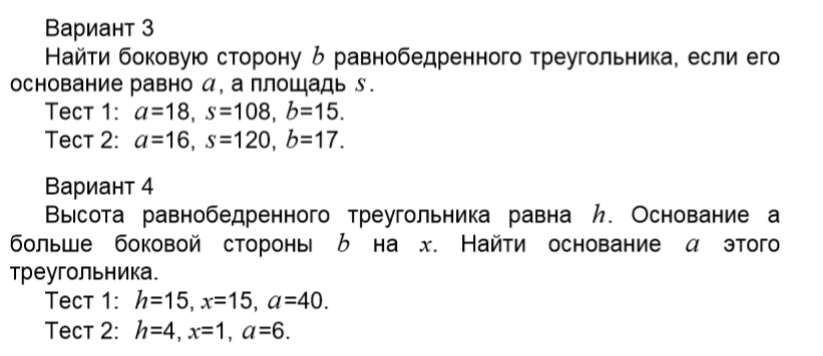
3. Вводятся числа a и b в 8-ичной системе счисления. Найти их сумму (a+b) и разность (a-b) в десятичной системе счисления (таблица 3).

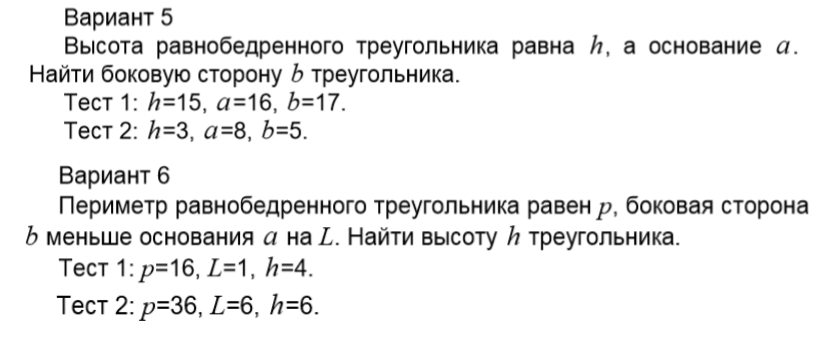
Таблица 3 – Задание на подсчет суммы и разности чисел по вариантам

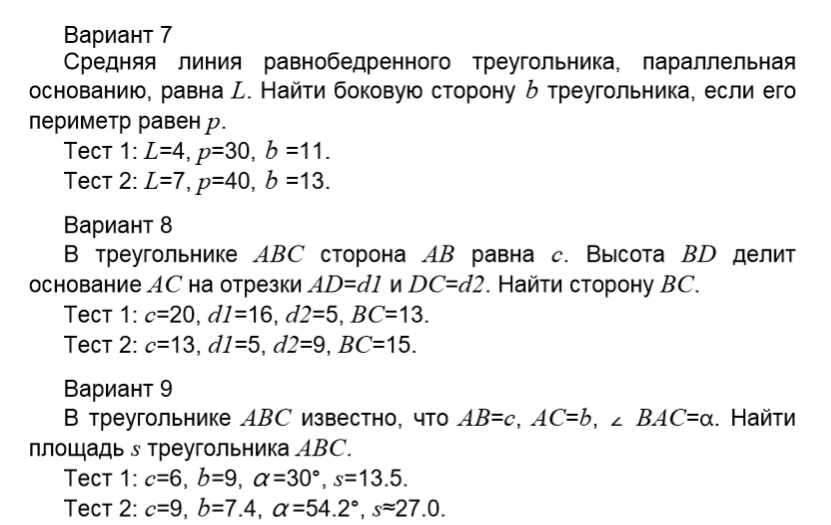
|  |  |  |
| --- | --- | --- |
| Вариант | Число a | Число b |
| 1 | 455 | 510 |
| 2 | 5156 | 47 |
| 3 | 4545 | 14 |
| 4 | 215 | 156 |
| 5 | 5215 | 55 |
| 6 | 215 | 30 |
| 7 | 151 | 20 |
| 8 | 215 | 40 |
| 9 | 50 | 50 |
| 10 | 5156 | 60 |
| 11 | 500 | 70 |
| 12 | 751 | 51 |
| 13 | 546 | 52 |
| 14 | 320 | 53 |
| 15 | 3153 | 54 |
| 16 | 3156 | 57 |
| 17 | 517 | 65 |
| 18 | 732 | 5 |
| 19 | 237 | 66 |
| 20 | 210 | 77 |
| 21 | 20 | 33 |

