ФОНД ОЦЕНОЧНЫХ СРЕДСТВ

для проведения текущей и промежуточной аттестации

по учебной дисциплине

«Языки и методы программирования»

для направления подготовки/специальности

Прикладная математика и информатика

Направленность программы:

Исследование операций и системный анализ

1. Описание показателей (дескрипторов) и критериев оценивания компетенций на различных этапах их формирования

Контроль качества освоения дисциплины включает в себя текущий контроль успеваемости и промежуточную аттестацию. Текущий контроль успеваемости и промежуточная аттестация обучающихся проводятся в целях установления соответствия достижений обучающихся поэтапным требованиям образовательной программы к результатам обучения и формирования компетенций.

Компетенции	Показатели* (дескрипторы)	Критерии в соответствии с уровнем освоения ОП			Оценочное средство (промежуточная аттестация)
		пороговый (удовлетворительно) 55-69 баллов	стандартный (хорошо) 70-84 балла	эталонный (отлично) 85-100 баллов	
ОПК-2. Способен использовать и адаптировать существующие математические методы и системы программирования для разработки и реализации алгоритмов решения прикладных задач	Знть	базовые термины программирования	основные методы и средства получения, хранения и переработки информации	значение для современного человека целостного представления о современных системах программирования.	Лабораторные работы
	Уметь	излагать основные концепции современных средств и систем программирования; разрабатывать в изученной системе программирования Assembler, Pascal, Delphi , C# собственные приложения; работать в пошаговом режиме отладки программы; компилировать и тестировать свою программу;	использовать автономный отладчик Turbo Debugger для обнаружения места и причины логических ошибок; пользоваться справочной системой изученных систем программирования	оценивать собственные программы, определять потребности в дальнейшем образовании; создавать на изученных языках приложения с интерфейсом различного типа (консольным и оконным); владеть диалоговым и графическим инструментарием ОС.	

	Владеть	использованием аппаратных ресурсов компьютера при написании программ.	языками высокого уровня с низким порогом вхождения и на их примере понять основы языка с высоким порогом и многочисленными возможностями	знанием формулировки для чего мне необходимо программирование, что я хочу уметь делать в итоге; понятийным аппаратом по использованию языка и платформы программирования	
ПК-1 Способен применять современные информационные технологии при проектировании, реализации, оценке качества и анализа эффективности программного обеспечения для решения задач в различных предметных областях	Знать	терминологическую систему программирования; специфику программирования, историю развития языков программирования	значение, иерархию и взаимосвязь различных языков программирования	закономерности развития современных систем программирования; актуальные проблемы программирования в рамках учебной информации.	
	Уметь	вводить и выводить данные в консольном режиме, с помощью инструментов оконного интерфейса и файлов; работать с графическим инструментарием программной среды	создавать и обрабатывать статические и динамические структуры данных; создавать и работать с типами данных, определяемых пользователем; выявлять существенные свойства и методы объектов; анализировать свою собственную программу; делать проверку корректности ввода-вывода данных	анализировать и оценивать полученные результаты и их обрабатывать; самостоятельно получать и расширять знания, пользоваться различными источниками информации и встроенной справочной системой	Тестирование

	Владеть	самостоятельным изучением баз данных, сетевых протоколов, особенностей графической подсистемы etc и т.д. ,	умением помочь начинать изучать основы программирования на примере уже выбранного языка	в зависимости от выбора языка программирования помимо языка программирования изучать углублённо, например, прикладные программы, приложения, серверные решения, веб-приложения, RIA, игры, низкоуровневые решения	
ПК-3 Способен проектировать информационные системы и программные комплексы на стадиях их жизненного цикла	Знать	знает основные концептуальные положения функционального, логического проектирования информационных систем и программных комплексов	знает основные концептуальные положения объектно-ориентированного и визуального проектирования информационных систем	знает основные концептуальные положения объектно-ориентированного и визуального проектирования информационных систем и программных комплексов	Лабораторные работы

	Уметь	умеет критически оценивать и тестировать свою собственную программу, выделять в ней главное, структурировать, представлять в доступном и понятном для других виде; применять модульное программирование для решения задач (совместное использование языков высокого и низкого уровня); усовершенствовать свои знания и изучать далее определённую систему программирования или новую современную систему программирования; использовать базовые положения изученной системы программирования для дальнейшего изучения других систем программирования	умеет проектировать информационные системы	умеет проектировать информационные системы и программные комплексы на стадиях их жизненного цикла	
	Владеть	умением определяться с выбором дальнейшего направления развития изучения систем программирования	умением разработки прикладных программ, приложений, низкоуровневых решений; ответственностью за результаты своих действий и качество выполненных заданий	умением разработки серверных решений, веб-приложений, RIA, игр, умением принимать нестандартные решения профессиональных задач	

*Показатели (дескрипторы) перечисляются по всей компетенции, если индикаторы компетенции сформулированы в виде «действия».

2. Описание критериев и шкал оценивания результатов обучения по дисциплине

2.1. Критерии и шкалы оценивания результатов обучения при проведении текущего контроля успеваемости

Текущий контроль предназначен для проверки хода и качества формирования компетенций, стимулирования учебной работы обучаемых и совершенствования методики освоения новых знаний. Он обеспечивается проведением семинаров, оцениванием контрольных заданий, проверкой конспектов лекций, выполнением индивидуальных и творческих заданий, периодическим опросом обучающихся на занятиях. Контролируемые разделы (темы) дисциплины, компетенции и оценочные средства представлены в таблице.

№ п/п	Контролируемые разделы (темы) дисциплины*	Код контролируемой компетенции и/или индикаторы компетенции	Наименование оценочного средства**
1	Структурный подход к программированию	ОПК-2	Лабораторные работы
2	Модульное программирование. Программирование абстрактных типов данных.	ОПК-2	Лабораторные работы
3	Объектно-ориентированное программирование.	ПК-3	Лабораторные работы, контрольная № 1
4	Объектно-ориентированный анализ и проектирование: основные понятия и терминология.	ПК-3	Лабораторные работы
5	Объектно-ориентированный анализ и проектирование: основные понятия и терминология.	ПК-3	Лабораторные работы, контрольная № 2
6	Цели анализ и проектирование приложений.	ПК-1	Тестирование
7	Сопоставление ОО языков программирования.	ПК-1	Тестирование

Критерии и шкала оценивания тестирования

Оценка	*Критерий оценки*
«зачтено»	*Выполнение более 60% тестовых заданий*
«не зачтено»	*Выполнение менее 60% тестовых заданий*

Критерии выполнения лабораторных работ

Критерии выполнения отчета на max балл

Лабораторная работа выполненена полностью, без погрешностей и замечаний.

Критерии выполнения отчета на min балл

Лабораторная работа полностью не выполнена.

Критерии оценки принятого отчета (в диапазоне от min до max балла)

• программный код не оптимален;

• использованы глобальные переменные;
• не на все вопросы получены верные ответы при защите работы;

Критерии дополнительных баллов за личностные качества

• работа выполнена верно с первого раза, на занятии по расписанию;
• соблюдение рекомендуемого стиля программирования;
• наличие, отсутствие или неполнота смысловых комментариев в программе.

2.2. Критерии и шкалы оценивания результатов обучения при проведении промежуточной аттестации

ДЕ 1 (100 баллов), 4 семестр	1. Контрольная работа № 1 по теме «Базовые алгоритмические структуры» (10 баллов) 2. Контрольная работа № 2 по теме «Объектно-ориентированное программирование.» (10 баллов) 3. Защита лабораторных работ (8 лаб. * (от 4 – 6 б.) = 32 - 48 баллов 4. Посещение лекций (16 л. * 2 б. = 32 б)
ДЕ 2 (100 баллов), 5 семестр	1. Защита лабораторных работ (8 лаб. * (от 4 – 8 б.) = 32 – 64 баллов 2. Посещение лекций (17 л. * 1 б.) = 17 б. 3. Тестирование 1 (9 баллов) 4. Итоговое тестирование (10 баллов)

Промежуточная аттестация предназначена для определения уровня освоения всего объема учебной дисциплины. Для оценивания результатов обучения при проведении промежуточной аттестации используется четырёх балльная шкала.

Основные виды систем оценивания, 4 семестр

Европейская	100-балльная	4-балльная	2-балльная
A	94-100	отлично	зачтено
A-	90-94		
B+	85-89		
B	80-84	хорошо	
B-	75-79		
C+	70-74		
C	65-69	удовлетворительно	
C-	60-64		
D	52-59		
F	50-53	неудовлетворительно	не зачтено

Основные виды систем оценивания, 5 семестр

Европейская	100-балльная	4-балльная	2-балльная
A	94-100	отлично	зачтено
A-	90-94		
B+	85-89		
B	80-84	хорошо	
B-	75-79		

C+	70-74		
C	65-69	удовлетворительно	
C-	60-64		
D	58-59		
F	50-57	неудовлетворительно	не зачтено

Промежуточная аттестация предназначена для определения уровня освоения всего объема учебной дисциплины. Для оценивания результатов обучения при проведении промежуточной аттестации в 5 семестре используется четырехбалльная шкала: «Отлично», «Хорошо», «Удовлетворительно», «Неудовлетворительно».

Шкала оценивания	Критерии	Уровень освоения компетенций
Отлично	наличие глубоких и исчерпывающих знаний в объеме пройденного программного материала, правильные и уверенные действия по применению полученных знаний на практике, грамотное и логически стройное изложение материала при ответе, знание дополнительно рекомендованной литературы	Эталонный
Хорошо	наличие твердых и достаточно полных знаний программного материала, незначительные ошибки при освещении заданных вопросов, правильные действия по применению знаний на практике, четкое изложение материала	Стандартный
Удовлетворительно	наличие твердых знаний пройденного материала, изложение ответов с ошибками, уверенно исправляемыми после дополнительных вопросов, необходимость наводящих вопросов, правильные действия по применению знаний на практике	Пороговый
Неудовлетворительно	наличие грубых ошибок в ответе,	Компетенции не

	непонимание сущности излагаемого вопроса, неумение применять знания на практике, неуверенность и неточность ответов на дополнительные и наводящие вопросы.	сформированы

3. Типовые контрольные задания или иные материалы, необходимые для оценки знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций в процессе освоения образовательной программы

3.1. Оценочные средства текущего контроля успеваемости

Задания к лабораторным работам

Лабораторная работа №1а. Арифметические функции

Найти значение алгебраического выражения, соответствующего варианту задания. Вывести результаты на печать. Все результаты выводить в развернутом виде (например: «Сумма чисел A и B равна 3.7854»). Значения вводимых величин должны иметь не менее четырех значащих цифр и задаются студентом самостоятельно.

Варианты

1. $$t = \frac{2\cos\left(x - \frac{\pi}{6}\right)}{0.5 + \sin^2 y}\left(1 + \frac{z^2}{3 - z^2/5}\right).$$

При x=14.26, y=-1.22, z=3.5×10-2 t=0.564849.

2. $$u = \frac{\sqrt[3]{8 + |x-y|^2 + 1}}{x^2 + y^2 + 2} - e^{|x-y|}\left(tg^2 z + 1\right)^x.$$

При x=-4.5, y=0.75×10-4, z=0.845×10-2 u=-55.6848.

3. $$v = \frac{1 + \sin^2(x+y)}{\left|x - \frac{2y}{1 + x^2 y^2}\right|} x^{|y|} + \cos^2\left(arctg\frac{1}{z}\right).$$

При x=3.74×10-2, y=-0.825, z=0.16×102, v=1.0553.

4. $$w = \left|\cos x - \cos y\right|^{\left(1+2\sin^2 y\right)}\left(1 + z + \frac{z^2}{2} + \frac{z^3}{3} + \frac{z^4}{4}\right).$$

При x=0.4×104, y=-0.875, z=-0.475×10-3 w=1.9873.

5. $$\alpha = \ln\left(y^{-\sqrt{|x|}}\right)\left(x - \frac{y}{2}\right) + \sin^2 arctg(z).$$

При x=-15.246, y=4.642×10-2, z=20.001×102 α =-182.036.

6. $$\beta = \sqrt{10\left(\sqrt[3]{x} + x^{y+2}\right)}\left(\arcsin^2 z - |x - y|\right).$$

При x=16.55×10-3, y=-2.75, z=0.15 β =-40.63069.

7. $$\gamma = 5arctg(x) - \frac{1}{4}\arccos(x)\frac{x + 3|x - y| + x^2}{|x - y|z + x^2}.$$

При x=0.1722, y=6.33, z=3.25×10-4 γ =-205.306.

8. $$\varphi = \frac{e^{|x-y|}|x - y|^{x+y}}{arctg(x) + arctg(z)} + \sqrt[3]{x^6 + \ln^2 y}.$$

При x=-2.235×10-2, y=2.23, z=15.221 φ =39.374.

9. $$\psi = \left|x^{\frac{y}{x}} - \sqrt[3]{\frac{y}{x}}\right| + (y - x)\frac{\cos y - \frac{z}{(y - x)}}{1 + (y - x)^2}.$$

При x=1.825×102, y=18.225, z=-3.298×10-2 ψ =1.2131.

10. $$a = 2^{-x}\sqrt{x + \sqrt[4]{|y|}}\sqrt[3]{e^{x-1/\sin z}}.$$

При x=3.981×10-2, y=-1.625×103, z=0.512 a=1.26185.

11. $$b = y^{\sqrt{|x|}} + \cos^3(y)\frac{|x-y|\left(1+\frac{\sin^2 z}{\sqrt{x+y}}\right)}{e^{|x-y|}+\frac{x}{2}}.$$

При x=6.251, y=0.827, z=25.001 b=0.7121.

12. $$n = 2^{(y^x)} + (3^x)^y - \frac{y\left(arctgz - \frac{\pi}{6}\right)}{|x| + \frac{1}{y^2+1}}.$$

При x=3.251, y=0.325, z=0.466×10-4 c=4.025.

13. $$f = \frac{\sqrt[4]{y+\sqrt[3]{x-1}}}{|x-y|(\sin^2 z + tgz)}.$$

При x=17.421, y=10.365×10-3, z=0.828×105 f=0.33056.

14. $$g = \frac{y^{x+1}}{\sqrt[3]{|y-2|}+3} + \frac{x+\frac{y}{2}}{2|x+y|}(x+1)^{-1/\sin z}.$$

При x=12.3×10-1, y=15.4, z=0.252×103 g=82.8257.

15. $$h = \frac{x^{y+1}+e^{y-1}}{1+x|y-tgz|}(1+|y-x|) + \frac{|y-x|^2}{2} - \frac{|y-x|^3}{3}.$$

При x=2.444, y=0.869×10-2, z=-0.13×103, h=-0.49871.

Лабораторная работа №1б. Управляющие конструкции

Составить программу вычисления функции F(x), разложенной в ряд, сходящийся в заданной области. Вычисления проводить до тех пор, пока модуль разности между последующим и предыдущим членами ряда не будет меньше или равен ε. Значение ε задается константой и для всех вариантов составляет $0{,}001$.

Решить задачу для различных значений x. При решении задач массивы не использовать. Специальные функции для возведения в степень не использовать.