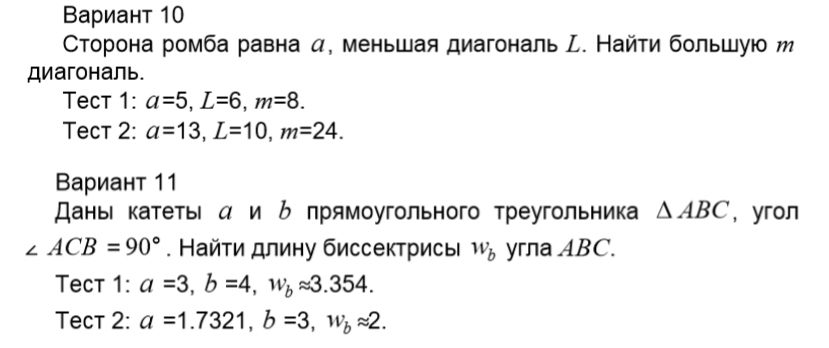
Часть 2

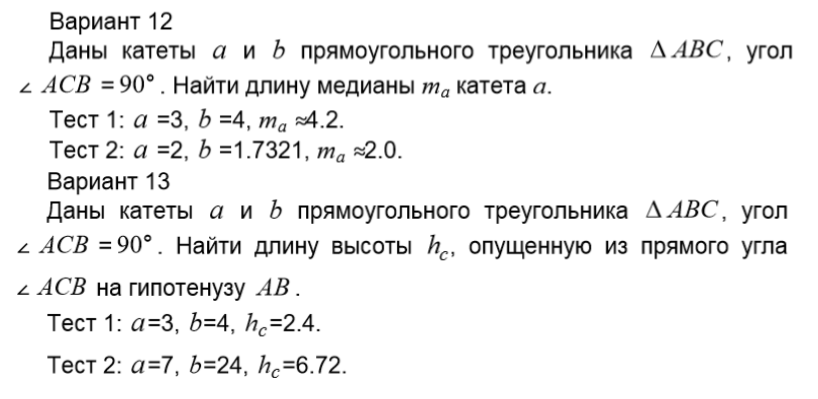


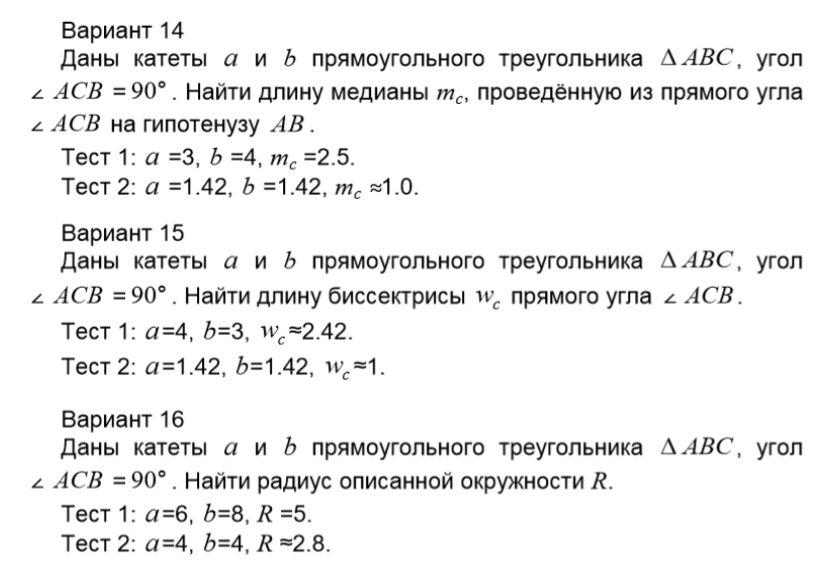


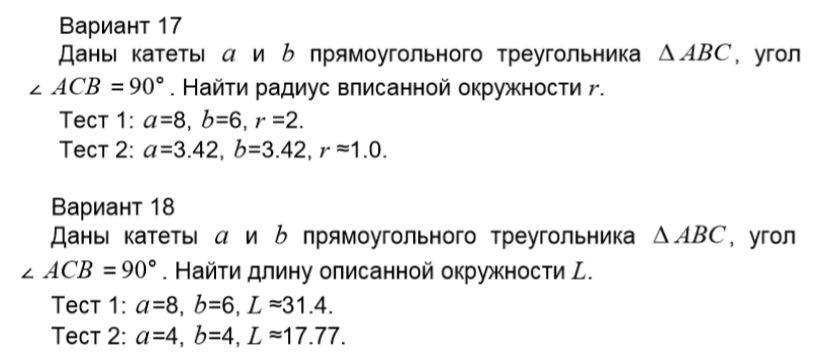


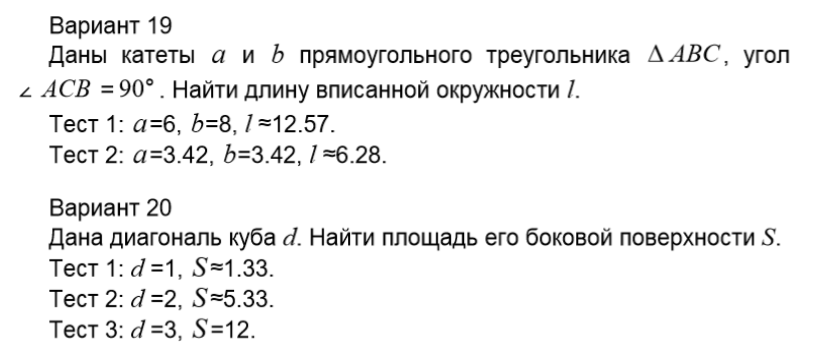


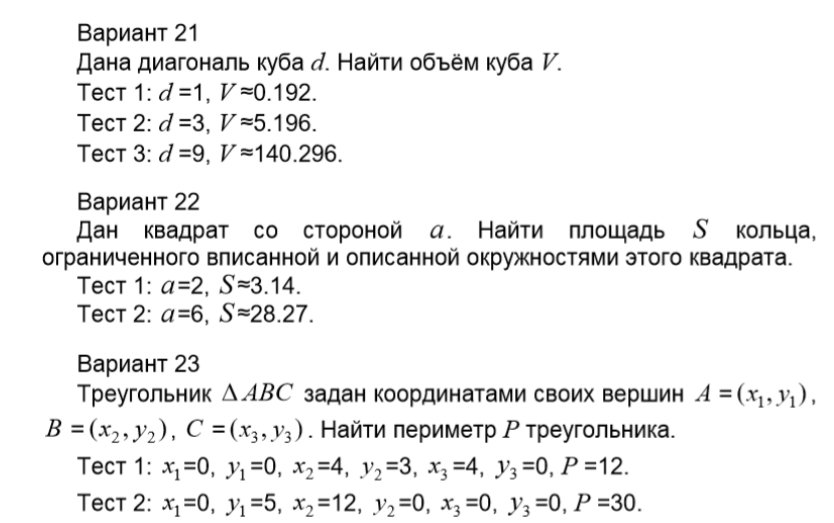


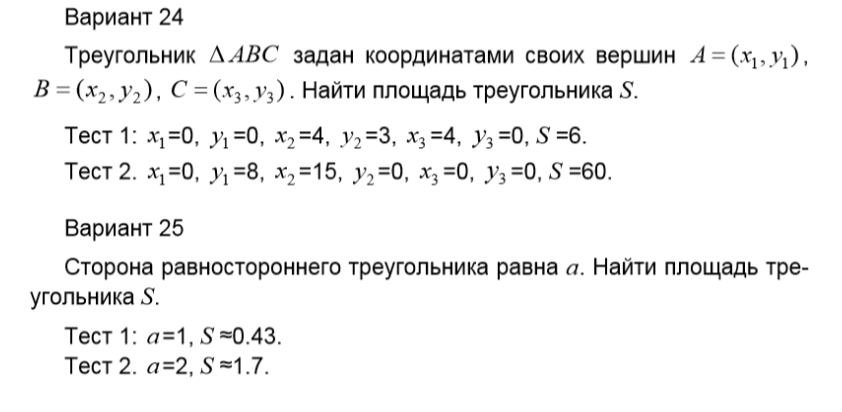




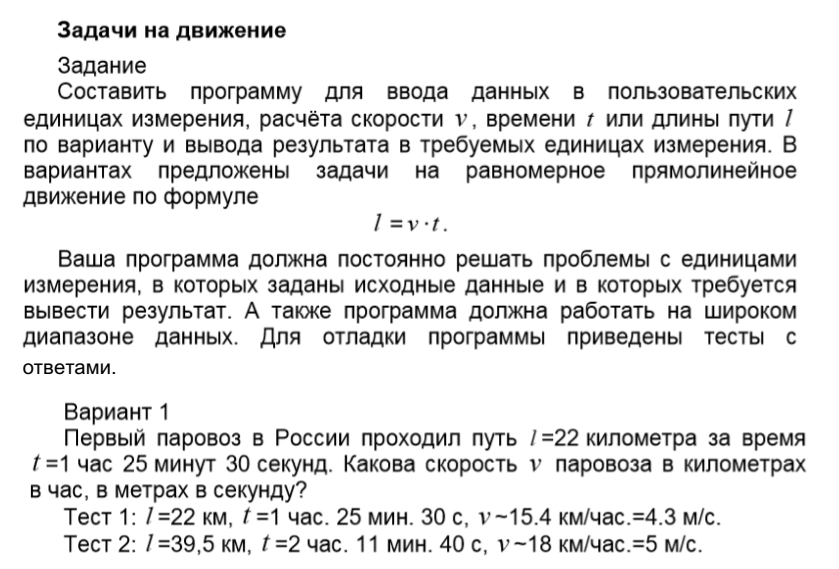




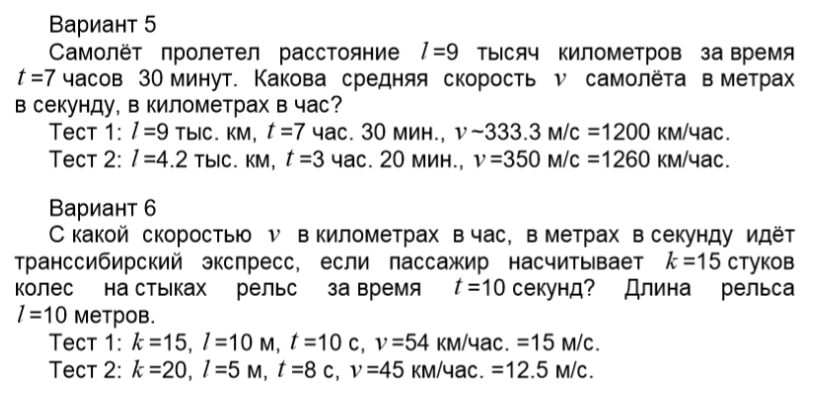


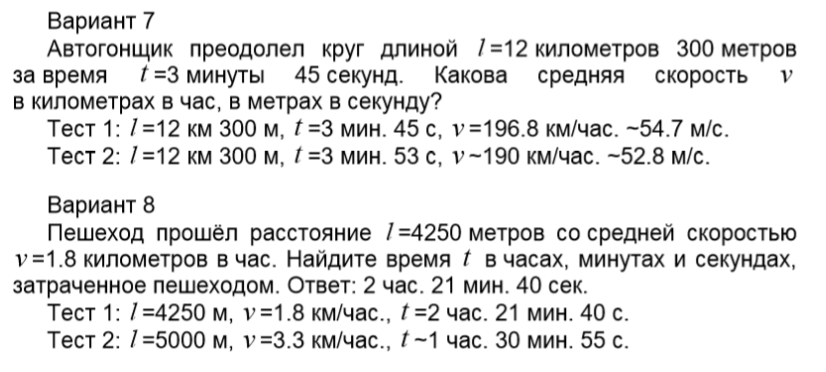


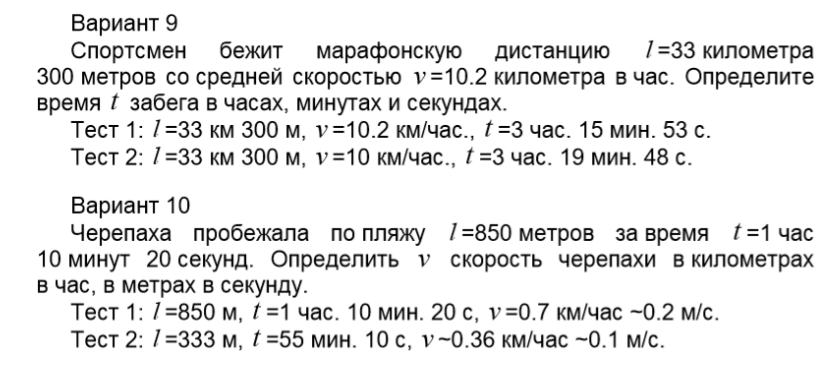
Часть 3

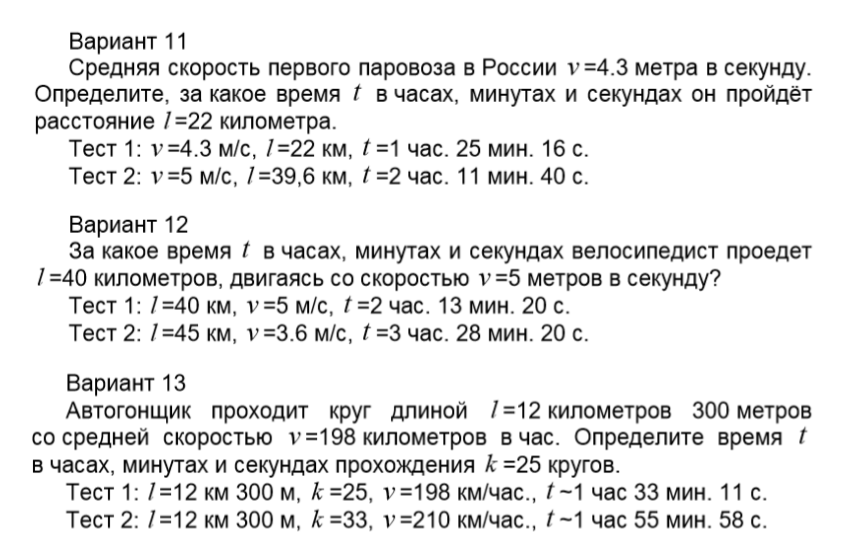


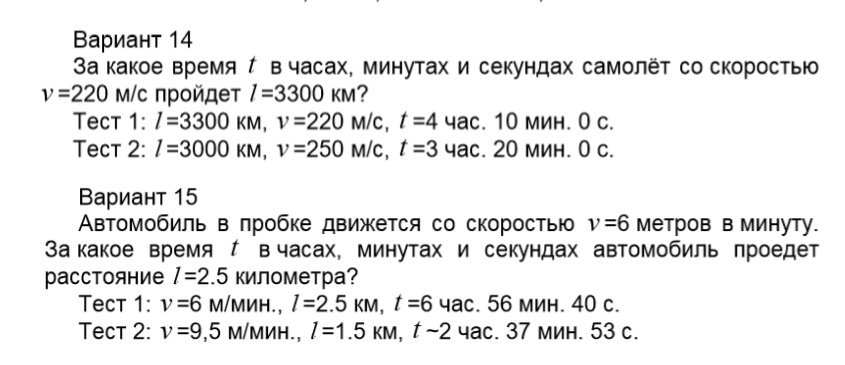


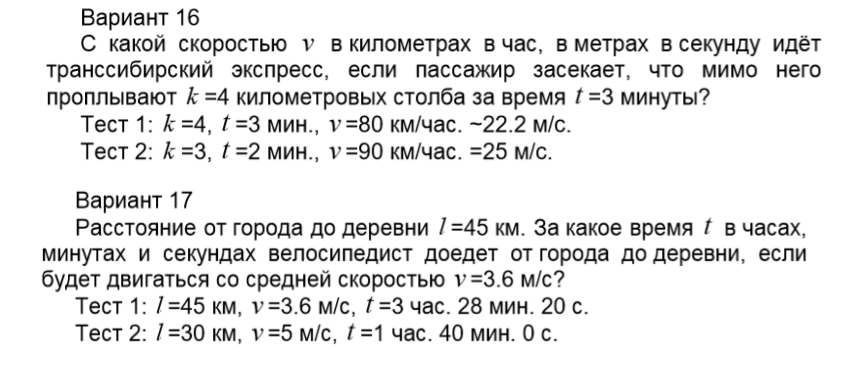


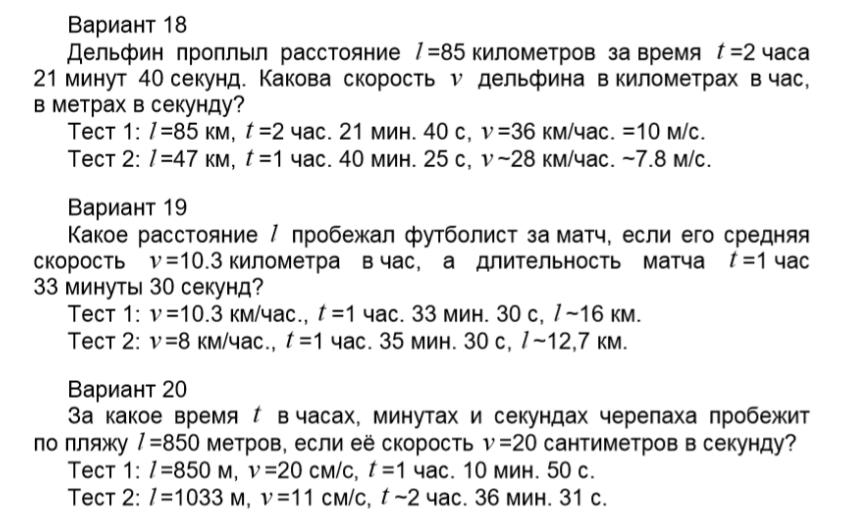


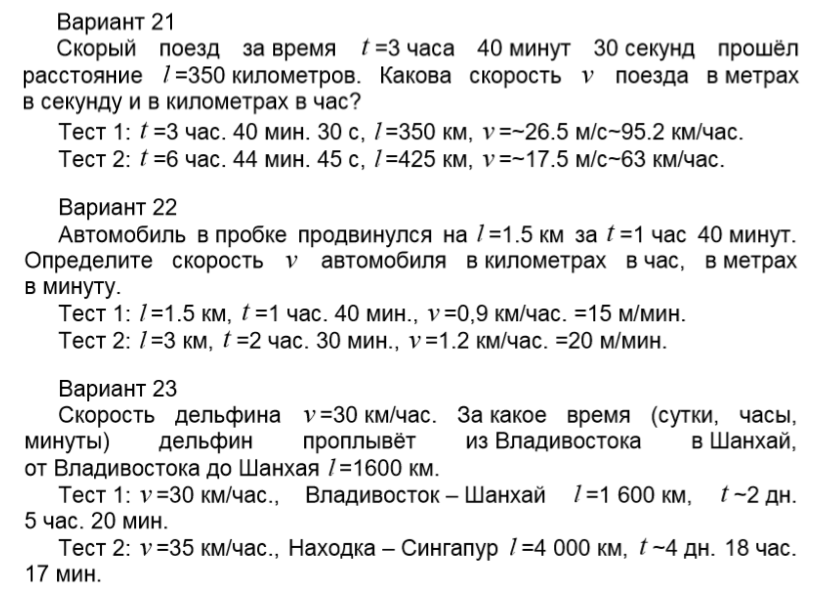


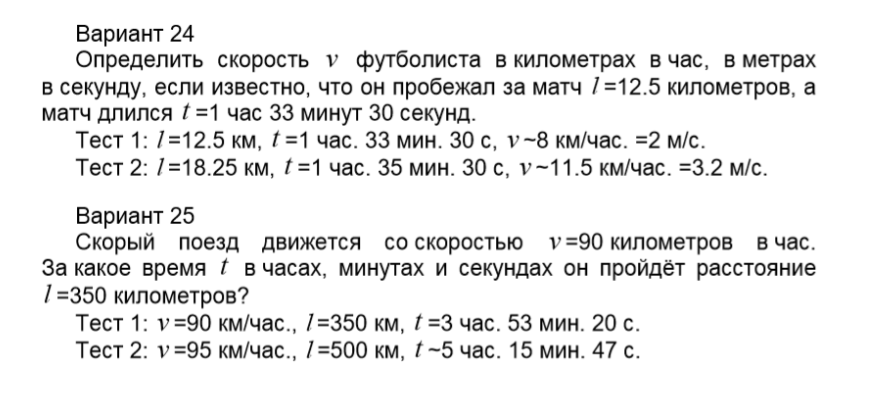












Лабораторная работа №2

Тема: Разработка приложения с несколькими формами

Цель: Закрепить навыки создания приложения с несколькими формами.

Задание

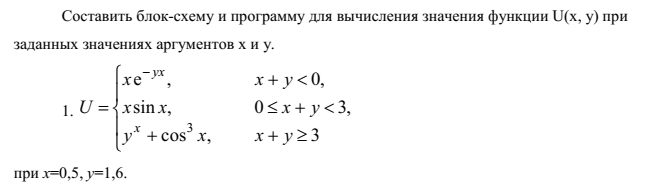
1. С помощью ComboBox выполнить выбор первого или второго задания. Выбранное задание открывается в новом окне.

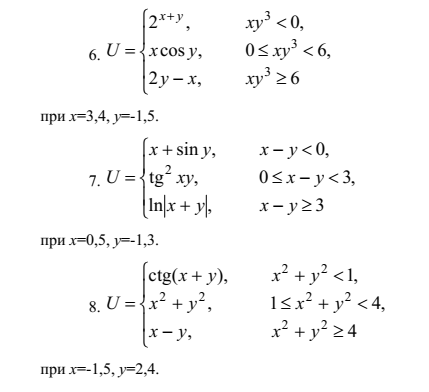
2. Задание 1. Создать программу на форме с выводом результата вычисления в новую форму, предусмотреть ввод исходных данных через TextBox, обязательно наличие объекта Label с пояснениями. Вывести на форму фото вашей формулы.

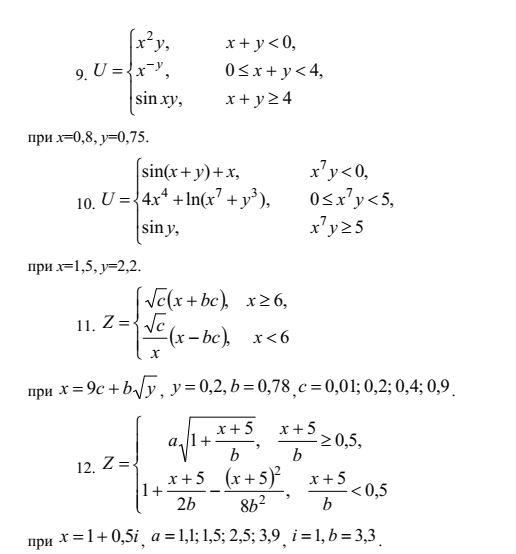
3. Задание 2. Создать программу на форме с выводом результата вычисления в новой форме, предусмотреть ввод исходных данных через TextBox, обязательно наличие объекта Label с пояснениями. С помощью RadioButton организовать выбор значения f(x) из списка предложенных.

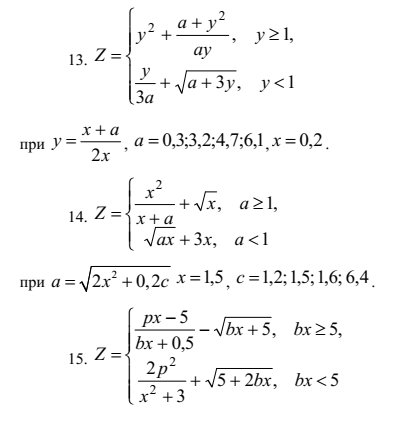
Индивидуальные задания:

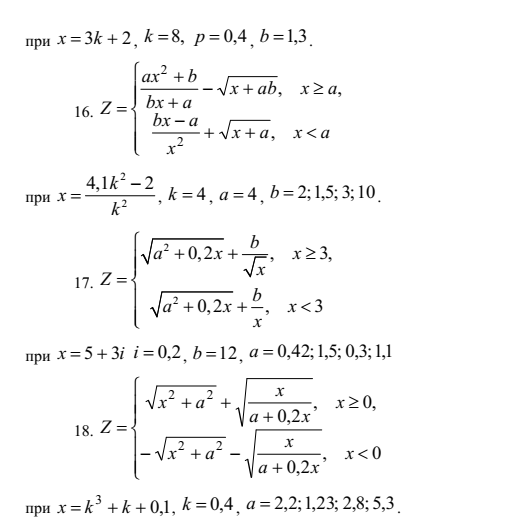
Задание 1

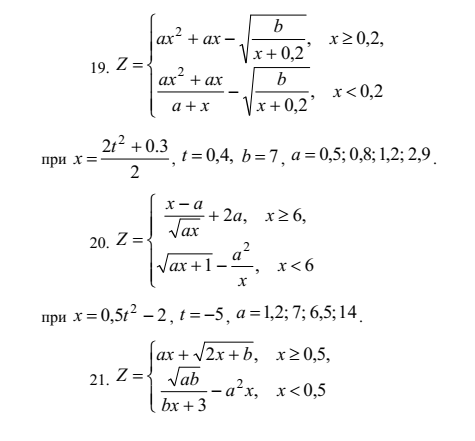


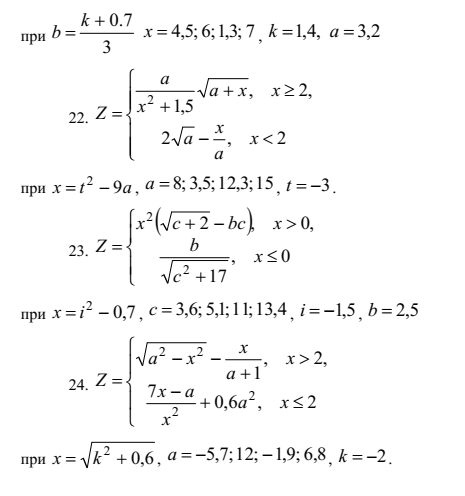
 

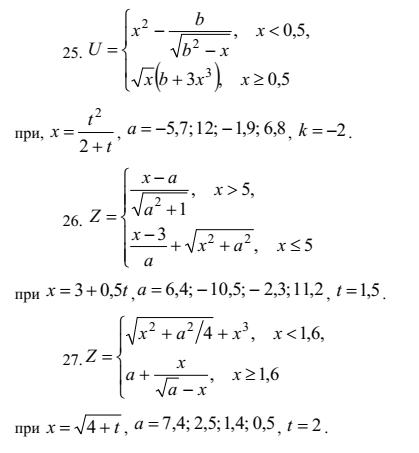


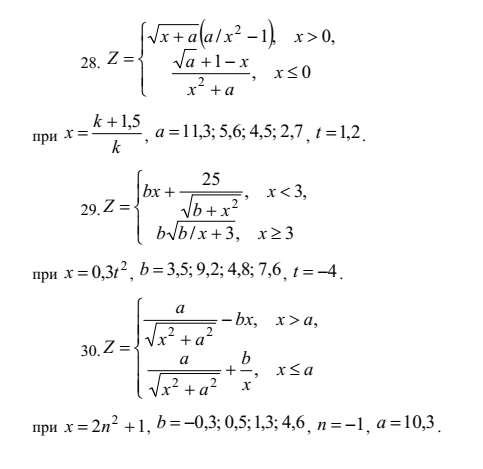












Задание 2

По указанию преподавателя выберите индивидуальное задание из нижеприведенного списка. В качестве *f(x)* использовать по выбору: *sh(x)*, *x2*, *ex*.

Отредактируйте вид формы и текст программы, в соответствии с полученным заданием.

|  |  |
| --- | --- |
| 1. | 2. |
| 3. | 4. |
| 5. | 6. |
| 7. | 8. |
| 9. | 10. |
| 11. | 12. |
| 13. | 14. |
| 15. | 16. |
| 17. | 18. |
| 19. | 20. |

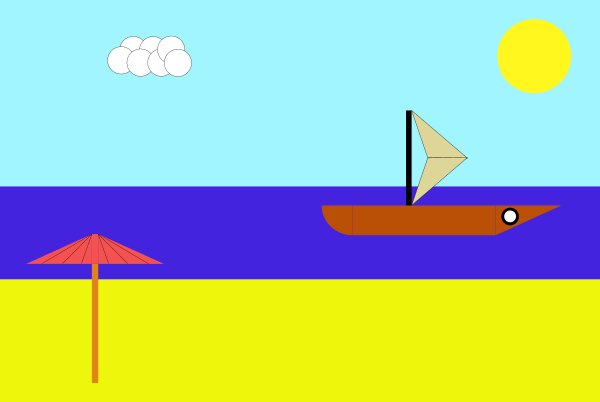
Лабораторная работа №3

Тема: «Разработка приложения с визуальными компонентами»

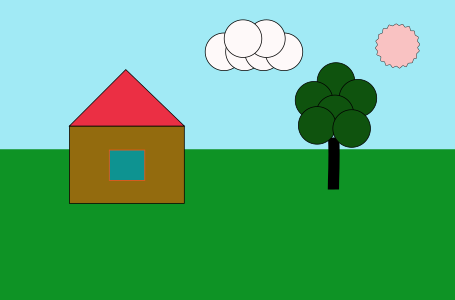
Цель: на практике изучить применение библиотеки Canvas.