Варианты

№ п/п	Функция F(x)	Область сходимости	*x*
1	$1 - x + x^2 - \ldots + (-1)^{n-1} x^{n-1} + \ldots$	$\lvert x \rvert < 1$	$x_1 = 0{,}07$ $x_2 = 0{,}95$ $x_3 = -0{,}5$
2	$1 - 2x + 3x^2 - \ldots + (-1)^{n-1} n x^{n-1} + \ldots$	$\lvert x \rvert < 1$	$x_1 = 0{,}1$ $x_2 = 0{,}77$ $x_3 = -0{,}9$
№ п/п	Функция F(x)	Область сходимости	*x*
3	$x - \frac{x^2}{2} + \frac{x^3}{3} - \ldots + (-1)^{n-1} \frac{x^n}{n} + \ldots$	$-1 < x \le 1$	$x_1 = 0{,}1$ $x_2 = 0{,}8$ $x_3 = -0{,}62$
4	$x - \frac{x^3}{3} + \frac{x^5}{5} - \ldots + (-1)^{n-1} \frac{x^{2n-1}}{2n-1} + \ldots$	$\lvert x \rvert \le 1$	$x_1 = 1$ $x_2 = 0{,}005$ $x_3 = -0{,}65$
5	$2x + \frac{(2x)^2}{2!} + \ldots + \frac{(2x)^n}{n!} + \ldots$	$\lvert x \rvert < \infty$	$x_1 = 1$ $x_2 = 17{,}5$ $x_3 = -0{,}35$
6	$x - \frac{x^3}{3!} + \frac{x^5}{5!} - \ldots + (-1)^{n-1} \frac{x^{2n-1}}{(2n-1)!} + \ldots$	$\lvert x \rvert < \infty$	$x_1 = -2{,}2$ $x_2 = 0{,}75$ $x_3 = 5{,}6$
7	$1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \ldots + (-1)^{n-1} \frac{x^{2n}}{(2n)!} + \ldots$	$\lvert x \rvert < \infty$	$x_1 = -2$ $x_2 = 0{,}98$ $x_3 = 15$

8	$\frac{\pi}{2} - \left[\frac{1}{x} - \frac{1}{3x^3} + \ldots + (-1)^{n-1}\frac{1}{(2n-1)x^{2n-1}} + \ldots\right]$	$\lvert x\rvert > 1$	$x_1 = 10$ $x_2 = 1{,}5$ $x_3 = -5$
9	$x\left[x^2 - \frac{x^4}{1!3} + \frac{}{2!5} + \ldots + (-1)^{n-1}\frac{x^{2n}}{n!(2n+1)} + \ldots\right]$	$\lvert x\rvert < \infty$	$x_1 = 1$ $x_2 = 7{,}5$ $x_3 = -2{,}1$
10	$\cos x + \frac{1}{2}\cos 2x + \ldots + \frac{1}{n}\cos nx + \ldots$	$0 < x \le \pi$	$x_1 = 1$ $x_2 = \frac{\pi}{5}$ $x_3 = 2{,}5$
11	$\cos x + \frac{1}{3}\cos 3x + \ldots + \frac{1}{2n-1}\cos(2n-1)x + \ldots$	$0 < x \le \pi$	$x_1 = \frac{\pi}{4}$ $x_2 = 1$ $x_3 = \frac{\pi}{2}$

№ п/п	Функция F(x)	Область сходимости	x
12	$\frac{\sin x}{2} + \frac{\sin 2x}{2^2} + \ldots + \frac{\sin nx}{2^n} + \ldots$	$\lvert x\rvert < \infty$	$x_1 = 1$ $x_2 = 3\pi$ $x_3 = -4{,}1$
13	$2\left[x + \frac{x^3}{3} + \ldots + \frac{x^{2n-1}}{2n-1} + \ldots\right]$	$\lvert x\rvert < 1$	$x_1 = 0{,}1$ $x_2 = -0{,}73$ $x_3 = 0{,}81$
14	$\frac{18}{\pi}\left(\sin x + \ldots + \frac{1}{(2n-1)^3}\sin(2n-1)x + \ldots\right)$	$0 < x \le 2\pi$	$x_1 = 5$ $x_2 = \frac{2\pi}{3}$ $x_3 = \frac{\pi}{6}$
15	$2\left[\frac{1}{x} + \frac{1}{3x^3} + \ldots + \frac{1}{(2n-1)x^{2n-1}} + \ldots\right]$	$\lvert x\rvert > 1$	$x_1 = 2$ $x_2 = -3{,}15$ $x_3 = 1{,}45$

16	$\frac{4a}{\pi}\left(\sin x + \frac{\sin 3x}{3} + \ldots + \frac{\sin(2n-1)x}{2n-1} + \ldots\right)$	$0 < x \le \pi$	$x_1 = \frac{\pi}{3}$ $x_2 = 2$ $x_3 = \frac{\pi}{4}$ $a = 12$
17	$\cos x - \frac{1}{2}\cos 2x + \ldots + (-1)^{n-1}\frac{1}{n}\cos nx + \ldots$	$0 \le x < \pi$	$x_1 = 0{,}5$ $x_2 = \frac{\pi}{3}$ $x_3 = 1$
18	$\frac{\pi^2}{3} - 4\left(\cos x - \frac{\cos 2x}{2^2} + \ldots + (-1)^{n-1}\frac{\cos nx}{n^2} + \ldots\right)$	$\lvert x \rvert \le \pi$	$x_1 = 0{,}1$ $x_2 = -3$ $x_3 = 1{,}5$
19	$\frac{4}{\pi}\left(\sin x + \frac{\sin 3x}{3^2} + \ldots + (-1)^{n-1}\frac{\sin(2n-1)x}{(2n-1)^2} + \ldots\right)$	$\lvert x \rvert \le \frac{\pi}{2}$	$x_1 = 0{,}15$ $x_2 = 1{,}25$ $x_3 = -0{,}85$
20	$(x-1) - \frac{(x-1)^2}{2} + \ldots + (-1)^{n-1}\frac{(x-1)^n}{n} + \ldots$	$0 < x \le 2$	$x_1 = 1$ $x_2 = 0{,}13$ $x_3 = 1{,}85$

Лабораторная работа №2. Отладка приложений

В задании всего 2 варианта: вариант 1 выполняют те, у кого основной вариант нечетный, вариант 2 выполняют те, у кого основной вариант четный. Каждый вариант содержит 15 задач, содержащих те или иные ошибки (логические, ошибки синтаксиса и т.п.). Задача состоит в следующем: используя средства отладчика среды, обнаружить все ошибки и сделать программу работоспособной (т.е. выполняющей то, что от нее требуется).

Для отчета по данной лабораторной работе необходимо:

- показать работающие программы;
- код программ должен быть структурированным;
- к каждой программе необходимо составить список обнаруженных ошибок (согласно сообщениям отладчика), а также пояснения того, как вы исправляли эти ошибки.

Вариант 1	**Вариант 2**
Задача 1. Дано натуральное число n. Получить все его натуральные делители.	

`Main()`
`{ Int n,j;`
`Cout>>'n'>>endl;`
`Cin<<n;`
`While j<n`
	`#include <iostream>`
	`main()`
	`for (i=1, n div 2, i++);`
	`if n % i =0`
	`cout<<i`

```
  { if (n/i = 0) and (n<>j)
 cout>>j>>endl; }
```

```
}
```

Задача 2. Дано 100 вещественных чисел. Вычислить разность между максимальным и минимальным из них.

```
Main()
{float a, min, max,res; i:int;
cout<<'n=, a1= , a2= '<<endl;
cin>>n,a1,a2;
if (a1>a2)
{a1=max; a2=min };
else a2=max; a1=min; }
i=3;
while i<100
{cout<<'a', I, '= '<<endl; cin>>a;
if (a<min) min=a;
if (a>max) max=a; i++;
res=max-min;
cout<<res; }
```

Задача 2. Вычислить величину y, равную (n!!)

$$Y = \begin{cases} 1\cdot 2\cdot 3...n, \text{если} & n - \text{нечетное} \\ 2\cdot 4\cdot 6...n, \text{если} & n - \text{четное} \end{cases}$$

```
main()
{int I, fn, n, y;
cout<<n; fn=1; y=1;
if (n % 2 <> 0)
{for (i=1; n;) { fn*=i; } }
for (i=1, fn;)
{y+=fn; } }
else { for (i=2; n;)
if I % 2 == 0 fn*=I else fn=fn*(i+1);
}
for (i=1; fn)
{ y+=fn }
cout<<y; }
```

Задача 3. Вычислить величину y, равную (n!!)

$$Y = \begin{cases} 1\cdot 3\cdot 5...n, \text{если} & n - \text{нечетное} \\ 2\cdot 4\cdot 6...n, \text{если} & n - \text{четное} \end{cases}$$

```
if n%1 <> 0
for (i=1; n; i++;)
y*=y*I;
else for (i=1; n; i+=2;) y:=y*I;
cout<<y<<endl;
```

Задача 3. Дано натуральное число *n*. Вычислить произведение первых *n* сомножителей:

$$\frac{2}{1}\cdot\frac{2}{3}\cdot\frac{4}{3}\cdot\frac{4}{5}\cdot\frac{6}{5}\cdot\frac{6}{7}\cdot...$$

```
{ For (i=2; n, i+=2)
{ P=p*(i/(i+1)); }
For (i=1; n; i+=2)
{P=p*((i+1)/i);}
cout<<p;
```

Задача 4. Вычислить $\sum_{k=1}^{n}\sum_{m=k}^{n}\frac{x+k}{m}$.

```
Main()
{ int k, x, m, n;
float s,rez=rezs=0;
cout<<('n:, x:'); cin>>(n,x);
for (k=1; n;) for (m=k; n;)
{ s:=(x+k)/m;
rez+=s; } rezs:=rezs+rez;
} cout<<rezs:8:3<<\n; }
```

```
   int n, j: integer;
   { cout<<'n';
   cin>>n; cout<<'x';
   cin>>x; r=res=0;
   for (k=1; n) for (m=k, n)
   {s=(x+k)/m;
   r=r+s;} } res+=r; }
cout<<res; }
```

Задача 5. Вычислить $\sum_{i=1}^{n}\sum_{j=1}^{i}\frac{1}{i+2j}$.

```
Main()
{
int I,j,n,s=0;
cin>>n;
for (i=1; n)
for (j=1; I)
s:=1/(i+2*j);
cout<<s;
```

```
{ int i, j, n;
float sum1,sum2;
cout<<'n:'; cin>>n;
sum1=0; sum2=0;
for (i=1; n) for (j=1; i)
sum1=sum1+(1/(i+2*j)); }
sum2= sum1 + sum2; }
cout<<sum2:8:3; }
```

Задача 6. Вычислить $\sum_{k=1}^{10} \frac{\sum_{n=1}^{k} \sin kn}{k!}$.

```
fac=1; s=0;
for (k=1; 10)
{ s=0;
for (n=1, k)
s=s+sin(1+k);
fac*=k;
zn=s/fac;
s=s+zn;}
```

Задача 6. Дано 200 вещественных чисел. Определить, сколько из них больше своих «соседей», т.е. предыдущего и последующего чисел.

```
{ cout<<'a1='; cin>>b;
cout<<'a2='; cin>>c
n=0;
for (i=1; 200)
{ cout<<'a',I,'='; cin>>(a);
if (c>b)and(a<c)
n=n++; b=c; c=a; }
cout<<n; }
```

Задача 7. Дано натуральное число n. Получить сумму тех чисел вида $i^3 - 3*i*n^2 + n$ (i =1, 2, …, n), которые являются утроенными нечетными.

```
Main()
{ int n,I,k,l; S: float;
Cuot<<'Enter n'
Cin>>n
For (i=1, n)
S=sqr(i)*i-3*i+sqr(n)+n
{ If s % 3 = 0
K=s % 3;
If k % <>0 L=l+s; }
cout<<'Result=';
coutl;}
```

Задача 7. Вычислить $\prod_{i,j=1}^{20} \frac{1}{i+j}^{2}$.

```
Main()
{ int I,j;
float P;
For (i=1; 20)
{ p=1;
For (j=1; 20)
P=p+1/(i+Sqr(j)); }
cout<<p; }
```

Задача 8. Дано 200 вещественных чисел. Определить, сколько из них больше своих «соседей», т.е. предыдущего и последующего чисел.

```
Main()
{cout<<‘a1=’; cin>b;
cout<<‘a2=’; cin>>(c);
n=0;
for (i=1 to 200,)
{cout<< ‘a’,I,’=’; cin>>a;
if (c>b) & (a<c) n+=n++;
b=c; c=a; }
cin<<n;}
```

Задача 8. Даны целые числа a, n, $x_1, \ldots, x_n$, ($n > 0$). Определить, каким по счету идет в последовательности $x_1, \ldots, x_n$ член, равный a. Если такого члена нет, ответом должно быть число0.

```
Float main()
{integer a, n, x, i, k;
cout<<‘a, n =’; cin>>a,n;
k=0;
for (i=1; n, ++i)
cout<<‘x i=’; cin>>x;
if x<>a k=k++ else
cout<<‘Это число по счету ’<<k;
if k= n cout<<’0’; } }
```

Задача 9. Даны целые числа a, n, $x_1, \ldots, x_n$, ($n > 0$). Определить, каким по счету идет в последовательности член, равный a.

```
double main()
{ a, n, I, k: int;
cout<<’a = ’<<‘n = ’; cin>>a,n;
k=0; for (i=1, n)
{ cout<<(‘x’,I,’=’); cin>>x;
if x<>a k=k++
else cout<<‘Это число по счету’<<k;
if k==n cout<<‘0’; } }
```

```
{ float x; int S,n,i,a;
cout<<‘n= ’; cin>>n;
cout>>‘a = ’; cin<<a;
S=0; i=1; s=0;
while i<=n
{ cout<<‘x’<<i<<‘=’; cin>>x;
if x<>a s=s++; i=i++ }
if n==s cout<<‘0’ else cout<<s; }
```

Задача 10. Дано 100 вещественных чисел. Определить, образуют ли они возрастающую последовательность.

```
Main()
{real i, s:integer; x, pred;
pred=0; s=0;
for (i=1, 100)
{cout<<'x='; cin>>x;
if x>pred s=s++;
pred=x;}
if s=100 Cout<<' возрастает'
else cout<<'Не возрастает';
return 0; }
```

Задача 10. Даны целое n>0 и последовательность из n вещественных чисел, среди которых есть хотя бы одно отрицательное число. Найти величину наибольшего среди отрицательных чисел этой последовательности.

```
int n,i; float a=max;
cout<<'Enter n'; cin>>n;
cout<<'a1='; cin>>max;
for (i=2; n)
{cout<<'a'<<I<<'='; cin>>a;
if a<0 and if a>max max=a; }
cout<<max;
```

Задача 11. Даны целые числа *m*, *n* (*m* <> 0, *n* <> 0). Получить все их общие делители (как положительные, так и отрицательные).

```
{cout<<'Enter m, n'; cin>>m,n;
if m> n for (i=-n; n)
if ((m % i)==0)and((n % i)==0)
cin>>i;}
```

```
Main()
{cout<<'Enter n, m'; cin>>n, m;
for (i=1, n, m)
if n % i=0 & m %i = 0 cout<<I; }
```

Задача 12. Вычислить $F = \sum_{n=1}^{15} \frac{x^{2n} \sin\left(x^n\right)}{n^2}$.

```
A=1; f=0;
For (i=1; n; ++i) A*=x; }
for (n=1; 15; n++)
{ f+=(sqr(a)Sin(a))/sqr(n); }
```

Задача 12. Вычислить $\sum_{k=1}^{10} \frac{\sum_{n=1}^{k} \sin kn}{k!}$.

```
{int k,n,f; float ch,sum=0; ch=0; f=1;
for (k=1; 10) for (n=1; k)
cout<<'k'<<n<<'='; cin>>k;
ch=ch+sin(k); } f*=k; sum=sum+ch/f; }
cout<<sum; }
```