Задание

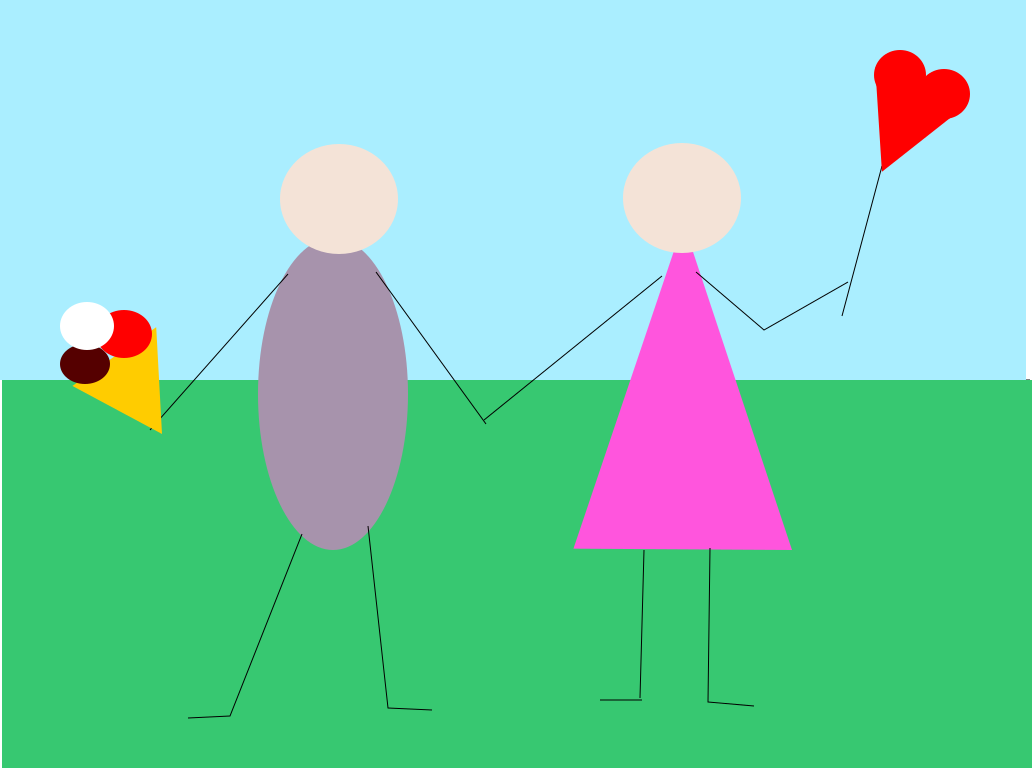
Вариант 1



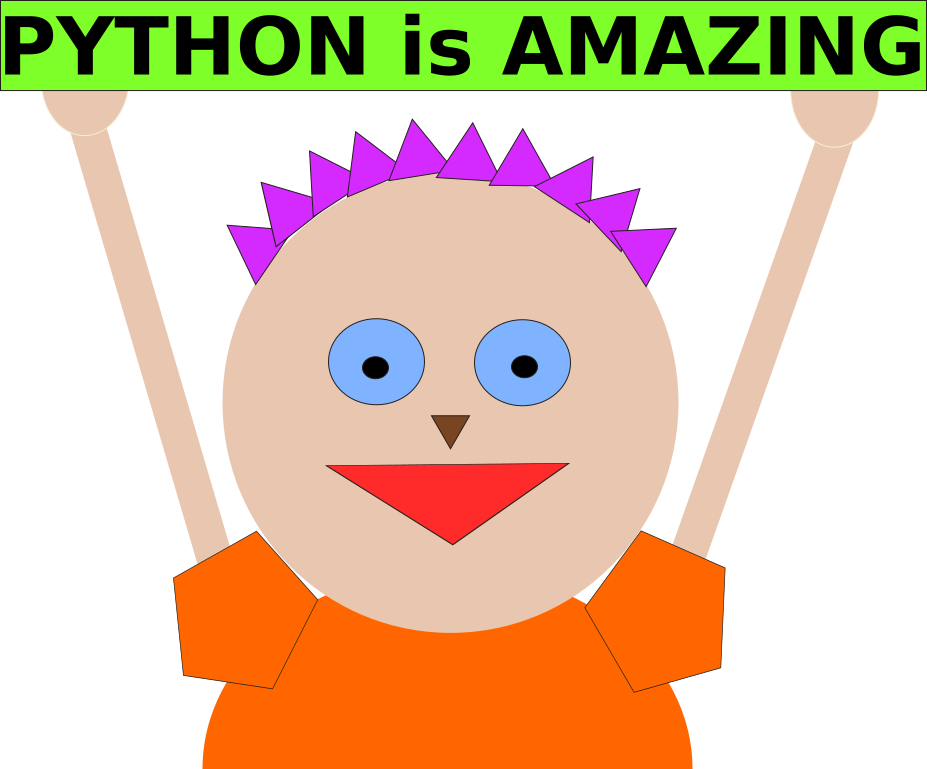
Вариант 2



Вариант 3



Вариант 4



Вариант 5



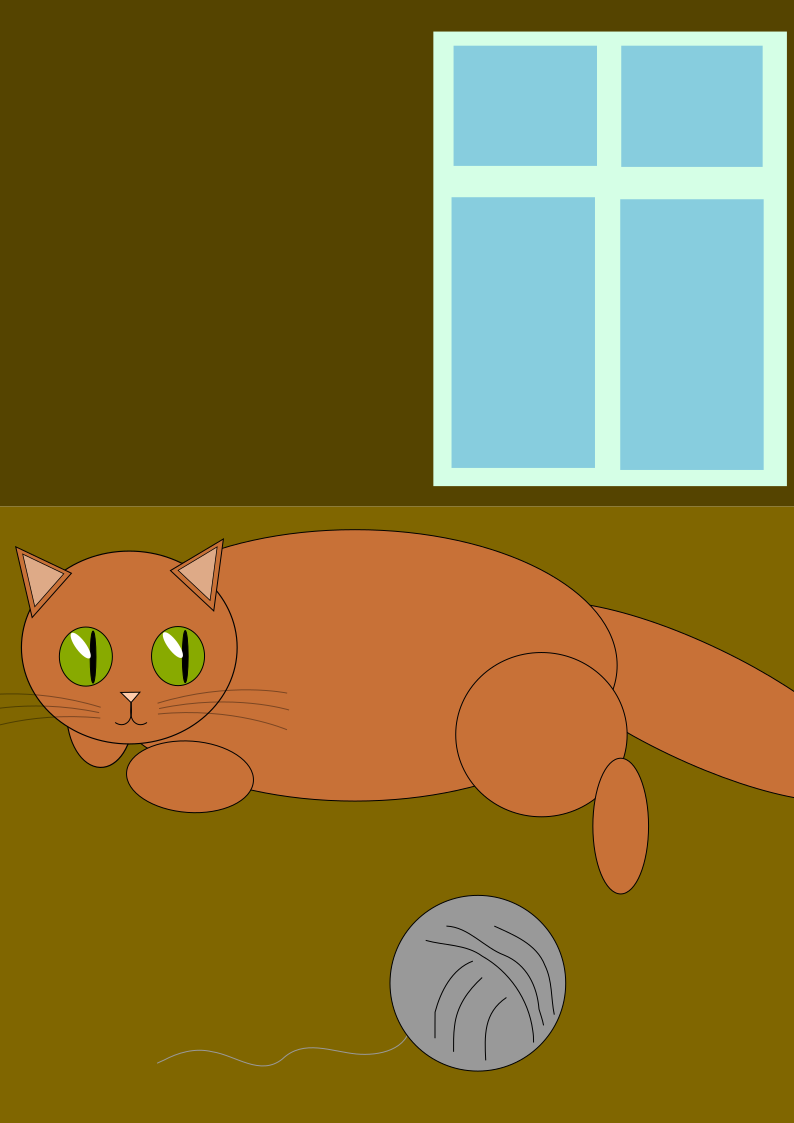
Вариант 6



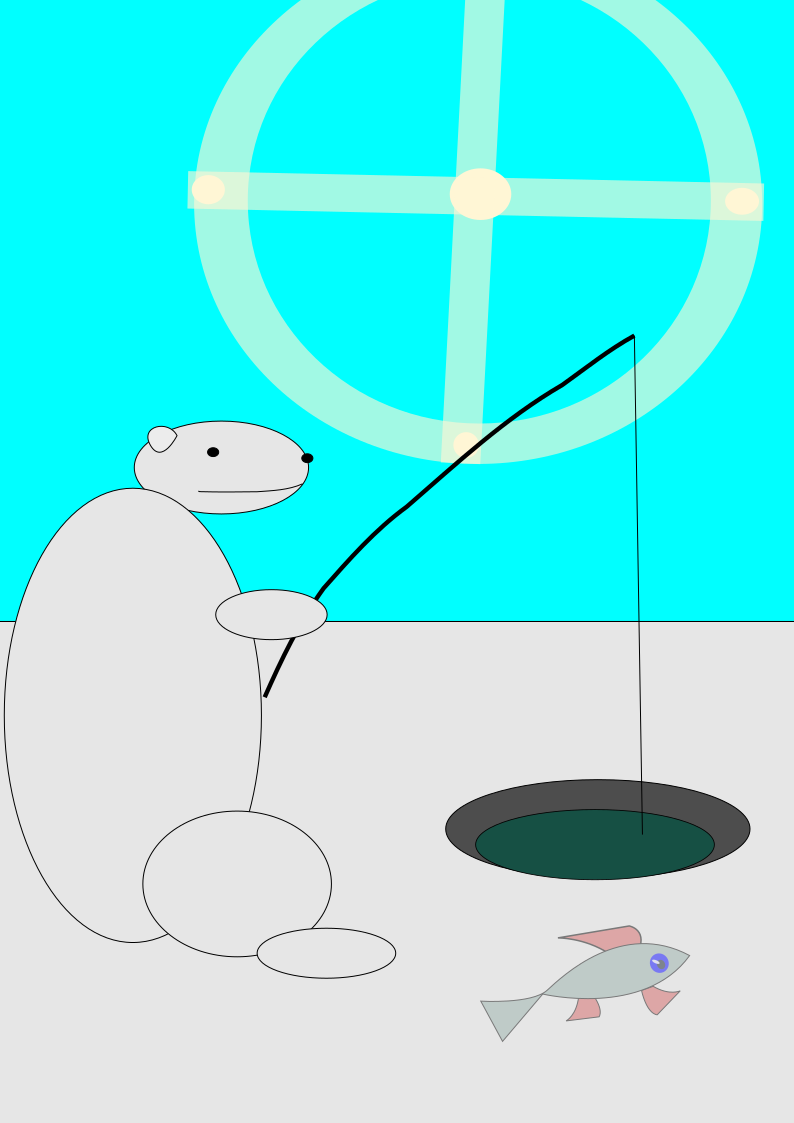
Вариант 7



Вариант 8



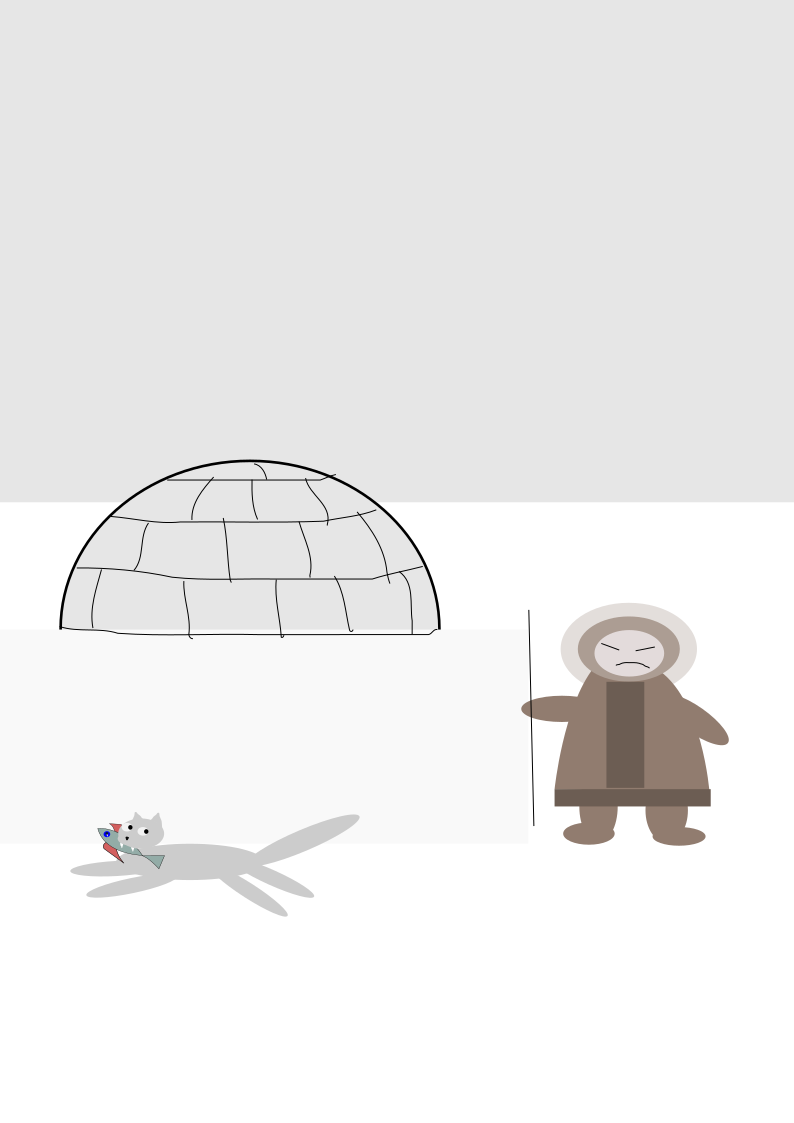
Вариант 9



Вариант 10



Вариант 11



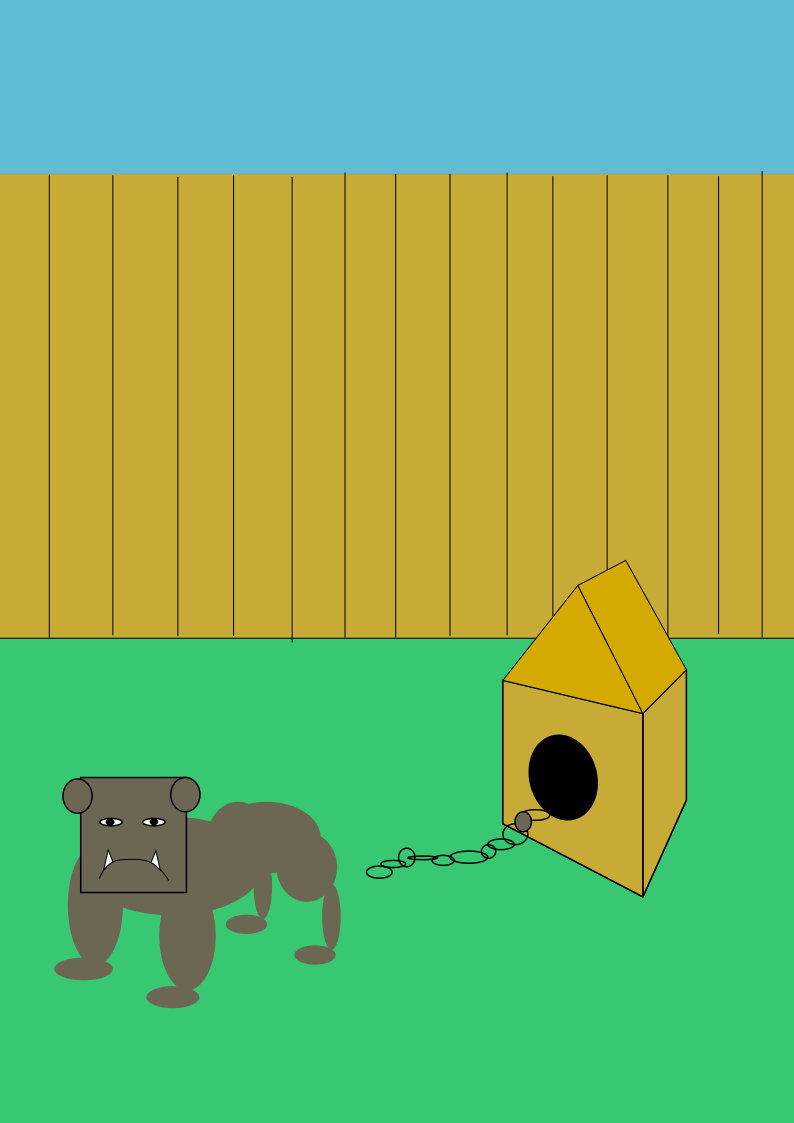
Вариант 12



Вариант 13



Вариант 14



Вариант 15



Групповая работа № 4

Тема: " Разработка модуля многооконного приложения "

Цель работы: Приобретение практических навыков разработки и отладки программ, реализующих многооконные приложения

Задание. Написать программу на форме, с помощью объекта RadioBatton должен выбираться номер задания. Задание открывается в новом окне.

Варианты для выполнения

Вариант 1

1. Составить программу, которая будет по паролю определять степень допуска сотрудника к секретной информации в базе данных. Доступ имеют 6 человек, которые разбиты на три группы по степени доступа. Они имеют следующие пароли:

9583, 1747 - доступны модули базы А, В, С;

3331, 7922 - доступны модули базы А, В;

9455, 8997 - доступен модуль базы С

1. Программа выполняет определенные действия в зависимости от значения второй цифры во введенном трехзначном числе:

* если вторая цифра числа принимает значения от 0 до 5, то определяется ее квадрат;
* если вторая цифра числа принимает значения 6 и 7, то определяется сумма крайних цифр введенного числа;
* если вторая цифра числа равна 8 и 9, то определяется сумма всех цифр введенного числа.

1. Составить программу, которая вычисляет дату следующего дня. Если следующий день является праздничным днём, (а именно, 1 января, 7 января, 23 февраля, 8 марта), то выдать поздравление с конкретным праздником. Если введённый день является последним днём месяца, то выдать также «Последний день месяца». Например, последовательность выдачи сообщений для введённого числа 31122014 будет такой «Последний день месяца! С наступающим новым годом! Завтра 01.01.2015»

Вариант 2

1. Составить программу, которая анализирует возраст человека и относит его к одной из возрастных групп: дошкольник, ученик, работник, пенсионер. Возраст вводится с клавиатуры
2. Составить программу, которая в зависимости от старшей цифры введен­ного четырехзначного числа выполняет следующие действия:

* при цифрах 1, 5, 9 - находит произведение крайних цифр введенного числа;
* при цифрах 2, 7, 0 - находит произведение средних цифр введенного числа;
* при цифрах 3,6 - находит результат целочисленного деления крайних цифр введенного числа;
* в остальных случаях - выводит нуль.

1. Составить программу, которая для введённого К (числа грибов), будет выдавать фразу «Мы нашли в лесу К грибов», причём согласовывать окончание слова «гриб» с числом К. Количество грибов может быть любым целым числом

Вариант 3

1. Составить программу, которая определяет номер квартала по введённо­му номеру месяца
2. Составить программу, выполняющую в зависимости от значения остатка от деления введенного пользователем четырехзначного числа на 5, следующие действия:

* остаток равен 1 - умножение введенного числа на 10;
* остаток равен 2 - целочисленное деление введенного числа на 3;
* остаток равен 3 - сложение цифр введенного числа;
* остаток равен 4 - вычитание цифр введенного числа

1. Составить программу, которая по введённому году рождения рассчитает возраст и выдаст фразу «Вам К лет», где К - рассчитанный возраст, при этом в нужных случаях слово «лет» заменить на слово «год» или «года». Напри­мер, «Вам 70 лет», «Вам 23 года» и так далее.

Вариант 4

1. Решить выражение в зависимости от введённого номера выражения:

1)5А+ЗА+1;

2)4А+ЗА-8;

3)8А+5А+10.

Значение переменной А вводится с клавиатуры

1. Программа просит ввести средний балл успеваемости (целое число). В зависимости от введённого значения вывести сообщение «отлично», «хорошо», «удовлетворительно», «плохо, ты отчислен»
2. Составить программу, которая сообщает сумму сдачи покупателю, при этом согласовывает окончание слова «рубль». Стоимость покупки и внесённая покупателем сумма денег вводится с клавиатуры

Вариант 5

1. Программа ведёт с пользователем диалог (Y - да, N - нет):

- У тебя есть дома компьютер?

Если ответ «Да», то выдать сообщение «хорошо, я очень рад». В случае отрицательного ответа выдать сообщение «Купи, не жалей денег». Весь диа­лог выводится на экран компьютера

1. Программа просит ввести два числа и один из знаков: +, -, \*, /, а затем выводит результат соответствующего арифметического действия
2. Составить программу, которая после ввода числа (диапазон от 1 до 99), обозначающего денежную единицу, правильно дописывает «копейка»

Вариант 6

1. Программа просит ввести число от 1 до 5, а затем выводит прописью введённое число
2. Вводится целое число. Если введённое число попадает в диапазон от 1 до 9, то определить чётное число или нечётное. Если введённое число попадает в диапазон от 10 до 99, то выдать сообщение «Число является двухзначным», если в диапазон от 100 [до 999, то выдать сообщение «Число является трёхзначным»
3. Составить программу, которая после ввода числа (диапазон от 1 до 99), обозначающего вес изделия, дописывает «килограмм» в правильной форме

Вариант 7

1. Программа запрашивает сезон дня рождения пользователя (Z - зима, V - весна, L - лето, О - осень). В зависимости от сезона выдать:

Z - Зима! Крестьянин торжествуя...

V - И даже пень весной берёзой стать мечтает

L - Ох, лето знойное! Любил бы я тебя...

О - Унылая пора! Очей очарованье...

1. Составить программу, которая в зависимости от введенного числа вы­полняет определённые арифметические действия:

* если число в диапазоне от 1 до 100, то число увеличить в 100 раз;
* если число в диапазоне от 101 до 500, то число возвести в квадрат;
* если число в диапазоне от 501 до 5000, то вывести целочисленный остаток от деления числа на 15;
* в.остальных случаях выдать сообщение «с таким числом работать не хочу и не буду»

1. Составить программу, которая по введенному номеру месяца и году выводит название месяца и количество дней в месяце. Например, вводится месяц 02 и год 2000, должно быть выведено «В феврале месяце 2000 года 29 дней». Високосным является, например, 2000 год.

Вариант 8

1. Программа запрашивает номер месяца вашего рождения и выводит сообщение «Вы родились летом (зимой, весной, осенью)»
2. Программа вычисляет выражение 5А, где число А вводится. В зависимости от результата выводится сообщение:

* если результат от 0 до 10, вывести сообщение «очень мало»;
* если результат от 11 до 50, вывести сообщение «всё равно мало»;
* если результат от 51 до 80, вывести сообщение «уже лучше»;
* если результат от 81 до 100, вывести сообщение «почти хорошо»;
* в остальных случаях вывести сообщение «в самый раз»

1. Составить программу, которая для введённого В (числа страниц), будет выдавать фразу «В книге пять страниц», причём согласовывать окончание слова «страница» с числом К. Количество страниц может быть любым целым числом

Вариант 9

1. Составить программу, которая запрашивает у пользователя номер дня недели и в зависимости от введённого значения выводит сообщение, в кото­ром указано название дня недели и добавлено «рабочий день» или «выходной день»
2. Составить программу, вычисляющую стоимость междугородного телефонного разговора. Исходными данными являются код города и продолжительность разговора в минутах:

Благовещенск - код города 416, стоимость одной минуты 4.80 руб.;

Владивосток - код города 423, стоимость одной минуты 5.50 руб.;

Мурманск - код города 815, стоимость одной минуты 25.50 руб.;

Амурск - код города 242, стоимость одной минуты 4.60 руб.

В сообщении пользователю выдать название города, цену одной минуты и стоимость разговора

1. Составить программу, которая для введённого А (числа цветков), будет выдавать фразу «У меня А цветков», причём согласовывать окончание слова «цветок» с числом А. Количество цветков может быть любым целым числом

Вариант 10

1. Программа вычисляет выражение по введённому номеру выражения и заданной переменной:

1)2Х+5; 2) 8Х +10; 3)15Х-12

1. Программа даёт оценку вашего веса. Исходные данные: вес и рост (в сантиметрах). Рассчитать значение «идеального веса», которое равно «рост минус 100». В зависимости от разницы между введённым весом и идеальным весом выдать сообщение:

* «ваш вес в норме» (разница имеет значение от 0 до 1);
* «обратите на себя внимание» (разница имеет значение от 2 до 5);
* «срочно физкультура и диета» (разница имеет значение от 5 до 10);
* «мне вас очень жаль» (разница имеет значение от 11 до 50);
* «вам нужны лишние калории» (в остальных случаях)

1. Составить программу, которая для введённого Z (числа тюльпанов), будет выдавать фразу «Мне подарили Z тюльпанов», причём согласовывать окончание слова «тюльпан» с числом Z. Количество тюльпанов может быть любым целым числом

Вариант 11

1. Программа преобразовывает введенное целое число в зависимости от ве­личины остатка его деления на 17 следующим образом:

* если остаток равен 0, то введенное число заменяется на 0;
* если остаток равен 1, то введенное число меняет знак;
* если остаток равен 2, 3 или 5, то веденное число удваивается;
* если остаток равен 4, то введенное число увеличивается в три раза;
* в остальных случаях число увеличивается в пять раз

1. Программа запрашивает у студента количество долгов по окончании сес­сии и выводит оценку результата его учебной деятельности:

* «молодец!» (студент не имеет долгов);
* «чуть-чуть не дотянул» (студент имеет один долг);
* «все плохо» (студент имеет два долга):
* «дальше учиться не стоит, пиши заявление» (студент имеет три долга);
* «пора в армию» (студент имеет более трех долгов)

1. Составить программу, которая для введённого X (числа роз), будет вы­давать фразу «У меня выросло X роз», причём согласовывать окончание слова «роза» с числом X. Количество роз может быть любым целым числом

Вариант 12

1. Программа запрашивает температуру дня и в зависимости от введённого значения выдаёт сообщения: «очень холодно», «холодно», «тепло», «жарко». Интервалы температур задать самостоятельно
2. Составить программу, вычисляющую стоимость междугородного телефонного разговора. Исходными данными являются стоимость одной минуты разговора и продолжительность разговора в минутах. В зависимости от продолжительности разговора абоненту предоставляется скидка:

* до 2 минут включительно - скидки нет;
* до 5 минут включительно - скидка 5 %;
* до 10 минут включительно - скидка 20 %;
* до 20 минут - скидка 70 %;
* более 20 минут - разговор предоставляется бесплатно

1. Составить программу, которая для введённого *М* (числа рыб), будет выдавать фразу «Я поймал М рыб», причём согласовывать окончание слова «рыба» с числом М. Количество рыб может быть любым целым числом

Вариант 13

1. Составить программу для определения числа дней в месяце, если даны:

* номер месяца К (целое число от 1 до 12);
* целое число А, которое для високосного года равно 1, в противном случае равно 0

1. Вычислить значение функции F по одной из формул в зависимости от значения положительного X:

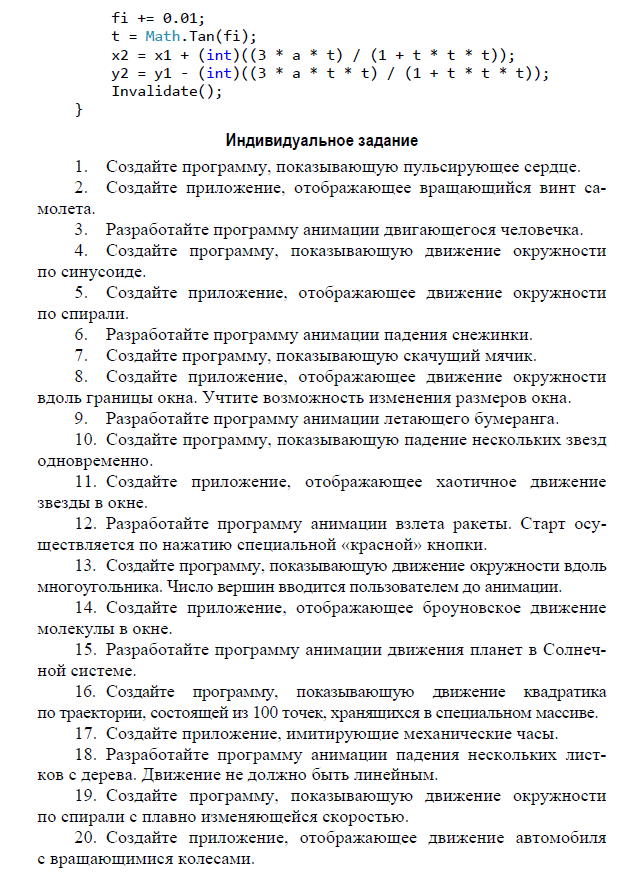
* при X < 6 F = X-A;
* при X = 15 или X = 20 F = X2 + B;
* в остальных случаях F = С + X

Значения А, В, С являются константами, значение X вводится с клавиатуры

Составить программу, которая для введённого В (числа подарков), будет выдавать фразу «У меня В подарков», причём согласовывать окончание слова «подарок» с числом В. Количество подарков может быть любым целым числом

Лабораторная работа 5

Тема «Разработка приложения с анимацией»



Лабораторная работа №6

Тема «Разработка приложения с анимацией. Анимация графиков»

Цель

Задание 1

Постройте графики следующих функций, используя шаг выборки данных по абсциссе 0,1

1. x2 на отрезке x ∈ [−2;2];

2. x3 на отрезке x ∈ [−2;2];

3. x4 на отрезке x ∈ [−2;2];

4. cos(2πt) на отрезке t ∈ [−10;10];

5. cos(2πt) на отрезке t ∈ [1;10];

6. e−t cos(2πt) на отрезке t ∈ [−10;10];

7. 4sin(πt + π/8) −1 на отрезке t ∈ [−10;10];

8. 2cos(t −2) +sin(2∗t−4) на отрезке t ∈ [−20π;10π];

9. ln(x +1) на отрезке x ∈ [0;e−1];

10. log2(|x|) на отрезке x ∈ [−4;4] за исключением точки x = 0;

11. 2x на отрезке x ∈ [−2;2];

12. ex на отрезке x ∈ [−2;2];

13. 2−x на отрезке x ∈ [−2;2];

14. на отрезке x ∈ [1;125];

15. на отрезке x ∈ [1;32]

Задание 2

Для построенного в рамках задания 1 графика измените:

• цвет линии;

• тип линии и маркеров;

• шаг выборки данных.

Далее введите сетку. Сохраните полученный график в файл, опробуйте сохранять файл в разных форматах: png, pdf, jpg, eps

Задание 3

Постройте семейство функций на одном графике различны ми цветами:

1. степенные многочлены с целыми степенями от 1 до 6 на отрезке [−1;1];

2. синусоиды y = sin(ωt) с частотами ω = 2π, ω = 3π, ..., ω = 8π на отрезке t ∈ [−1;1];

3. синусоиды y = sin(2πt + φ0) с начальными фазами φ0 = 0, φ0 = π/6, ..., φ0 =5π/6 на отрезке t ∈ [−1;1]

4. логарифмические функции log~~2~~(x), ln(x) и log10(x) на отрезке x ∈ [1;10];

5. гиперболические функции sh(x), ch(x) и th(x) на отрезке x ∈ [−10;10], для их вычисления воспользуйтесь их выражением через экспоненту.

Лабораторная работа 7-8

Тема: «Разработка игрового приложения»

Цель: научиться разрабатывать игровое приложение в среде разработки Python

Задание: Разработать игру на языке программирования Python

Вариант 1

Игра «Угадай число» для 2 игроков

Вариант 2

Игра «Крестики-нолики»

Вариант 3

Игра «Камень-ножницы-бумага»

Вариант 4

Игра «Вокруг слова»

Вариант 5

Игра «Виселица»

Вариант 6

Игра «Битва с драконом»

Вариант 7

Игра «Поле чудес на 3х игроков»

Вариант 8

Игра «Викторина»

Вариант 9

Игра «Кости на 2-4 игроков»

Вариант 10

Игра «Приключения в подземелье»

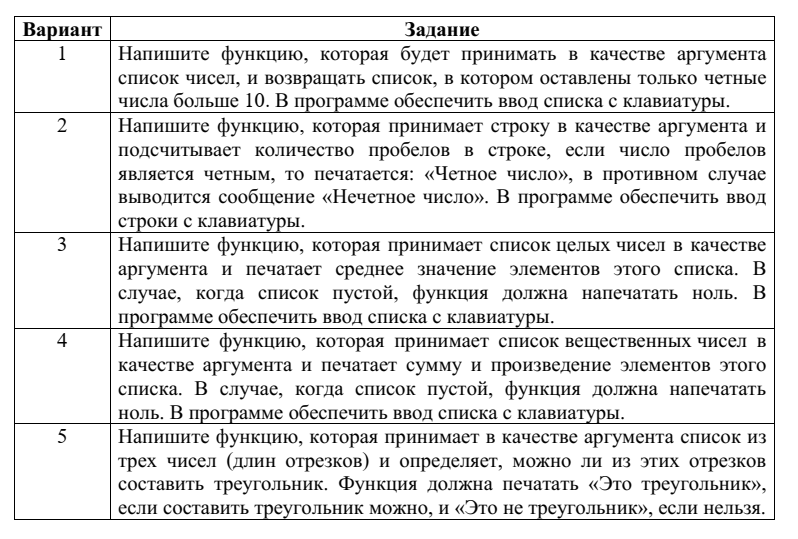
Лабораторная работа №9

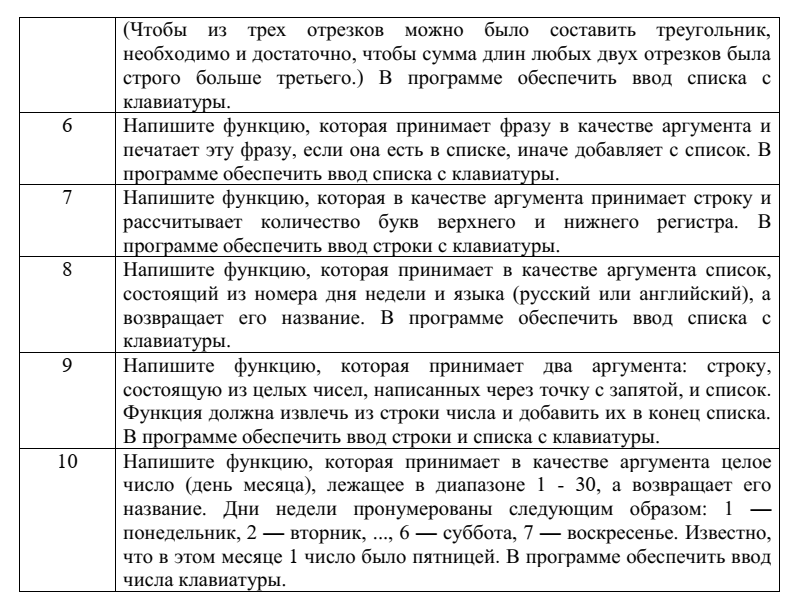
Тема Использование основных шаблонов

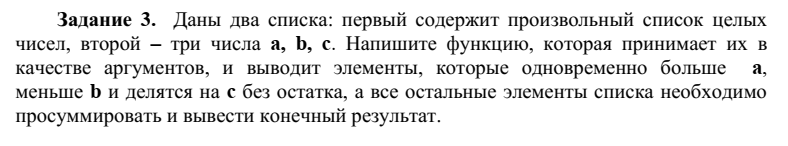
Цель

Задание: Используя основные шаблоны создайте функции по индивидуальному заданию. Можно использовать текстовые шаблоны для вывода данных из словаря или списка.