Задача 13. Даны натуральные числа *n*, *m*, (*m* < *n*), целые числа $a_1, \ldots, a_n$. Получить число отрицательных членов последовательности $a_1, \ldots, a_m$ и число нулевых членов всей последовательности $a_1, \ldots, a_n$.

```
int m,n,p,k,i,a;
{ cout<<'Enter n, m'; cin>>n,m;
k=p=0;
if n>m for (i=1; m)
{cout<<'ai=';cin>>a;
if (a<0) k=k++; if (a=0) p=p++; }
for (i=1; m)
cout<<'ai=';cin>>a;
if (a=0) p=p++; } }
cout<<'отрицательных членов = ',k;
cin>>число нулевых = ', p); }
```

Задача 13. Дано натуральное число *n*. Получить все простые делители этого числа.

```
Main()
{ int I,n;
cout<<'n='; cin>>n;
while i<n
{n= n % I;
if (n+i) % (n+i) == 0 and (n+i) % 1 ==0
cout<<n+I; else i=i++; }
return 0; }
```

Задача 14. Вычислить $\sum_{k=1}^{n}\sum_{m=k}^{n}\frac{x+k}{m}$.

```
Int main (){
for (k=1; n, k+1)
for (m=k; n)
s+=(x+k)/m;
cout<<S;
```

Задача 14. Вычислить $\sum_{k=1}^{n} k^k x^{2k}$.

```
main ()
{int n, k, sp1; float x,p,sum,sp2;
cout<<'n='<<'x='; cin>>x; cin>>n;
p=1; sum=0; sp1=k; sp2=k;
for (k=1; n)
{sp1*=k; sp2=sqr(x)*sp2;
p=p*sp1*sp2; sum+=p; }
cout<<sum; }
```

Задача 15. Вычислить $\sum_{k=1}^{n} k^k x^{2k}$.

```
Main()
{int n, k, sp1; float x,p,sum,sp2;
cout<<'n='<<endl; cout<<'x='<<endl;
cin>>x; cin<<n;
p=1; sum=0; sp1=k; sp2=k;
for (k=1; n,)
{sp1*=k;
sp2=sqr(x)*sp2;
p:=p*sp1*sp2;
sum+=p; }
cout<<(sum);}
```

Задача 15. Дано 10 вещественных чисел. Определить, сколько из них меньше своих «соседей», т.е. предыдущего и последующего чисел.

```
Main()
{Const n=10; int I,f; real X;
cout<<'x1=';  cin>>x;  f=0;
for (i=2; n; i++)
{cout<<'x'<<I<<'='; cin>>x;
if x>x1max=x;
if max>x f=f++;}
cout<<f; }
```

Лабораторная работа №3. Массивы и функции

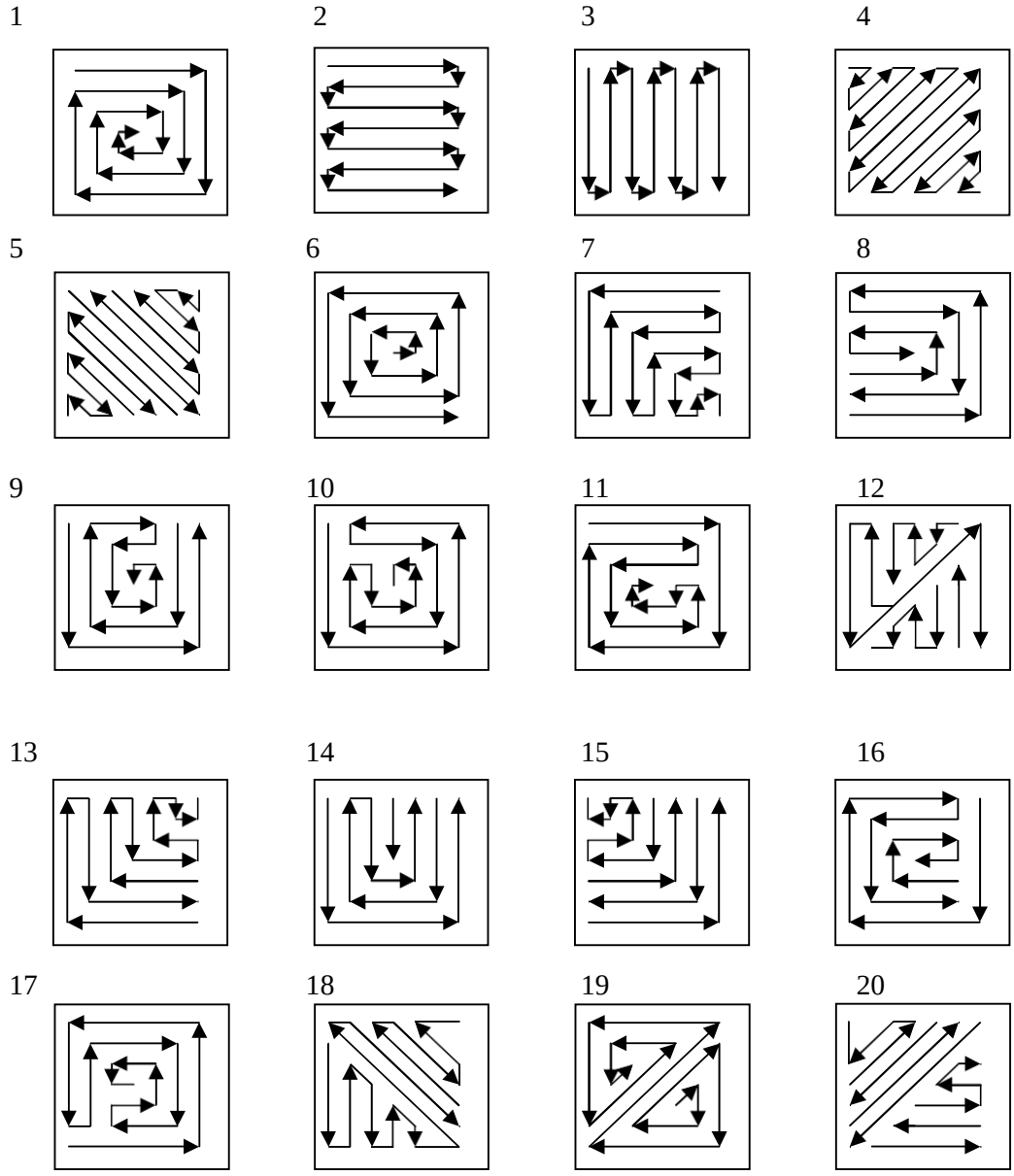
Написать функцию для заполнения квадратной матрицы размерностью $n \times n$

возрастающей последовательностью целых чисел по заданной схеме. Размерность матрицы задается константой и должна находиться в пределах $4 \le n \le 16$. Написать также функции вывода матрицы на экран и функцию нахожденияобратной матрицы, функцию умножения матриц. Показать, что найдена обратная матрица, перемножив ее сисходной. Работу всех функций проиллюстрировать в программе.

Пример заполненной матрицы размером 4×4 по схеме варианта 1:

1	2	3	4
12	13	14	5
11	16	15	6
10	9	8	7

Варианты



Лабораторная работа №4. Работа со строками и файлами

С помощью текстового редактора создать файл text1.txt и заполнить словами из латинских букв (не менее 30 слов).

Файл number1.txt программно заполнить случайными целыми числами, принадлежащими диапазону от -100 до 100. Количество чисел в файле задается с клавиатуры. Количество чисел в каждой строке является произвольным числом из диапазона от 0 до 10 и задается с помощью функции rand(). Вывести числа на консоль, сохранив разбиение на строки, соответствующее файлу number1.txt.

При реализации программы необходимо для заполнения файла number1.txt случайными целыми числами использовать заголовочный файл, содержащий функцию для заполнения файла. При работе с файлом, заполненным словами, можно использовать функции работы со строками. При выводе результата в файл number3.txt сохранить разбиение на строки.

Записать в файл total.txt сначала все элементы файла number2.txt, выровненные по правому краю, а затем все элементы файла text2.txt, выровненные по центру. Переименовать файл total.txt (новое имя файла вводится с клавиатуры).

Вывести на консоль сначала все элементы файла number2.txt, выровненные по правому краю, а затем все элементы файла text2.txt, выровненные по центру.

Варианты

1. Найти сумму максимальных элементов файла number1.txt. Все числа, меньшие первого максимального, вывести в файл number3.txt и на консоль. В конец файла number2.txt добавить слова "Result =", номер строки и максимальное число этой строки, вывести полученный результат на консоль. Вывести все слова файла text1.txt, начинающиеся с гласных букв латинского алфавита. Результатзаписать в файл text2.txt.

2. Подсчитать, число элементов файла number1.txt, кратных 9. Вывести все числа, кратные 9, в файл number3.txt и на консоль. В конец файла number2.txt добавить слова "Result = " и число элементов, вывести полученный результат на консоль.
   Вывести все строки файла text1.txt, начинающиеся с заглавной буквы. Результат записать в файл text2.txt.

3. Подсчитать, сколько раз каждое число встречается в тексте файла number1.txt. Все встречающиеся числа и их количество вывести в файл number3.txt и на консоль. В конец файла number2.txt добавить слова "Result = " и количество различных чисел в тексте, вывести полученный результат на консоль.
   В словах файла text1.txt заменить заданную букву на букву, вводимую с клавиатуры. Результат записать в файл text2.txt.

4. Найти максимальное число в каждой строке файла number1.txt. Вывести в файл number3.txt и на консоль все числа файла number1.txt, стоящие до найденного максимального. В конец файла number2.txt добавить слова "Result = " и найденное максимальное число, вывести полученный резуль- тат на консоль.
   Вывести все слова файла text1.txt, начинающиеся с согласных букв латинского алфавита. Результат записать в файл text2.txt.

5. Найти сумму чисел каждой строки файла number1.txt. Вывести в файл number3.txt и на консоль номер строки и полученную сумму. В конец файла number2.txt добавить слова "Result = " и количество строк файла number1.txt, вывести полученный результат на консоль.
   Вывести все слова файла text1.txt, начинающиеся и оканчивающиеся одной и той же буквой. Результат записать в файл text2.txt.

6. Преобразовать в логарифм все неотрицательные элементы файла number1.txt. Результат вывести в файл number3.txt и на консоль. В конец файла number2.txt добавить слова "Result = " и количество четных неотрицательных элементов, вывести полученный результат на консоль.
   Все слова файла text1.txt, заканчивающиеся на букву, заданную с клавиатуры, вывести в файл text2.txt.

7. Подсчитать факториал каждого положительного элемента файла number1.txt. Полученные элементы вывести в файл number3.txt и на консоль. В конец файла number2.txt добавить слова "Result = " ичисло положительных элементов, вывести полученный результат на консоль.
   Удалить все пустые строки файла text1.txt. Результат записать в файл text2.txt.

8. Найти разность пар соседних элементов файла number1.txt. Результат вывести в файл number3.txt и на консоль. В конец файла number2.txt добавить слова "Result = " и количество элементов, равных нулю, вывести полученный результат на консоль.

   Вывести все слова файла text1.txt, предварительно преобразовав каждое из них по следующему правилу: оставить в слове только первые вхождения каждой буквы. Результат записать в файл text2.txt.

9. Возвести в квадрат все отрицательные элементы файла number1.txt. Результат вывести в файл number3.txt и на консоль. В конец файла number2.txt добавить слова "Result = " и число нечетных отрицательных элементов, вывести полученный результат на консоль.
   Вывести все слова файла text1.txt, которые содержат определенное количество заданной буквы. Результат записать в файл text2.txt.

10. Каждый элемент файла number1.txt заменить на экспоненту в степени, равной этому элементу. Результат вывести в файл number3.txt и на консоль. В конец файла number2.txt добавить слова "Result
    = " и число элементов, больших единицы, вывести полученный результат на консоль.
    Каждое слово файла text1.txt циклически сдвинуть вправо на половину своей длины. Например, сдвиг слова asdhjk приводит к hjkasd. Результат записать в файл text2.txt.

11. Уменьшить каждый элемент строки файла number1.txt на число, соответствующее номеру этой строки. Результат вывести в файл number3.txt и на консоль. В конец файла number2.txt добавить слова "Result = " и количество отрицательных элементов, вывести полученный результат на консоль.
    Вывести различные слова файла text1.txt. Результат записать в файл text2.txt.

12. Извлечь квадратный корень из каждого положительного элемента файла number1.txt. Результат вывести в файл number3.txt и на консоль. В конец файла number2.txt добавить слова "Result = " и число положительных элементов, вывести полученный результат на консоль.
    Удалить из файла text1.txt все подслова, содержащие последовательность трех букв, вводимых с клавиатуры. Результат записать в файл text2.txt.

13. Найти логарифм каждого элемента файла number1.txt, большего единицы. Все элементы, меньшие единицы, вывести в файл number3.txt и на консоль. В конец файла number2.txt добавить слова "Result = " и число элементов, больших единицы.
    Самые длинные и самые короткие слова файла text1.txt вывести в файл text2.txt.

14. Подсчитать произведение отрицательных элементов файла number1.txt. Все неотрицательные элементы вывести в файл number3.txt и на консоль. В конец файла number2.txt добавить слова "Result = " и найденное произведение, вывести полученный результат на консоль.
    Вывести все строки файла text1.txt, начинающиеся со строчной буквы. Результат записать в файл text2.txt.

15. Найти разность элементов соседних строк файла number1.txt. Все полученные элементы вывести в файл number3.txt и на консоль. В конец файла number2.txt добавить слова "Result = " и число элементов кратных 5, вывести полученный результат на консоль.
    Все слова файла text1.txt заменить на обратные (например, kot заменить на tok), вывести в файл text2.txt.

16. Подсчитать произведение четных элементов файла number1.txt. Все нечетные элементы вывести в файл number3.txt и на консоль. В конец файла number2.txt добавить слова "Result = " и найденное произведение, вывести полученный результат на консоль.
    Все слова файла text1.txt, начинающиеся с заданной буквы (вводится с клавиатуры), вывести в файл text2.txt.

17. Умножить каждый элемент файла number1.txt на номер соответствующей ему строки. Результат вывести в файл number3.txt и на консоль. В конец файла number2.txt добавить слова "Result = " и

количество элементов, кратных 11, вывести полученный результат на консоль.
Каждую строку файла text1.txt циклически сдвинуть вправо на n символов. Например, сдвиг строки asdfghjk на 3 символа: hjkasdfg. Результат записать в файл text2.txt.

18. Найти сумму пар соседних элементов файла number1.txt. Результат вывести в файл number3.txt и на консоль. В конец файла number2.txt добавить слова "Result = " и количество четных элементов, вы- вести полученный результат на консоль.
Зашифровать слова в текстовом файле text1.txt по принципу: каждую гласную продублировать, а каждую согласную заменить на следующую согласную в слове. Например, слово "строка" после шифрования будет выглядеть как "тркоосаа". Результат записать в файл text2.txt.
19. Подсчитать произведение элементов каждой строки файла number1.txt. Все числа, кратные 10, вы- вести в файл number3.txt и на консоль. В конец файла number2.txt добавить слова "Result = ", номер строки и найденные произведения. Вывести полученный результат на консоль.
После каждой гласной слова файла text1.txt добавить последовательность символов, заданную с клавиатуры. Результат записать в файл text2.txt.
20. Подсчитать, сколько раз каждое положительное число встречается в каждой строке файла num- ber1.txt. Номер строки, все встречающиеся числа и их количество вывести в файл number3.txt и на консоль. В конец файла number2.txt добавить слова "Result = " и количество строк в тексте, вывести полученный результат на консоль.
Обратить все строки файла text1.txt и вывести их в файл text2.txt.