Задание 1. Напишите две функции S(r) и l(r), принимающие в качестве  
аргумента радиус окружности и возвращающие площадь круга и длину этой  
окружности соответственно. Затем напишите функцию krug(), которая спрашивает у пользователя радиус окружности, а затем при помощи функций S(r) и l(r) выводит на экран площадь круга и длину окружности, разделённые пробелом.







**Лабораторная работа №10-11**

**Тема «**Использование порождающих шаблонов**»**

***Цель работы***

Изучить шаблоны проектирования, относящиеся к классу порождающих, освоить применение шаблонов этого класса при разработке программных систем с применением объектно-ориентированных языков программирования.

***Теоретические сведения***  
Порождающие шаблоны *(creationalpatterns)* – шаблоны проектирования, которые абстрагируют процесс наследования. Они позволяют сделать систему независимой от способа создания, композиции и представления объектов. Шаблон, порождающий классы, использует наследование, чтобы изменять наследуемый класс, а шаблон, порождающий объекты, делегирует наследование другому объекту.

Эти шаблоны оказываются важны, когда система больше зависит от композиции объектов, чем от наследования классов. Получается так, что основной упор делается не на жестком кодировании фиксированного набора поведений, а на определении небольшого набора фундаментальных поведений, с помощью композиции которых можно получать любое число более сложных. Таким образом, для создания объектов с конкретным поведением требуется нечто большее, чем простое инстанцирование класса.

Порождающие шаблоны инкапсулируют знания о конкретных классах, которые применяются в системе. Они скрывают детали того, как эти классы создаются и стыкуются. Единственная информация об объектах, известная системе, – это их интерфейсы, определенные с помощью абстрактных классов. Следовательно, порождающие шаблоны обеспечивают большую гибкость при решении вопроса о том, что создается, кто это создает, как и когда. Можно собрать систему из «готовых» объектов с самой различной структурой и функциональностью статически (на этапе компиляции) или динамически (во время выполнения).

Иногда допустимо выбирать между тем или иным порождающим шаблоном. Например, есть случаи, когда с пользой для дела можно использовать как прототип, так и абстрактную фабрику. В других ситуациях порождающие шаблоны дополняют друг друга. Так, применяя строитель, можно использовать другие шаблоны для решения вопроса о том, какие компоненты нужно строить, а прототип часто реализуется вместе с одиночкой. Порождающие шаблоны тесно связаны друг с другом, их рассмотрение лучше проводить совместно, чтобы лучше были видны их сходства и различия.

**К порождающим шаблонам относятся:**

* абстрактная фабрика (abstract factory);
* строитель (builder);
* фабричный метод (factory method);
* отложенная (ленивая) инициализация (lazy initialization);
* пул одиночек (multiton);
* объектный пул (object pool);
* прототип (prototype);
* инициализация при получении ресурса (resource acquisition is initialization –RAII);
* одиночка (singleton).

***Задание***  
Реализовать программу для решения задачи согласно варианту с использованием требуемого порождающего шаблона проектирования.

**Содержание отчета**

1. Титульный лист.
2. Цель работы, задание, вариант.
3. Краткие теоретические сведения.
   * 3.1. Описание шаблона проектирования.
   * 3.2. Классовая диаграмма для шаблона проектирования в нотации UML.
4. Описание условий и хода работы.
   * 4.1. Используемые аппаратные и программные средства.
   * 4.2. Интерпретация шаблона проектирования для решения задачи.
   * 4.3. Классовая диаграмма для задачи в нотации UML.
5. Результаты работы.
   * 5.1. Текст программы (только классы, реализующие шаблон проектирования и клиентский код, использующий шаблон).
6. Анализ результатов работы. (Скриншоты с результатами выполнения программы, содержание базы данных, файлов – в зависимости от варианта).
7. Выводы о достоинствах и недостатках используемого шаблона проектирования.

***Контрольные вопросы***

1. Назначение порождающих шаблонов проектирования. Перечень порождающих шаблонов.
2. Назначение и структура шаблона *«абстрактная фабрика» (abstract factory)*. Преимущества и недостатки шаблона.
3. Назначение и структура шаблона *«строитель» (builder)*. Преимущества и недостатки шаблона.
4. Назначение и структура шаблона *«фабричный метод» (factory method)*. Преимущества и недостатки шаблона.
5. Назначение и структура шаблона *«отложенная (ленивая) инициализация» (lazy initialization)*. Преимущества и недостатки шаблона.
6. Назначение и структура шаблона *«пул одиночек» (multiton)*. Преимущества и недостатки шаблона.
7. Назначение и структура шаблона *«объектный пул» (object pool)*. Преимущества и недостатки шаблона.
8. Назначение и структура шаблона *«прототип» (prototype)*. Преимущества и недостатки шаблона.
9. Назначение и структура шаблона *«инициализация при получении ресурса» (resource acquisition is initialization – RAII)*. Преимущества и недостатки шаблона.
10. Назначение и структура шаблона *«одиночка» (singleton)*. Преимущества и недостатки шаблона.

***Варианты***

1) **Фабричный метод**

1.1. В базе данных системы электронного документооборота обрабатываются документы 3-х типов: письма, приказы и распоряжения о командировке. Каждый документ имеет номер, дату и краткую информацию о содержании. Кроме того, в письме указывается тип (входящее/исходящее) и корреспондент, в приказе –подразделение, срок выполнения и ответственный исполнитель, в распоряжении о командировке – сотрудник, период и место назначения.  
*Необходимо вывести:*

1. Полный перечень документов.

2. Содержание выбранного документа (по номеру документа).

1.2. В базе данных компьютерного интернет-магазина хранится номенклатура товаров 3-х типов: материнские платы, процессоры, жесткие диски. Каждый товар имеет номенклатурный номер, наименование и стоимость. Кроме того, для материнских плат указывается тип сокета, количество процессоров, тип оперативной памяти, частота системной шины, для процессоров – тип сокета, количество ядер, тактовая частота, техпроцесс, для жестких дисков – объем, скорость вращения, тип интерфейса.

*Необходимо вывести:*

1. Полную номенклатуру комплектующих.
2. 2. Детальную информацию по товару (по номенклатурному номеру).

1.3. В информационной системе реализуется ввод и сохранение данных об абитуриентах: фамилия, имя, отчество, дата рождения, баллы за ЕГЭ (3 предмета), желательные специальности (3 специальности).  
Необходимо реализовать ввод и сохранение информации об абитуриенте в 3-х форматах: текстовом, xml, таблица реляционной БД.

2) **Строитель**

2.1. Информационная система обеспечивает формирование отчета по лабораторным работам студента ИТ-специальности. Рассматриваются дисциплины «Базы данных», «Компьютерные сети», «Программирование». Отчет должен включать следующие разделы: титульный лист, цель работы, задание, теоретические сведения, описание экспериментальной установки, результаты работы, анализ результатов работы, выводы.  
Необходимо реализовать ввод данных и подготовку отчета в формате html с учетом особенностей дисциплины (наличие схем и других изображений или текста программы и т.п.).

2.2. Информационная система обеспечивает формирование отчетов по студенческим работам (рассмотреть любые 3 вида студенческих работ). Отчет в общем случае может включать следующие части: титульный лист, задание кафедры, аннотация, оглавление, основная часть, список источников, приложения.  
Необходимо реализовать ввод данных и подготовку отчета в формате html с учетом особенностей конкретного вида студенческой работы.

2.3. Информационная система предназначена для «сборки» комплектации автомобиля. В зависимости от марки и модели комплектация может включать, например, следующие категории элементов: экстерьер, интерьер, комфорт, безопасность, мультимедиа и т.п.  
Реализовать «сборку» различных комплектаций для выбранной модели автомобиля и подготовку сравнительной таблицы опций в формате html. При выполнении задания использовать реальный каталог выбранного автопроизводителя.

3) **Абстрактная фабрика**

3.1. В строительной компании информационная система решает задачу оптимального выбора поставщиков строительных материалов (всего 3 поставщика). В строительстве используется 3 вида материалов: кирпич, железобетонные плиты, бетон. Для каждого поставщика задан прайс-лист на материалы и максимальные суточные поставки каждого материала.  
Реализовать ввод необходимых данных (включая суточную потребность в материалах) и расчет оптимального плана поставок с использованием методов теории принятия решений. Учесть ситуацию, когда поставщик не поставляет конкретный вид материала.

3.2. Информационная система предназначена для «сборки» моделей автомобилей. Модель автомобиля может включать, например, следующие компоненты: кузов, двигатель, коробка передач, диски/шины (выбрать не менее 3 компонентов). Описать каждый из компонентов классом с подходящим набором атрибутов, реализовать также класс «модель».  
Реализовать «сборку» различных моделей выбранного автопроизводителя (не менее 3 моделей) на основе реального каталога. Вывести описание моделей в html-файл.

4) **Одиночка**

4.1. Реализоватьведение журнала событий программы в форматах: текстовый файл, xml-файл (конкретный формат должен выбираться). Сообщения, посылаемые в журнал, должны иметь тип (информация, предупреждение или ошибка), источник (наименование модуля программы), метку даты/времени и содержание. Реализациядолжна быть потокобезопасной, т.е. обеспечивать корректную работу журнала в случае, когда в приложении параллельно работает несколько потоков.

4.2. Реализовать хранение настроек программы в реестре Windows и xml-файле (конкретный формат должен выбираться). Настройки могут иметь строковый или числовой формат. Реализация должна быть потокобезопасной, т.е. обеспечивать корректную работу хранилища настроек в случае, когда в приложении параллельно работает несколько потоков.

5) **Пул объектов**

5.1. Реализовать менеджер для загрузки файлов с разделяемого сетевого ресурса (файл-сервера) с поддержкой многопоточной загрузки. Пользователь задает URI-адреса нескольких файлов, которые требуется загрузить. Для загрузки используется заданное количество объектов (потоков) – не более 1 потока на 1 файл. Должна поддерживаться авторизация для доступа к файлам.

5.2. Реализовать менеджер для загрузки html-страниц с web-сервера с поддержкой многопоточной загрузки. Пользователь задает URL страниц, которые требуется загрузить. Для загрузки используется заданное количество объектов (потоков) – не более 1 потока на 1 html-страницу. Должна поддерживаться работа через прокси-сервер.

Вариант 1 – задания 1.1 и 2.1

Вариант 2 – задания 1.2 и 2.2

Вариант 3 – задания 3.1 и 2.2

Вариант 4 – задания 4.1 и 3.2

Вариант 5 – задания 4.2 и 5.1

Вариант 6 – задания 5.2 и 3.1

Вариант 7 – задания 1.1 и 3.1

Вариант 8 – задания 2.1 и 4.1

Вариант 9 – задания 3.2 и 5.1

Вариант 10 – задания 1.2 и 2.3

**Лабораторная работа №12-13**

**Тема «Использование структурных шаблонов проектирования»**

***Цель работы***  
Изучить шаблоны проектирования, относящиеся к классу структурных, освоить применение шаблонов этого класса при разработке программных систем с применением объектно-ориентированных языков программирования.

***Теоретические сведения***  
В структурных *(structural)* паттернах рассматривается вопрос о том, как из классов и объектов образуются более крупные структуры. Структурные паттерны уровня класса используют наследование для составления композиций из интерфейсов и реализаций.

Простой пример – использование множественного наследования для объединения нескольких классов в один. В результате получается класс, обладающий свойствами всех своих родителей. Особенно полезен этот паттерн, когда нужно организовать совместную работу нескольких независимо разработанных библиотек.

Другой пример паттерна уровня класса – «адаптер». В общем случае адаптер делает интерфейс одного класса (адаптируемого) совместимым с интерфейсом другого, обеспечивая тем самым унифицированную абстракцию разнородных интерфейсов. Это достигается за счет закрытого наследования адаптируемому классу. После этого адаптер выражает свой интерфейс в терминах операций адаптируемого класса.

Вместо композиции интерфейсов или реализаций структурные паттерны уровня объекта компонуют объекты для получения новой функциональности. Дополнительная гибкость в этом случае связана с возможностью изменить композицию объектов во время выполнения, что недопустимо для статической композиции классов.

Примером структурного паттерна уровня объектов является «компоновщик». Он описывает построение иерархии классов для двух видов объектов: примитивных и составных. Последние позволяют создавать произвольно сложные структуры из примитивных и других составных объектов.

В паттерне «заместитель» объект берет на себя функции другого объекта. У заместителя есть много применений. Он может действовать как локальный представитель объекта, находящегося в удаленном адресном пространстве. Или представлять большой объект, загружаемый по требованию. Или ограничивать доступ к критически важному объекту. Заместитель вводит дополнительный косвенный уровень доступа к отдельным свойствам объекта. Поэтому он может ограничивать, расширять или изменять эти свойства.

Паттерн «приспособленец» определяет структуру для совместного использования объектов. Владельцы разделяют объекты, по меньшей мере, по двум причинам: для достижения эффективности и непротиворечивости. Приспособленец акцентирует внимание на эффективности использования памяти. В приложениях, в которых участвует очень много объектов, должны снижаться накладные расходы на хранение. Значительной экономии можно добиться за счет разделения объектов вместо их дублирования. Но объект может быть разделяемым, только если его состояние не зависит от контекста. У объектов-приспособленцев такой зависимости нет. Любая дополнительная информация передается им по мере необходимости. В отсутствие контекстных зависимостей объекты-приспособленцы могут легко разделяться.

Если паттерн «приспособленец» дает способ работы с большим числом мелких объектов, то «фасад» показывает, как один объект может представлять целую подсистему. Фасад представляет набор объектов и выполняет свои функции, перенаправляя сообщения объектам, которых он представляет.

Паттерн «мост» отделяет абстракцию объекта от его реализации, так что их можно изменять независимо.

Паттерн «декоратор» описывает динамическое добавление объектам новых обязанностей. Это структурный паттерн, который рекурсивно компонует объекты с целью реализации заранее неизвестного числа дополнительных функций. Например, объект-декоратор, содержащий некоторый элемент пользовательского интерфейса, может добавить к нему оформление в виде рамки или тени либо новую функциональность, например возможность прокрутки или изменения масштаба. Два разных оформления прибавляются путем простого вкладывания одного декоратора в другой. Для достижения этой цели каждый объект-декоратор должен соблюдать интерфейс своего компонента и перенаправлять ему сообщения. Свои функции (скажем, рисование рамки вокруг компонента) декоратор может выполнять как до, так и после перенаправления сообщения.

Многие структурные паттерны в той или иной мере связаны друг с другом.

**Таким образом, к структурным шаблонам относятся:**

* адаптер (adapter / wrapper);
* мост (bridge);
* компоновщик (composite);
* декоратор (decorator);
* фасад (facade);
* приспособленец (flyweight);
* заместитель (proxy).

***Задание***  
Реализовать программу для решения задачи согласно варианту с использованием требуемого структурного шаблона проектирования.

**Содержание отчета**

1. Титульный лист.
2. Цель работы, задание, вариант.
3. Краткие теоретические сведения.
   * 3.1. Описание шаблона проектирования.
   * 3.2. Классовая диаграмма для шаблона проектирования в нотации UML.
4. Описание условий и хода работы.
   * 4.1. Используемые аппаратные и программные средства.
   * 4.2. Интерпретация шаблона проектирования для решения задачи.
   * 4.3. Классовая диаграмма для задачи в нотации UML.
5. Результаты работы.
   * 5.1. Текст программы (только классы, реализующие шаблон проектирования и клиентский код, использующий шаблон).
6. Анализ результатов работы. (Скриншоты с результатами выполнения программы, содержание базы данных, файлов – в зависимости от варианта).
7. Выводы о достоинствах и недостатках используемого шаблона проектирования.