Лабораторная работа №5. Классы

Часть 1 (до 5 баллов). Реализуйте класс «Линейный однонаправленный список», как показано в заготовке, поясните принцип организации полей и методов в структуре «Узел», классе «Линейный список». В каком доступе находятся поля и методы структуры TNode? Как называется класс/структура, содержащие в качестве поля ссылку или указатель на самого себя? Какие методы класса TList могут быть сделаны константными? Дополните код класса TList деструктором, адекватно удаляющим список. Продемонстрируйте его работу. Вынесите реализацию всех методов класса за пределы класса. Убедитесь в работоспособности класса на примере приведенного кода в теле main().

```
#include <iostream>

using namespace std;


struct TNode //структура для описания узла в списке

{

      int info;
      TNode *next;
      TNode(int x){

            info=x;
            next=0;}

};
```

```
class TList // сам класс «Линейный список»
{
private:
	TNode * head; //голова списка
	TNode * tail; //хвост списка
public:
	TList()
	{
		//ваш код
	}

	bool addToTail(int x){//добавляем элемент в конец списка
		//ваш код
	}

	bool isEmpty(){ // пуст ли список
		//ваш код
	}

	void print() // вывод списка на печать
	{
		//ваш код
	}

	bool delFromEnd()
	{
		//ваш код
	}

	bool addAfterNode(int valueToSearch, int valueToAdd)
		//добавление после заданного узла
	{
		//ваш код
	}
```

```
      bool delNode(int valueToDelete) //удаление заданного узла
      {
            //ваш код
      }
};

int main()
{
TList myList;

if (myList.isEmpty())
      for (int i=0; i<5; i++)
            myList.addToTail(i);

if (!myList.isEmpty()) myList.print();
myList.addAfterNode(0,-1000);

if (!myList.isEmpty()) myList.print();
myList.addAfterNode(2,-222);

if (!myList.isEmpty()) myList.print();
myList.addAfterNode(4,-444);

if (!myList.isEmpty()) myList.print();
myList.delNode(-222);
```

```
if (!myList.isEmpty()) myList.print();


while (!myList.isEmpty())

      {

            myList.delFromEnd();
            myList.print();
      }

return 0;

}
```

Часть 2 (до 5 баллов). В данной лабораторной работе все описания классов должны быть вынесены в отдельный заголовочный файл. Реализация функций-элементов класса должна быть написана в отдельном модуле, а основную программу, иллюстрирующую применение всех методов вашего класса, следует реализовать еще одним модулем. Во всех заданиях предусмотрите конструкторы с аргументами по умолчанию, а также дружественную перегруженную операцию вывода в поток и чтения из потока. Помните, что в каждом классе должны быть предусмотрены константные функции get.

Варианты

1. Создайте класс с именем Complex для выполнения арифметических действий с комплексными числами. Напишите программу для проверки вашего класса. Комплексные числа должны быть представлены в форме RePart+ImPart*i, где i*i = –1. Используйте переменные с плавающей точкой для представления закрытых данных класса.
   Создайте конструктор, деструктор и открытые функции-элементы для следующих действий:
   a) сложение 2 комплексных чисел;
   b) вычитание 2 комплексных чисел;
   c) умножение 2 комплексных чисел.
   Перегрузите операцию вывода в поток для печати комплексного числа в виде |A|exp{i*z}.

2. Создайте класс с именем Rational для выполнения арифметических действий с дробями. Напишите программу для проверки вашего класса. Используйте целые переменные для представления закрытых данных класса – числителя и знаменателя. Создайте функцию-конструктор класса, которая позволяет объекту этого класса принимать начальные значения при его объявлении. Конструктор должен содержать значения по умолчанию на случай отсутствия заданных начальных и должен хранить дроби в сокращенном виде. Создайте открытые функции-элементы для случаев:
   a) сложение 2 дробей (здесь и далее результат должен храниться в сокращенном виде);
   b) вычитание 2 дробей;
   c) перемножение 2 дробей;
   d) деление 2 дробей;
   e) печать дроби в десятичном виде.
   Перегрузите операцию вывода в поток для печати дроби в виде a/b.

3. То же, что и в варианте 2, но для дроби, числитель и знаменатель которой – комплексные числа. Перегруженная операция печати должна выглядеть следующим образом: на экран должна выводиться дробь в виде (ReA+i*ImA)/(ReB+i*ImB).

4. Модифицируйте класс myTime (о котором говорилось на лекциях) так, чтобы включить функцию-элемент tick, которая дает приращение времени, хранящегося в объекте myTime, равное одной секунде. Объект Time должен всегда находиться в непротиворечивом состоянии. Напишите программу для проверки функции-элемента tick в цикле, которая печатала бы время в каком-либо

стандартном формате на каждой итерации цикла и иллюстрировала правильную работу функции-элемента tick. Удостоверьтесь в правильности работы в следующих случаях:

a) приращение с переходом в следующую минуту;
b) приращение с переходом в другой час;
c) приращение с переходом в другой день.

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

5. Модифицируйте класс myDate (о котором говорилось в лекциях) так, чтобы в нем присутствовала функция-элемент nextDay, которая будет увеличивать дату на 1 день. Объект myDate должен всегда находиться в непротиворечивом состоянии. Напишите программу, проверяющую функцию nextDay в цикле и печатающую время в стандартном формате на каждой итерации цикла. Проверьте правильность работы функции в следующих случаях:

   a) приращение с переходом в следующий месяц;
   b) приращение с переходом в следующий год.

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

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

7. Создайте класс Rectangle (прямоугольник). Класс имеет атрибуты length (длина) и width (ширина), каждый из которых по умолчанию равен 1. Он имеет функции-элементы, которые вычисляют периметр и площадь прямоугольника. Он имеет функции записи и чтения для длины и ширины. Функции записи должны проверять, что длина и ширина – числа с плавающей запятой, находящиеся в пределах от 0,0 до 20,0.
   Перегрузите операцию вывод в поток для печати всей информации о прямоугольнике.

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

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

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

10. Создайте класс треугольник, хранящий только декартовы координаты вершин. Конструктор вызывает набор функций, которые принимают 3 группы координат и проверяют, чтобы каждая из координат x и y находилась в первом квадранте в диапазоне от 0,0 до 50,0. Функции также должны проверять, чтобы треугольник не «схлопывался» в прямую линию. Должны быть предусмотрены функции-элементы, вычисляющие длину сторон, периметр и площадь треугольника. Включите функцию, которая определяла бы, не является ли треугольник равнобедренным, равносторонним или прямоугольным.
    Перегрузите операцию вывода в поток для печати всей информации о треугольнике.

11. Создайте класс треугольник, хранящий длины двух сторон и значение угла между ними. Должны быть предусмотрены функции-элементы, вычисляющие длину третьей стороны, значения 2 оставшихся углов, периметр и площадь треугольника. Включите функцию, которая определяла бы, не является ли треугольник равнобедренным, равносторонним или прямоугольным.
    Перегрузите операцию вывода в поток для печати всей информации о треугольнике.

12. Создайте класс прямая призма, хранящий только декартовы координаты вершин основания и высоту призмы. Конструктор вызывает набор функций, которые принимают группы координат и проверяют, чтобы каждая из координат x и y находилась в первом квадранте в диапазоне от 0,0 до 250,0. Должны быть предусмотрены функции-элементы, вычисляющие длину ребер, периметр и площадь основания, а также площадь боковой поверхности, площадь поверхности и объем призмы. Перегрузите операцию вывода в поток так, чтобы она печатала, какая фигура лежит в основании, и ее основные характеристики.