***Контрольные вопросы***

1. Назначение структурных шаблонов проектирования. Перечень структурных шаблонов.
2. Назначение и структура шаблона «адаптер» (adapter/ wrapper). Преимущества и недостатки шаблона.
3. Назначение и структура шаблона «мост» (bridge). Преимущества и недостатки шаблона.
4. Назначение и структура шаблона «компоновщик» (composite). Преимущества и недостатки шаблона.
5. Назначение и структура шаблона «декоратор» (decorator). Преимущества и недостатки шаблона.
6. Назначение и структура шаблона «фасад» (facade). Преимущества и недостатки шаблона.
7. Назначение и структура шаблона «приспособленец» (flyweight). Преимущества и недостатки шаблона.
8. Назначение и структура шаблона «заместитель» (proxy). Преимущества и недостатки шаблона.

***Варианты***

1) **Декоратор**

1.1. На базе класса .NET Framework System.IO.FileStream реализовать набор классов, реализующих следующие преобразования двоичных данных с сохранением в файл:  
1) кодирование в формате Base64;  
2) шифрование криптографическим алгоритмом (любым).  
Тестовое приложение должно обладать функциями записи в файл двоичных данных с использованием любой комбинации преобразований в любой последовательности и чтения данных из файла с обратным преобразованием.

1.2. На базе классов .NET Framework System.IO.StreamWriter, StreamReader реализовать набор классов, реализующих следующие преобразования текстовых данных с сохранением в файл:  
1) перевод по словам с использованием Google Translate API;  
2) шифрование методом перестановки слов по заданной схеме.  
Тестовое приложение должно обладать функциями записи в файл текстовых данных с использованием любой комбинации преобразований в любой последовательности и чтения данных из файла с обратным преобразованием.

1.3. На базе классов .NET Framework System.IO.StreamWriter, StreamReader реализовать набор классов, реализующих следующие преобразования текстовых данных с сохранением в файл:  
1) перевод фразами (предложениями) с использованием Google Translate API;  
2) исправление регистра символов в предложениях (первый символ – прописной, остальные – строчные).  
Тестовое приложение должно обладать функциями записи в файл текстовых данных с использованием любой комбинации преобразований в любой последовательности и чтения данных из файла с обратным преобразованием.

2) **Фасад**

2.1. Реализовать систему управления домашним кинотеатром, включающим следующие электронные компоненты: широкоформатный телевизор, Blu-Ray проигрыватель, ресивер с акустической системой. Каждый из компонентов имеет собственный интерфейс управления и ряд настроек (например, телевизор – формат отображения, режим яркости, 3D; ресивер – режим объемного звучания, громкость).  
Реализовать общий интерфейс управления домашним кинотеатром, предусматривающий включение и комплекс настроек для нескольких типовых режимов просмотра / прослушивания (например, мультфильмы, классическая музыка и т.п.).

2.2. Реализовать систему криптографической защиты данных на основе служб криптографии .NET Framework (пространство имен System.Security.Cryptography). Система криптографической защиты должна реализовывать цифровую подпись файла и шифрование данных с использованием выбранной комбинации алгоритмов цифровой подписи, хеширования и симметричного шифрования.  
Должно быть реализовано и обратное преобразование защищенного файла – дешифрование и проверка цифровой подписи. Дополнительно система защиты данных должна генерировать пару ключей для цифровой подписи и выгружать открытый ключ в формате xml. Секретный ключ для алгоритма симметричного шифрования должен запрашиваться у пользователя.

3) **Компоновщик**

3.1. В информационной системе хранится организационная структура предприятия (иерархия структурных подразделений). Для каждого структурного подразделения задан код, наименование и перечень должностей по штатному расписанию: наименование должности, количество ставок, оклад.  
Необходимо разработать набор классов для представления организационной структуры с использованием заданного шаблона проектирования. Должны быть реализованы возможности вывода штатного расписания для любого подразделения (включая подчиненные подразделения) с расчетом общего количества штатных единиц и суммарного оклада. Программа должна предусматривать добавление и удаление подразделений, должностей.

3.2. Реализовать утилиту анализа дерева каталогов и файлов для выбранного диска/каталога. Утилита должна считывать содержимое файловой системы и использовать набор классов, соответствующий заданному шаблону проектирования, для хранения содержимого файловой системы. Утилита должна реализовывать анализ файловой системы, вывод дерева каталогов и файлов с указанием занимаемого на диске места.

3.3. В САПР продукции машиностроительного предприятия хранится информация о комплектующих выпускаемой продукции, организованная в виде иерархии, узлами которого являются сборочные единицы, элементарные детали и закупаемые компоненты. Для элементарных деталей и закупаемых компонентов задается стоимость, для сборочных единиц, кроме того, время сборки и количество используемых деталей/компонентов каждого вида. САПР должна обеспечивать вывод иерархии производства продукции, расчет времени и стоимости производства изделия.

3.4. В системе моделирования используется представление математического выражения (функции nпеременных) в виде дерева вычисления, листьями которого являются переменные или константы, а промежуточными узлами – математические операции, включающие операторы (сумма, разность, произведение, частное) и элементарные функции (sqrt, exp, logи т.п.).  
Реализовать хранение выражения в виде дерева с использованием заданного шаблона проектирования, вычисление значения выражения при заданных значениях переменных и вывод выражения в текстовом виде в традиционной (инфиксной) нотации. Ввод выражения должен осуществляться в польской (префиксной) нотации с автоматическим формированием дерева вычисления.

Вариант 1 – задания 1.1 и 2.1

Вариант 2 – задания 1.2 и 2.2

Вариант 3 – задания 3.1 и 1.3

Вариант 4 – задания 1.3 и 3.4

Вариант 5 – задания 2.2 и 1.1

Вариант 6 – задания 3.2 и 1.1

Вариант 7 – задания 3.3 и 2.1

Вариант 8 – задания 3.4 и 2.2

Вариант 9 – задания 3.3 и 1.1

Вариант 10 – задания 3.4 и 1.1

**Лабораторная работа №3**

**Тема «Использование поведенческих шаблонов проектирования»**

***Цель работы***  
Изучить шаблоны проектирования, относящиеся к классу поведенческих, освоить применение шаблонов этого класса при разработке программных систем с применением объектно-ориентированных языков программирования.

***Теоретические сведения***  
Паттерны поведения связаны с алгоритмами и распределением обязанностей между объектами. Речь в них идет не только о самих объектах и классах, но и о типичных способах взаимодействия. Паттерны поведения характеризуют сложный поток управления, который трудно проследить во время выполнения программы. Внимание акцентировано не на потоке управления как таковом, а на связях между объектами.

В паттернах поведения уровня класса используется наследование – чтобы распределить поведение между разными классами. Самым простым и распространенным паттерном такого вида является «шаблонный метод», который представляет собой абстрактное определение алгоритма. Алгоритм здесь определяется пошагово. На каждом шаге вызывается либо примитивная, либо абстрактная операция. Алгоритм «обрастает мясом» за счет подклассов, где определены абстрактные операции. Другой паттерн поведения уровня класса – «интерпретатор», который представляет грамматику языка в виде иерархии классов и реализует интерпретатор как последовательность операций над экземплярами этих классов.

В паттернах поведения уровня объектов используется не наследование, а композиция. Некоторые из них описывают, как с помощью кооперации множество равноправных объектов справляется с задачей, которая ни одному из них не под силу. Важно здесь то, как объекты получают информацию о существовании друг друга. Объекты-коллеги могут хранить ссылки друг на друга, но это увеличит степень связанности системы. При максимальной степени связанности каждому объекту пришлось бы иметь информацию обо всех остальных. Эту проблему решает паттерн посредник. Посредник, находящийся между объектами-коллегами, обеспечивает косвенность ссылок, необходимую для разрывания лишних связей.

Паттерн «цепочка ответственности» позволяет и дальше уменьшать степень связанности. Он дает возможность посылать запросы объекту не напрямую, а по цепочке «объектов-кандидатов». Запрос может выполнить любой «кандидат», если это допустимо в текущем состоянии выполнения программы. Число кандидатов заранее не определено, а подбирать участников можно во время выполнения.

Паттерн «наблюдатель» определяет и отвечает за зависимости между объектами. Классический пример наблюдателя встречается в схеме модель/вид/контроллер (MVC), где все виды модели уведомляются о любых изменениях ее состояния.

Прочие паттерны поведения связаны с инкапсуляцией поведения в объекте и делегированием ему запросов. Паттерн «стратегия» инкапсулирует алгоритм объекта, упрощая его спецификацию и замену. Паттерн «команда» инкапсулирует запрос в виде объекта, который можно передавать как параметр, хранить в списке истории или использовать как-то иначе. Паттерн «состояние» инкапсулирует состояние объекта таким образом, что при изменении состояния объект может изменять поведение. Паттерн «посетитель» инкапсулирует поведение, которое в противном случае пришлось бы распределять между классами, а паттерн «итератор» абстрагирует способ доступа и обхода объектов из некоторого агрегата. Паттерн «хранитель» фиксирует и выносит за пределы объекта его внутреннее состояние, не нарушая инкапсуляции так, чтобы позднее можно было восстановить в нем объект.

**Таким образом, к поведенческим шаблонам относятся:**

* Цепочка ответственности (Chain of responsibility);
* Команда (Command);
* Интерпретатор (Interpreter);
* Итератор (Iterator);
* Посредник (Mediator);
* Хранитель (Memento);
* Наблюдатель (Observer);
* Состояние (State);
* Стратегия (Strategy);
* Шаблонный метод (Template method);
* Посетитель (Visitor).

***Задание***  
Реализовать программу для решения задачи согласно варианту с использованием требуемого поведенческого шаблона проектирования.

**Содержание отчета**

1. Титульный лист.
2. Цель работы, задание, вариант.
3. Краткие теоретические сведения.
   * 3.1. Описание шаблона проектирования.
   * 3.2. Классовая диаграмма для шаблона проектирования в нотации UML.
4. Описание условий и хода работы.
   * 4.1. Используемые аппаратные и программные средства.
   * 4.2. Интерпретация шаблона проектирования для решения задачи.
   * 4.3. Классовая диаграмма для задачи в нотации UML.
5. Результаты работы.
   * 5.1. Текст программы (только классы, реализующие шаблон проектирования и клиентский код, использующий шаблон).
6. Анализ результатов работы. (Скриншоты с результатами выполнения программы, содержание базы данных, файлов – в зависимости от варианта).
7. Выводы о достоинствах и недостатках используемого шаблона проектирования.

***Контрольные вопросы***

1. Назначение поведенческих шаблонов проектирования. Перечень поведенческих шаблонов.
2. Назначение и структура шаблона «Цепочка ответственности» (a Chain of responsibility). Преимущества и недостатки шаблона.
3. Назначение и структура шаблона «Команда» (Command). Преимущества и недостатки шаблона.
4. Назначение и структура шаблона «Интерпретатор» (Interpreter). Преимущества и недостатки шаблона.
5. Назначение и структура шаблона «Итератор» (Iterator). Преимущества и недостатки шаблона.
6. Назначение и структура шаблона «Посредник» (Mediator). Преимущества и недостатки шаблона.
7. Назначение и структура шаблона «Хранитель» (Memento). Преимущества и недостатки шаблона.
8. Назначение и структура шаблона «Наблюдатель» (Observer). Преимущества и недостатки шаблона.
9. Назначение и структура шаблона «Состояние» (State). Преимущества и недостатки шаблона.
10. Назначение и структура шаблона «Стратегия» (Strategy). Преимущества и недостатки шаблона.
11. Назначение и структура шаблона «Шаблонный метод» (Template method). Преимущества и недостатки шаблона.
12. Назначение и структура шаблона «Посетитель» (Visitor). Преимущества и недостатки шаблона.

***Варианты***

**1)** **Реализовать программу с использованием указанного паттерна.**

1.1. Цепочка ответственности.  
Реализовать программу с оконным интерфейсом и системой справки, которая дает справочную информацию по объекту окна, на который указал пользователь, а если таковая отсутствует, то справочную информацию для более общего объекта.  
Например, для получения справки пользователь может щелкнуть правой кнопкой мыши по какому-либо элементу графического интерфейса, выбрать пункт контекстного меню “?”, и получить информацию об объекте, по которому щелкнул (получить справку о кнопке “Печать”, получить справку о диалоговом окне “Печать документов”, или получить общую справку обо всем текстовом редакторе).

1.2. Интерпретатор.  
Реализовать библиотеку для задания образцов поиска строк (регулярных выражений). Должны поддерживаться:

* поиск строки по точному совпадению;
* поиск строки по неточному совпадению;
* применение оператора AND (и);
* применение оператора OR (или).

1.3. Посредник.  
Реализовать программное обеспечение авиадиспетчерской вышки, которая контролирует движение самолетов. Сами самолеты между собой напрямую не взаимодействуют. Должно быть минимум 3 взлетно-посадочных полосы и 5 терминалов, одновременно на каждой полосе и у каждого терминала может находиться только один самолет.

1.4. Хранитель.  
Реализовать графический редактор простых геометрических объектов с функцией пошагового восстановления состояний и возврата изменений.

1.5. Наблюдатель.  
Реализовать программу для построения графиков курса валют и/или биржевых курсов в режиме реального времени. Должна быть реализована возможность добавлять и удалять необходимые графики.

1.6. Состояние.  
Реализовать систему ведения заказов в интернет-магазине. Должны быть реализованы различные статусы заказа – Принят, В обработке, Оплачен, Разрешена отгрузка, Разрешена доставка, Доставлен и Отменен. При каждом изменении статуса должен быть оповещен покупатель, для статуса Разрешена отгрузка, должен быть оповещен кладовщик, а для статуса Разрешена доставка – Курьер.

1.7. Посетитель.  
Реализовать форму обратной связи с полями Имя, email-адрес, номер телефона и сообщение. Поля должны проверяться на корректность введенных данных, наличие SQL-инъекций и использование XSS-уязвимостей. Программа должна поддерживать дальнейшее расширение проверок для указанных полей.

**2) Необходимо определить проблему текущей архитектуры приложения (согласно варианту), поставить задачи на изменение программы, построить диаграмму классов и внести изменения в код, получив работоспособное приложение.**

2.1. Цепочка ответственности.  
Есть четыре приемника монет, каждый из которых принимает монеты только одного номинала (1, 5, 10 и 25 центов). Необходимо вывести на экран общую внесенную сумму и количество внесенных монет каждого номинала.

#include <iostream.h>

#include <stdlib.h>

#include <time.h>

class Slot {

public:

Slot(int val) {

value\_ = val;

count\_ = 0;

}

int accept(int coin) {

if (coin == value\_) {

count\_++;

return 1;

}

return 0;

}

int getCount() {

return count\_;}

private:

int value\_;

int count\_;

};

class Quarter : public Slot {

public:

Quarter() : Slot(25) { }

};

class Dime : public Slot {

public:

Dime() : Slot(10) { }

};

class Nickel : public Slot {

public:

Nickel() : Slot(5) { }

};

class Penny : public Slot {

public:

Penny() : Slot(1) { }

};

int pickCoin(){

static int choices[4] = { 1, 5, 10, 25 };

return choices[rand() % 4];

}

void main(void){

Slot\* slots[4];

slots[0] = new Quarter;

slots[1] = new Dime;

slots[2] = new Nickel;

slots[3] = new Penny;

int i, coin, total;

time\_t t;

srand((unsigned)time(&t));

for (i = 0, total = 0; i < 10; i++)

{

coin = pickCoin();

cout << coin << " ";

total += coin;

for (int j = 0; j < 4; j++)

if (slots[j]->accept(coin)) break;

}

cout << "\ntotal deposited is $" << total / 100 << "."

<< (total % 100 < 10 ? "0" : "") << total % 100 << endl;

cout << "quarters -" << slots[0]->getCount() << endl;

cout << "dimes -" << slots[1]->getCount() << endl;

cout << "nickels -" << slots[2]->getCount() << endl;

cout << "pennies -" << slots[3]->getCount() << endl;

}

// 10 5 25 5 1 25 5 25 1 25

// total deposited is $1.27

// quarters - 4

// dimes - 1

// nickels - 3

// pennies - 2

//

// 25 25 10 5 1 5 10 5 1 5

// total deposited is $0.92

// quarters - 2

// dimes - 2

// nickels - 4

// pennies - 2

//

// 5 10 25 10 25 1 25 10 1 25

// total deposited is $1.37

// quarters - 4

// dimes - 3

// nickels - 1

// pennies - 2

2.2. Команда.  
Необходимо реализовать командный интерфейс с поддержкой очереди команд. Поддерживаемые команды – сжатие, распаковка и передача файла.

#include <iostream.h>

#include <string.h>

struct Command;

class Queue {

public:

Queue() {

add\_ = remove\_ = 0;

}

void enque(Command\* c) {

array\_[add\_] = c;

add\_ = (add\_ + 1) % SIZE;

}

Command\* deque() {

int temp = remove\_;

remove\_ = (remove\_ + 1) % SIZE;

return array\_[temp];

}

private:enum { SIZE = 10 };

Command\* array\_[SIZE];

int add\_;

int remove\_;

};

class File {

public:

File(char\* n) { strcpy(name\_, n); }

void unarchive() { cout << "unarchive " << name\_ << endl; }

void compress() { cout << "compress " << name\_ << endl; }

void transfer() { cout << "transfer " << name\_ << endl; }

private:

char name\_[30];

};

enum Action { unarchive, transfer, compress };

struct Command {

Command(File\* f, Action a) { receiver = f; action = a; }

File\* receiver;

Action action;

};

Command\* input[8] = {

new Command(new File("irImage.dat"), unarchive),

new Command(new File("screenDump.jpg"), transfer),

new Command(new File("paper.ps"), unarchive),

new Command(new File("widget.tar"), compress),

new Command(new File("esmSignal.dat"), unarchive),

new Command(new File("msword.exe"), transfer),

new Command(new File("ecmSignal.dat"), compress),

new Command(new File("image.gif"), transfer)

};

void main(void)

{

Queue que;

Command\* cmd;

int i;

for (i = 0; i < 8; i++)

que.enque(input[i]);

for (i = 0; i < 8; i++)

{

cmd = que.deque();

if (cmd->action == unarchive)

cmd->receiver->unarchive();

else if (cmd->action == transfer)

cmd->receiver->transfer();

else if (cmd->action == compress)

cmd->receiver->compress();

}

}

// unarchive irImage.dat

// transfer screenDump.jpg

// unarchive paper.ps

// compress widget.tar

// unarchive esmSignal.dat

// transfer msword.exe

// compressecmSignal.dat

// transfer image.gif

2.3. Итератор.  
Реализовать обход бинарного дерева с возможностью управления обходом.

#include <iostream.h>

#include <stdlib.h>

#include <time.h>

struct Node {

int value;

Node\* left;

Node\* right;

Node() { left = right = 0;}

friend ostream& operator<< (ostream& os, Node& n) { return os << n.value;

}

};

class BST {

private:

Node\* root;

int size;

public:BST() {

root = 0;

}

void add(int in) {

if (root == 0) {

root = new Node;

root->value = in;

size = 1;

return;

}

add(in, root);

}

void traverse() { traverse(root); }

private:

void add(int in, Node\* current) {

if (in < current->value)

if (current->left == 0) {

current->left = new Node();

current->left->value = in;

size++;

}

else add(in, current->left);

else

if (current->right == 0) {

current->right = new Node();

current->right->value = in;

size++;

}

else add(in, current->right);

}

void traverse(Node\* current) {

if (current->left != 0) traverse(current->left);

cout << current->value << " ";

if (current->right != 0) traverse(current->right);

}

};

void main(void) {

BST bst;

time\_t t;

srand((unsigned)time(&t));

cout << "original: ";

for (int i = 0, val; i < 15; i++) {

val = rand() % 49 + 1;

cout << val << " ";

bst.add(val);

}

cout << "\ntraverse: ";

bst.traverse();

cout << endl;

}

// original: 11 43 7 2 22 3 25 40 41 36 32 11 24 11 37

// traverse: 2 3 7 11 11 11 22 24 25 32 36 37 40 41 43

// Iterator: 2 3 7 11 11 11 22 24 25 32 36 37 40 41 43

// Iterator: 2 3 7 11 11 11 22 24 25 32 36 37 40 41 43

2.4. Хранитель.  
Реализовать игру, состоящую из нескольких раундов, в которой несколько участников угадывают случайное число. Каждый клиент подключается отдельно и отгадывает свое число.

#include <stdlib.h>

#include <time.h>

#include <string.h>

class GuessGame {

private:int numbers[10];

char names[10][20];

int total;

public:GuessGame() {

time\_t t;

srand((unsigned)time(&t));

total = 0;

}

void join(char\* name) {

strcpy(names[total], name);

numbers[total++] = rand() % 30 + 1;

}

int evaluateGuess(char\* name, int guess) {

int i;

for (i = 0; i < total; i++)

if (!strcmp(names[i], name)) break;

if (guess == numbers[i]) return 0;

return ((guess > numbers[i]) ? 1 : -1);

}

};

struct Game {

char name[20];

int min, max, done;

Game() { min = 1; max = 30; done = 0; }

};

void main(void) {

GuessGame guessServer;

const int MAX = 3;

Game games[MAX];

int gamesComplete = 0;

int guess, ret;

for (int i = 0; i < MAX; i++) {

cout << "Enter name:";

cin >> games[i].name;

guessServer.join(games[i].name);

}

while (gamesComplete != MAX) {

for (int j = 0; j < MAX; j++) {

if (games[j].done) continue;

cout << games[j].name << " ("<<games[j].min << '-' << games[j].max << "): ";

cin >> guess;

ret = guessServer.evaluateGuess(games[j].name, guess);

if (ret == 0) {

cout << " lights!! sirens!! balloons!!" << endl;

games[j].done = 1;

gamesComplete++;

}

else if (ret < 0) {

cout << " too low" << endl;

games[j].min = guess;

}

else {

cout << " too high" << endl;

games[j].max = guess;

}

}

}

}

// Enter name: Tom

// Enter name: Dick

// Enter name: Harry

// Tom (1-30): 10

// too low

// Dick (1-30): 15

// too high

// Harry (1-30): 20

// too high

// Tom (10-30): 22

// too high

// Dick (1-15): 8

// lights!! sirens!! balloons!!

// Harry (1-20): 10

// too low

// Tom (10-22): 16

// too high

// Harry (10-20): 15

// too low

// Tom (10-16): 13

// too high

// Harry (15-20): 17

// too low

// Tom (10-13): 11

// lights!! sirens!! balloons!!

// Harry (17-20): 18// lights!! sirens!! balloons!

2.5. Состояние.  
Реализовать переключение состояний конечного автомата:

|  |  |  |  |
| --- | --- | --- | --- |
| Команда | on | off | ack |
| Состояние A | A | B | C |
| Состояние B |  | A | C |
| Состояние C |  |  | B |

#include <iostream.h>

enum State { A, B, C };

enum Message { on, off, ack };

State currentState;

Message messageArray[10] = { on, off, off, ack, ack, ack, ack, on, off, off };

void main(void) {

currentState = B;

for (int index = 0; index < 10; index++) {

if (currentState == A) {

if(messageArray[index] == on) {

cout << "A, on ==> A" << endl;

currentState = A;

}

else if (messageArray[index] == off) {

cout << "A, off ==> B" << endl;

currentState = B;

}

else if (messageArray[index] == ack) {

cout << "A, ack ==> C" << endl;

currentState = C;

}

}

else if (currentState == B) {

if (messageArray[index] == on) {

cout << "undefined combo" << endl;

}

else if (messageArray[index] == off) {

cout << "B, off ==> A" << endl;

currentState = A;

}

else if (messageArray[index] == ack) {

cout << "B, ack ==> C" << endl;

currentState = C;

}

}

else if (currentState == C) {

if (messageArray[index] == on) {

cout << "undefined combo" << endl;

}

else if (messageArray[index] == off) {

cout << "undefined combo" << endl;

}

else if (messageArray[index] == ack) {

cout << "C, ack ==> B" << endl;

currentState = B;

}

}

}

}

// undefined combo

// B, off ==> A

// A, off ==> B

// B, ack ==> C

// C, ack ==> B

// B, ack ==> C

// C, ack ==> B

// undefined combo

// B, off ==> A

// A, off ==> B

2.6. Стратегия.  
Реализовать проверку введенной пользователем строки – число, строка в нижнем регистре и строка в верхнем регистре.

#include <iostream.h>

#include <string.h>

int isdigit(char ch) { return (ch >= '0' && ch <= '9') ? 1 : 0; }

int isupper(char ch) { return (ch >= 'A' && ch <= 'Z') ? 1 : 0; }

int islower(char ch) { return (ch >= 'a' && ch <= 'z') ? 1 : 0; }

class DataEntry {

public:

void setValidationType(char ch){ type = ch; }

void interact();

private:

char type;

int validate(char\*);

};

void DataEntry::interact() {

char answer[20];

cout << "Prompt: ";

cin >> answer;

while (strcmp(answer, "quit")) {

if (validate(answer)) cout << "\*\*\* good \*\*\*" << endl;

else cout << "\*\*\* bad \*\*\*" << endl;

cout << "Prompt: ";cin >> answer;

}

}

int DataEntry::validate(char\* input) {

int valid;

unsigned i;

valid = 1;

if (type == 'n') {

for (i = 0; i < strlen(input); i++)

if (!isdigit(input[i])) {

valid = 0;

break;

}

}

else if (type == 'u') {

for (i = 0; i < strlen(input); i++)

if (!isupper(input[i])) {

valid = 0;

break;

}

}

else if (type == 'l') {

for (i = 0; i < strlen(input); i++)

if (!islower(input[i])) {

valid = 0;

break;

}

}

return valid;

}

void main(void) {

DataEntry dialog;

char answer;

cout << "Input type [ (n)umber, (u)pper, (l)ower, e(x)it ]: ";

cin >> answer;

while (answer != 'x') {

dialog.setValidationType(answer);

dialog.interact();

cout << "Input type [ (n)umber, (u)pper, (l)ower, e(x)it ]: ";

cin >> answer;

}

}

// Input type [ (n)umber, (u)pper, (l)ower, e(x)it ]: n

// Prompt: sdf

// \*\*\* bad \*\*\*

// Prompt: 234

// \*\*\* good \*\*\*

// Prompt: quit

// Input type [ (n)umber, (u)pper, (l)ower, e(x)it ]: u

// Prompt: sdf

// \*\*\* bad \*\*\*

// Prompt: 234

// \*\*\* bad \*\*\*

// Prompt: GHJ

// \*\*\* good \*\*\*

// Prompt: quit

// Input type [ (n)umber, (u)pper, (l)ower, e(x)it ]: x

Вариант 1 – задания 1.1 и 2.1

Вариант 2 – задания 1.2 и 2.2

Вариант 3 – задания 1.3 и 2.3

Вариант 4 – задания 1.4 и 2.4

Вариант 5 – задания 1.5 и 2.5

Вариант 6 – задания 1.6 и 2.6

Вариант 7 – задания 1.7 и 2.1

Вариант 8 – задания 1.1 и 2.3

Вариант 9 – задания 1.3 и 2.6

Вариант 10 – задания 1.5 и 2.1

ЛАБОРАТОРНАЯ РАБОТА 16

Тема «Оптимизация вычислительного алгоритма»

**Цель занятия:**

- приобретение практических навыков оптимизации программ, используя методы и законы оптимизации;

- способствовать формированию аналитического мышления, общих и профессиональных компетенций;

- формировать осознание принадлежности к конкретной профессиональной группе.

**Задачи:**

- применение различных методов и приемов оптимизации для улучшения программ, написанных на различных языках программирования и оценка целесообразности оптимизации.

|  |  |
| --- | --- |
| *До оптимизации* | *После оптимизации* |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

1. **Выполнение лабораторной работы**

Порядок выполнения работы:

*-* Решите задачу своего варианта, используя только элементарные конструкции (последовательность, ветвления, циклы). Программа должна быть рабочей!

- Оптимизировать программу (можно использовать процедуры или функции).

- Для созданных программ оценить метрические характеристики по Холстеду;

- Сравнить полученные результаты. Оформить результаты в таблицу. Сделать соответствующие выводы.

**Теоретические сведения:**

Оптимизация-это улучшение программы.

Оптимизация бывает двух видов: глобальная и локальная.

**Глобальная оптимизация** *–* это оптимизация, при которой экономия ресурсов достигается путём изменения алгоритма всей программы или большей ее части.

**Локальная оптимизация –** это оптимизация, при которой экономия ресурсов достигается путём изменения участка программы.

*Способы локальной оптимизации:*

* Чистка программы
* Разгрузка участков повторения
* Реализация действий
* Упрощение действий
* Экономия памяти
* Сокращение программы

Основные законы оптимизации:

1. Делать проверки всего того, что вводит пользователь.
2. Не делать проверки внутри цикла.

Так как циклы - это слабые места любой программы, и оптимизацию нужно начинать с них.

1. Надо знать принцип работы компьютера.Чем лучше будете знать как компьютер выполняет код программы, тем лучше можно оптимизировать этот код.

**Расчет метрики**

### Метрики Холстеда

Метрика Холстеда относится к метрикам, вычисляемым на основании анализа числа строк и синтаксических элементов исходного кода программы.

Метрика Холстеда позволяет оценить размер (в словах) и объем в битах программы на стадии анализа требований. Используя нормы выработки операторов в день можно оценить время на разработку.

Основу метрики Холстеда составляют четыре измеряемые характеристики программы:

* n1 — число уникальных операторов программы, включая символы-разделители, имена процедур и знаки операций (словарь операторов);
* n2 — число уникальных операндов программы (словарь операндов);
* N1 — общее число операторов в программе;
* N2 — общее число операндов в программе.

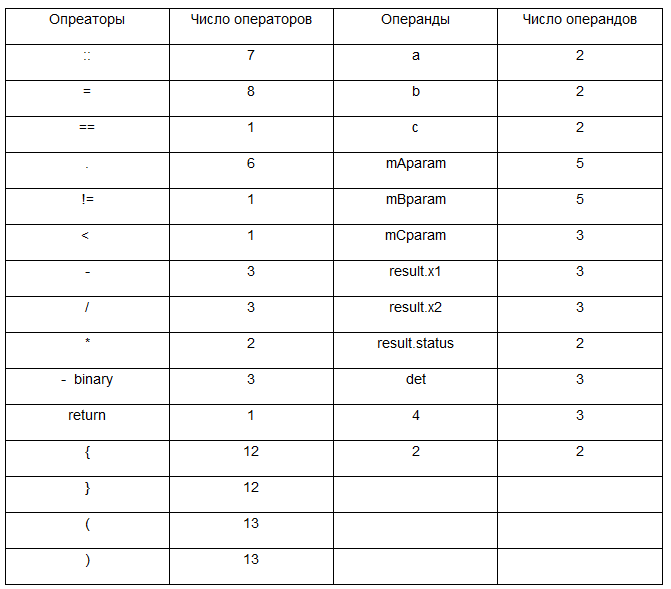
+, \*, /, - это операторы

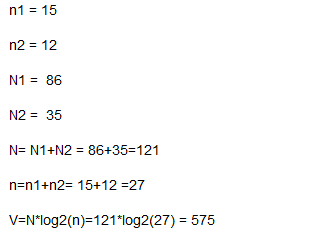
x, у, z, 999, -25, number1 - это операнды

На основании этих характеристик рассчитываются оценки:

* ***Словарь программы***(Halstead Program Vocabulary, HPVoc): n = n1 + n2;
* ***Длина программы***(Halstead Program Length, HPLen): N = N1 + N2;
* ***Объем программы***(Halstead Program Volume, HPVol): V = N log2 n;
* ***Сложность программы***(Halstead Difficulty, HDiff): D = (n1/2) × (N2 / n2);
* На основе показателя HDiff предлагается оценивать усилия программиста при разработке при помощи показателя HEff (Halstead Effort): H = D × V.

Пример вычисления метрики Холстеда:





**Задание:**

* 1. **Решите задачу своего варианта, используя только элементарные конструкции (последовательность, ветвления, циклы). Программа должна быть рабочей!**
  2. **Оптимизировать программу (можно использовать процедуры или функции).**

**Внимание! Оптимизированная программа должна содержать проверки всех переменных, которые вводятся с клавиатуры.**

* 1. **Для созданных программ оценить метрические характеристики по Холстеду;**
  2. **Сравнить полученные результаты. Оформить результаты в таблицу. Сделать соответствующие выводы.**

|  |  |
| --- | --- |
| ***Вариант задания №*** | |
| *Код программы:* | *Код оптимизированной программы:* |
|  |  |
| *Расчет метрики:* | *Расчет метрики:* |
|  |  |

**Варианты:**

*Вариант 1*. Даны натуральные числа n,a1,…,an. Определить количество членов ak последовательности a1,…,an:

а) являющихся нечетными числами;

б) кратных 3 и не кратных 5;

в) имеющих четные порядковые номера и являющихся нечетными числами.

*Вариант 2*. Даны натуральные числа n,q1,…,qn. Найти те члены qi, последовательности q1,…,qn, которые

а) являются удвоенными нечетными числами;

б) при делении на 7 дают остаток 1, 2 или 5;

в) делятся на 5 и не делятся на 7, найти количество и сумму тех членов последовательности.

*Вариант 3*. Даны целые числа A1, ..., A80. Получить сумму тех чисел данной последовательности, которые

а) кратны 5;

б) нечетны и отрицательны;

в) удовлетворяют условию A[i]<i2.

**Отчет** о выполненной лабораторной работе должен содержать:

- цель лабораторной работы, вариант задания и условие задачи;

- таблицу, содержащую задачу, решенную двумя способами (два кода программы) и расчет соответствующих метрик;

- выводы о выполненной лабораторной работе.

**Контрольные вопросы:**

* + 1. Какой процесс называют «Оптимизацией программы»?
    2. Какие существуют способы оптимизации? Для чего они применяются?
    3. Приведите примеры приемов алгоритмической оптимизации.

ЛАБОРАТОРНАЯ РАБОТА № 17

Тема: «Оптимизация вычислительного алгоритма»

Цель: ознакомиться с методами одномерного поиска, используемыми в многомерных методах минимизации функций *n* переменных. Сравнить различные алгоритмы по эффективности на тестовых примерах. Сравнить изменение программы при оптимизации вычислительного алгоритма.

Порядок выполнения работы:

*-* Решите задачу своего варианта, используя только элементарные конструкции (последовательность, ветвления, циклы). Программа должна быть рабочей;

- Оптимизировать программу (можно использовать процедуры или функции).

- Для созданных программ оценить метрические характеристики по Холстеду;

- Сравнить полученные результаты. Оформить результаты в таблицу. Сделать соответствующие выводы.

Задание 1. Реализовать методы дихотомии, золотого сечения, исследовать их сходимость и провести сравнение по числу вычислений функции для достижения заданной точности от 10-1 до 10-7 . Построить график зависимости количества вычислений минимизируемой функции от десятичного логарифма задаваемой точности  .

Задание 2. Реализовать алгоритм поиска интервала, содержащего минимум функции

Задание 3. Реализовать метод Фибоначчи, сравнить его с методами дихотомии и золотого сечения

**Варианты** **заданий**

* 1. *f*  *x*   *x*  12 , *x* 2, 20Задание 1
  2. *f*  *x*   *x*  22 , *x* 2, 20Задание 2
  3. *f*  *x*   *x*  32 , *x* 2, 20Задание 3
  4. *f*  *x*   *x*  42 , *x* 2, 20Задание 1
  5. *f*  *x*   *x*  52 , *x* 2, 20Задание 2
  6. *f*  *x*   *x*  62 , *x* 2, 20Задание 3
  7. *f*  *x*   *x*  72 , *x* 2, 20Задание 1
  8. *f*  *x*   *x*  82 , *x* 2, 20Задание 2
  9. *f*  *x*   *x*  92 , *x* 2, 20Задание 3
  10. *. f*  *x*   *x*  102 , *x* 2, 20Задание 1
  11. *f*  *x*   *x*  112 , *x* 2, 20Задание 2
  12. *f*  *x*   *x*  122 , *x* 2, 20. Задание 3

1. *f*  *x*   *x*  12 , *x* 2, 20Задание 2
2. *f*  *x*   *x*  22 , *x* 2, 20Задание 3
3. *f*  *x*   *x*  32 , *x* 2, 20Задание 1
4. *f*  *x*   *x*  42 , *x* 2, 20Задание 2
5. *f*  *x*   *x*  52 , *x* 2, 20Задание 3
6. *f*  *x*   *x*  62 , *x* 2, 20Задание 1
7. *f*  *x*   *x*  72 , *x* 2, 20Задание 2
8. *f*  *x*   *x*  82 , *x* 2, 20Задание 3
9. *f*  *x*   *x*  92 , *x* 2, 20Задание 1
10. *. f*  *x*   *x*  102 , *x* 2, 20Задание 2
11. *f*  *x*   *x*  112 , *x* 2, 20Задание 3
12. *f*  *x*   *x*  122 , *x* 2, 20. Задание 1

# Контрольные вопросы

1. Метод дихотомии.
2. Метод золотого сечения.
3. Метод Фибоначчи.
4. Метод квадратичной интерполяции (*метод* *парабол*)
5. Алгоритм поиска интервала, содержащего минимум функции.

ЛАБОРАТОРНАЯ РАБОТА № 18

Тема: «Оптимизация данных и выражений»

Цель: ознакомиться с методами поиска минимума функции *n* переменных в оптимизационных задачах без ограничений

Порядок выполнения работы:

*-* Решите задачу своего варианта, используя только элементарные конструкции (последовательность, ветвления, циклы). Программа должна быть рабочей;

- Оптимизировать программу (можно использовать процедуры или функции).

- Проведите оптимизацию данных, проведя правильность ввода исходных данных (сообщения если введены данные неверного типа или не разрешенные значения);

- Для созданных программ оценить метрические характеристики по Холстеду;

- Сравнить полученные результаты. Оформить результаты в таблицу. Сделать соответствующие выводы.

Варианты заданий

Условие задачи: найти максимум заданной функции:

Реализовать **два** **метода** поиска экстремума функции (разного порядка). Включить в реализуемый алгоритм собственную процедуру, реализующую одномерный поиск по направлению. Методы поиска для самостоятельной реализации выбираются студентом в зависимости от уровня сложности.

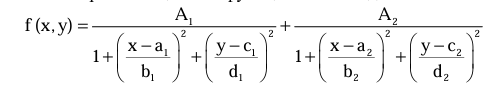
**Методы** **поиска** **для** **самостоятельной** **реализации**

|  |  |  |
| --- | --- | --- |
| **Метод** | **Порядок** | **Уровень** **сложности** |
| метод Гаусса | 0 | 1 |
| метод Хука и Дживса | 0 | 1 |
| метод Пауэлла | 0 | 2 |
| метод вращающихся координат  (Розенброка) | 0 | 2 |
| метод деформируемого  многогранника | 0 | 3 |
| метод наискорейшего спуска | 1 | 2 |
| метод сопряженных градиентов в  модификации Флетчера-Ривса | 1 | 3 |
| метод Ньютона | 2 | 3 |

Для нечетных вариантов целевая функция имеет вид:



Для четных вариантов целевая функция имеет вид:



Варианты:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № варианта | A1 | A2 | a1 | a2 | b1 | b2 | c1 | c2 | d1 | d2 |
| 1 | 2 | 3 | 1 | 2 | 2 | 3 | 1 | 3 | 1 | 2 |
| 2 | 1 | 3 | 2 | 1 | 3 | 1 | 2 | 1 | 3 | 2 |
| 3 | 1 | 2 | 3 | 2 | 1 | 2 | 1 | 2 | 3 | 1 |
| 4 | 2 | 1 | 1 | 3 | 2 | 3 | 2 | 1 | 1 | 3 |
| 5 | 3 | 1 | 2 | 1 | 1 | 2 | 3 | 1 | 2 | 1 |
| 6 | 2 | 1 | 2 | 3 | 3 | 1 | 2 | 1 | 1 | 3 |
| 7 | 2 | 3 | 1 | 1 | 2 | 3 | 1 | 3 | 2 | 3 |
| 8 | 3 | 2 | 2 | 2 | 1 | 3 | 2 | 3 | 2 | 1 |
| 9 | 2 | 3 | 3 | 1 | 1 | 1 | 2 | 1 | 1 | 3 |
| 10 | 1 | 2 | 3 | 2 | 1 | 2 | 2 | 2 | 1 | 1 |
| 11 | 2 | 1 | 1 | 2 | 2 | 2 | 1 | 1 | 3 | 2 |
| 12 | 3 | 1 | 3 | 3 | 2 | 1 | 2 | 1 | 1 | 3 |

Контрольные вопросы

1. Метод Гаусса.

2. Метод Хука и Дживса.

3. Метод Розенброка (вращающихся координат).

4. Метод Пауэлла.

5. Метод деформируемого многогранника.

6. Метод наискорейшего спуска.

7. Метод сопряженных градиентов и его модификации.

8. Метод Ньютона и его модификации.

9. Методы переменной метрики.

ЛАБОРАТОРНАЯ РАБОТА № 19

Тема: «Оптимизация данных и выражений»

Цель: ознакомиться с методами штрафных функций при решении задач нелинейного программирования. Изучить типы штрафных и барьерных функций, их особенности, способы и области применения, влияние штрафных функций на сходимость алгоритмов, зависимость точности решения задачи нелинейного программирования от величины коэффициента штрафа.

Порядок выполнения работы:

*-* Решите задачу своего варианта, используя только элементарные конструкции (последовательность, ветвления, циклы). Программа должна быть рабочей;

- Оптимизировать программу (можно использовать процедуры или функции).

- Проведите оптимизацию данных, проведя правильность ввода исходных данных (сообщения если введены данные неверного типа или не разрешенные значения);

- Для созданных программ оценить метрические характеристики по Холстеду;

- Сравнить полученные результаты. Оформить результаты в таблицу. Сделать соответствующие выводы.

|  |  |
| --- | --- |
| № | Вид работы |
| 1. | Применяя методы поиска минимума 0-го порядка, реализовать программу для решения задачи нелинейного программирования с использованием **метода** **штрафных** **функций**. |
| 2. | Исследовать сходимость **метода** **штрафных** **функций** в зависимости от   * выбора штрафных функций, * начальной величины коэффициента штрафа, * стратегии изменения коэффициента штрафа, * начальной точки, * задаваемой точности  .   Сформулировать выводы. |
| 3\* | Применяя методы поиска минимума 0-го порядка, реализовать программу для решения задачи нелинейного программирования с ограничением типа неравенства (**только** **задача** **а**) с использованием  **метода** **барьерных** **функций**. |
| 4\* | Исследовать сходимость **метода** **барьерных** **функций** (**только** **задача** **а**) в зависимости от   * выбора барьерных функций, * начальной величины коэффициента штрафа, * стратегии изменения коэффициента штрафа, * начального приближения, * задаваемой точности  . |

Варианты заданий

|  |  |  |  |
| --- | --- | --- | --- |
| № |  |  |  |
| 1 | f  x, y   5 x  y 2   x  22  min  x  y  1 | 13 | f  x, y   5 x  y 2   x  22  min  x   y |
| 2 | f  x, y   10 y  x2  y2  min  x  y  1 | 14 | f  x, y   10 y  x2  y2  min  x  2  y |
| 3 | f  x, y    x  y 2  10 x  52  min  x  y  0 | 15 | f  x, y    x  y 2  10 x  52  min  x  1  y |
| 4 | f  x, y   2 x  y 2  14 y  32  min  y  x  1 | 16 | f  x, y   2 x  y 2  14 y  32  min  x   y |
| 5 | f  x, y   4 y  x2  3 x 12  min  x  y  1 | 17 | f  x, y   4 y  x2  3 x 12  min  y  x  1 |
| 6 | f  x, y   7 x  y2   y  62  min  y  x  2 | 18 | f  x, y   7 x  y2   y  62  min  x   y |
| 7 | f  x, y    x  y 2  4 y2  min  x  y  5 | 19 | f  x, y    x  y 2  4 y2  min  y  x  2 |
| 8 | f  x, y   5 x  y 2   x  22  min  x  y  1 | 20 | f  x, y   5 x  y 2   x  22  min  x  y |
| 9 | f  x, y   4 x  y 2  x2  min  y  x  5 | 21 | f  x, y   4 x  y 2  x2  min  x  2  y |
| 10 | f  x, y    x  y 2  10 y  22  min  y  x | 22 | f  x, y    x  y 2  10 y  22  min  y  x 1 |
| 11 | f  x, y   10 x  y 2   y  22  min  y  x  1 | 23 | f  x, y   10 x  y 2   y  22  min  x  y |
| 12 | f  x, y   8 x  y 2   x  22  min  x  y  5 | 24 | f  x, y   8 x  y 2   x  22  min  x  y |

# Контрольные вопросы

1. Метод штрафных функций.
2. Метод барьерных функций.
3. Стратегии изменения коэффициентов штрафа.
4. Виды штрафных функций для ограничений равенств.
5. Виды штрафных функций для ограничений неравенств.
6. Виды барьерных функций.

ЛАБОРАТОРНАЯ РАБОТА № 20

Тема: «Оптимизация циклов»

Цель: исследовать результаты применения методов оптимизации цмклов. Ознакомиться со статистическими методами поиска при решении задач нелинейного программирования. Изучить методы случайного поиска при определении глобального экстремума функции.

.

Порядок выполнения работы:

*-* Решите задачу своего варианта, используя только элементарные конструкции (последовательность, ветвления, циклы). Программа должна быть рабочей;

- Оптимизировать программу (можно использовать процедуры или функции).

- Проведите оптимизацию данных, проведя правильность ввода исходных данных (сообщения если введены данные неверного типа или не разрешенные значения);

- Для созданных программ оценить метрические характеристики по Холстеду;

- Сравнить полученные результаты. Оформить результаты в таблицу. Сделать соответствующие выводы.

1.1. Разработать программу для решения задачи поиска глобального экстремума с использованием метода простого случайного поиска и трех алгоритмов глобального поиска.

1.2. Исследовать метод простого случайного поиска глобального экстремума при различных ε и P. Результат представить в таблице:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ε | P | N | (x\*, y\*) | F(x\*, y\*) |
|  |  |  |  |  |

1.3. Исследовать алгоритмы поиска глобального экстремума. Сравнить результаты поиска по количеству вычислений функции и найденной точке экстремума. Исследование провести при различных значениях числа попыток m.

Условие задачи: Найти максимум заданной функции:



на области 10 10 x − ≤ ≤ , 10 10 y − ≤ ≤ .

Варианты

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| С1 | 5 | 2 | 6 | 4 | 1 | 7 | 2 | 2 | 10 | 9 | 4 | 10 |
| С2 | 5 | 4 | 2 | 9 | 2 | 9 | 3 | 1 | 9 | 2 | 9 | 2 |
| С3 | 3 | 2 | 4 | 1 | 10 | 10 | 8 | 7 | 7 | 2 | 8 | 10 |
| С4 | 2 | 2 | 8 | 5 | 7 | 5 | 2 | 8 | 3 | 9 | 8 | 5 |
| С5 | 2 | 2 | 8 | 5 | 7 | 5 | 2 | 8 | 3 | 9 | 8 | 5 |
| С6 | 3 | 3 | 8 | 6 | 9 | 7 | 8 | 4 | 10 | 5 | 8 | 3 |
| a1 | -9 | -3 | -3 | 7 | 0 | -6 | 3 | 5 | -5 | -6 | -9 | 3 |
| a2 | -10 | -6 | 4 | -9 | 0 | -7 | -5 | 2 | 8 | -7 | 9 | 6 |
| a3 | 8 | 2 | -8 | 6 | 3 | 8 | 0 | -9 | -8 | -8 | -10 | -10 |
| a4 | -6 | 6 | -6 | -8 | -7 | -9 | 3 | 0 | 0 | 6 | -3 | 6 |
| a5 | -9 | -3 | 3 | -10 | 6 | 9 | -4 | -3 | -10 | 6 | -2 | -6 |
| a6 | 1 | 8 | -6 | -2 | 6 | 0 | 6 | -3 | 5 | 9 | -9 | -1 |
| b1 | -3 | 6 | 9 | 9 | -1 | 9 | -4 | 4 | -1 | 7 | 4 | 4 |
| b2 | -1 | -8 | -7 | -1 | -4 | 7 | -6 | 0 | -9 | 5 | 8 | 5 |
| b3 | 8 | -8 | 3 | 5 | -2 | -8 | -1 | -6 | -4 | 0 | -2 | -1 |
| b4 | -5 | 8 | -9 | -2 | -6 | 3 | 7 | -3 | 3 | 4 | -2 | 8 |
| b5 | -3 | -4 | -2 | -8 | -10 | 8 | 0 | 7 | 1 | -9 | -7 | -2 |
| b6 | 5 | -1 | -8 | -4 | 1 | 7 | 5 | 3 | 0 | -2 | 0 | -1 |

Контрольные вопросы

1. Алгоритм простого случайного поиска.

2. Алгоритм с парной пробой.

3. Алгоритм статистического градиента.

4. Алгоритм наилучшей пробы с направляющим гиперквадратом.

5. Алгоритмы глобального поиска.

ЛАБОРАТОРНАЯ РАБОТА № 21

Тема: «Оптимизация циклов»

Цель: ознакомиться с методами штрафных функций при решении задач нелинейного программирования. Изучить типы штрафных и барьерных функций, их особенности, способы и области применения, влияние штрафных функций на сходимость алгоритмов, зависимость точности решения задачи нелинейного программирования от величины коэффициента штрафа.

Порядок выполнения работы:

*-* Решите задачу своего варианта, используя только элементарные конструкции (последовательность, ветвления, циклы). Программа должна быть рабочей;

- Оптимизировать программу (можно использовать процедуры или функции).

- Проведите оптимизацию данных, проведя правильность ввода исходных данных (сообщения если введены данные неверного типа или не разрешенные значения);

- Для созданных программ оценить метрические характеристики по Холстеду;

- Сравнить полученные результаты. Оформить результаты в таблицу. Сделать соответствующие выводы.

Задание:

С использованием разработанного программного обеспечения из лабораторной работы № 18 исследовать алгоритмы на квадратичной функции , функции Розенброка и на заданной в соответствии с вариантом тестовой функции, осуществляя спуск из различных исходных точек (не менее двух). Исследовать сходимость алгоритма, фиксируя точность определения минимума/максимума, количество итераций метода и количество вычислений функции в зависимости от задаваемой точности поиска. Результатом выполнения данного пункта должны быть выводы об объёме вычислений в зависимости от задаваемой точности и начального приближения.

Контрольные вопросы

1. Метод Гаусса.

2. Метод Хука и Дживса.

3. Метод Розенброка (вращающихся координат).

4. Метод Пауэлла.

5. Метод деформируемого многогранника.

6. Метод наискорейшего спуска.

7. Метод сопряженных градиентов и его модификации.

8. Метод Ньютона и его модификации.

9. Методы переменной метрики.