13. Создайте класс пирамида, хранящий только декартовы координаты вершин основания и вершины пирамиды. Конструктор вызывает набор функций, которые принимают группы координаты. Должны быть предусмотрены функции-элементы, вычисляющие длину ребер, периметр и площадь основания, а также площадь боковой поверхности, площадь поверхности и объем пирамиды. Перегрузите операцию вывода в поток так, чтобы она печатала, какая фигура лежит в основании, иее основные характеристики.

14. Создайте класс конус, хранящий только декартовы координаты центра основания, радиус основания и высоту конуса. Должны быть предусмотрены функции-элементы, рассчитывающие периметр и площадь основания, а также площадь боковой поверхности, площадь поверхности и объем конуса.
    Перегрузите операцию вывода в поток для печати всей информации об объекте.

15. Создайте класс усеченный конус, хранящий только декартовы координаты центра основания, радиусы оснований и высоту конуса. Должны быть предусмотрены функции-элементы, рассчитывающие периметр и площадь оснований, а также площадь боковой поверхности, площадь поверхности и объем конуса.
    Перегрузите операцию вывода в поток для печати всей информации об объекте.

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

площадь основания, а также площадь боковой поверхности, площадь поверхности и объем пирамиды.

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

17. Создайте класс "Спортсмен", в котором будут храниться данные о ФИО атлета, представляемой стране, виде спорта, установленных рекордах, завоеванных местах на первенствах страны, континента, мира и олимпиадные достижения. Предусмотрите методы "Установить рекорд", "Получить медаль" (помните, что медали бывают разные...). Предусмотрите также все необходимые методы установки и чтения данных-элементов.

    Перегрузите операцию вывода в поток для вывода информации о спортсмене.

18. Создайте класс «Запись в адресной книге». В нем хранятся фамилия и имя человека, номера телефонов (нескольких, в т.ч. домашних, рабочих и сотовых), район и адрес проживания, e-mail (несколько). Конструктор должен вызывать функцию, считывающую эти данные из текстового файла. Напишите функции-элементы для установки и чтения данных. Предусмотрите метод поиска номеров сотовых телефонов, метод формирования текстового файла с заголовком и подписью (как заготовки письма на электронный адрес).

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

19. Создайте класс матрица размерностью $n \times n$, который хранит только размерность матрицы и максимальное по модулю значение элемента матрицы (и указатель на целое). Конструктор должен вызывать функцию заполнения матрицы случайными числами в заданном диапазоне. Напишите функции-элементы для:
    a) транспонирования матрицы;
    b) умножения матрицы на число;
    c) сложения матриц;
    d) умножения двух матриц.

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

20. Создайте класс правильный многоугольник, который хранит число вершин и их координаты. Конструктор вызывает набор функций, которые проверяют, чтобы число вершин было не менее 3, что-бы многоугольник был правильным, и в случае ошибки устанавливают значения всех вершин в 0.Должны быть предусмотрены функции-элементы, вычисляющие периметр, площадь многоугольника. Должна быть предусмотрена функция-элемент вывода на печать информации о числе сторонмногоугольника. Напишите программу-драйвер, иллюстрирующую применение вашего класса. Перегрузите операцию вывода в поток для печати всей информации об объекте.

Лабораторная работа № 6. Наследование

Создайте иерархию классов, используя наследование. В каждой программе необходимо соблюсти принцип разделения интерфейса и реализации класса (иными словами, не забывайте выделять заголовочные файлы). В каждом варианте необходимо написать программу, иллюстрирующую применение всех методов ваших классов. Прежде чем приступить к написанию программ, продумайте (или уточните у преподавателя), какие необходимы функции в каждом из классов (может, где-то необходимо считать координаты, где-то площади и объемы, а где-то хранить фамилии и года поступления): как в базовом, так и в классах-наследниках. Также продумайте, что следует поместить в закрытые (a, возможно, защищенные) переменные. Предусмотрите возможность переопределения методов базового класса в производном. Приветствуется демонстрация наследования перегруженных операций. Возможно, в некоторых вариантах лучше воспользоваться композицией, чем наследованием.

Варианты

1. Точка -> Квадрат -> Куб.
2. Четырехугольник -> Трапеция -> Параллелограмм.
3. Точка -> Четырехугольник -> Параллелепипед.
4. Точка -> Треугольник -> Треугольная призма.
5. Точка -> Круг -> Сфера.
6. Факультет:
    a) администрация;
    b) преподаватель;
    c) студент.
7. Учащийся в университете:
    a) студент;
    b) аспирант;
    c) слушатель.
8. Студент:
    a) первокурсник;
    b) студент 2-4 курса;
    c) дипломник.
9. Круг -> Конус -> Усеченный конус.
10. Треугольник -> Треугольная пирамида -> Усеченная треугольная пирамида.
11. Прямолинейное движение делится на равномерное и равноускоренное. Равноускоренное в свою очередь может реализовываться свободным падением по вертикали.
12. Криволинейное движение можно разбить на движение по окружности и падение тела, брошенного под углом к горизонту. Из движения тела, брошенного под углом к горизонту, можно выделить свободное падение тела, брошенного горизонтально.
13. Спортсмен -> Легкоатлет -> Спринтер.
14. Точка -> Треугольник -> Равнобедренный треугольник -> Равносторонний треугольник.
15. Транспортное средство:
    a) трамвай;
    b) троллейбус;
    c) автобус.
16. Чемпионат по программированию -> Университетский тур -> Городской тур -> Областной тур.
17. Студент -> Математик -> Математик-программист.
18. Среди накопителей информации можно выделить такие классы, как жесткий диск и флеш-карта. Среди жестких дисков, в свою очередь, можно выделить класс съемных дисков.
19. Точка -> Треугольник -> Прямоугольный треугольник -> Равнобедренный прямоугольный

треугольник.

20. Точка –> Четырехугольник -> Ромб -> Квадрат.

Лабораторная работа № 7. Виртуальные функции и полиморфизм

Часть 1 (до 3 баллов)

Вам даны классы BinaryOperation (бинарный оператор) и Number (число), которые наследуются от базового класса Expression (выражение). Ваша задача реализовать базовый класс Expression так, чтобы не было утечек памяти. Кроме этого подумайте, какие методы стоит сделать виртуальными.

```
#include <cassert> // assert

using namespace std;


struct Expression

{

      // здесь должен быть ваш код

};


struct Number : Expression

{

      Number(double value) : value_(value) {}
      double value() const { return value_; }
      double evaluate() const { return value_; }

private:

      double value_;

};


struct BinaryOperation : Expression

{

enum {

      PLUS = '+',

      MINUS = '-',

      DIV = '/',
      MUL = '*'

      };


BinaryOperation(Expression const *left, int op,

                  Expression const *right):left_(left), op_(op), right_(right)
```

```
        {
        assert(left_ && right_);
        }

~BinaryOperation()
        {
        delete left_;
        delete right_;
        }

Expression const *left() const { return left_; }
Expression const *right() const { return right_; }
int operation() const { return op_; }
```

```
double evaluate() const
      {
      double left = left_->evaluate();
      double right = right_->evaluate();
      switch (op_)
            {
            case PLUS: return left + right;
            case MINUS: return left - right;
            case DIV: return left / right;
            case MUL: return left * right;
            }
      assert(0);
      return 0.0;
      }

private:
      Expression const *left_;
      Expression const *right_;
      int op_;
};
```

Проверьте полученную иерархию на следующем фрагменте кода:

```
//.....................................................................................
Expression * e1 = new Number(1.234);
Expression * e2 = new Number(-1.234);
Expression * e3 = new   BinaryOperation(e1,
                              BinaryOperation::DIV, e2);
cout<<e3->evaluate()<<endl;
//.....................................................................................
```

Часть 2 (до 5 баллов)

Добавьте к иерархии из предыдущего упражнения класс-наследник FunctionCall. FunctionCall должен представлять вызов одной из двух предопределенных математических функций: sqrt — извлечение квадратного корня и abs — вычисление модуля числа. Функция идентифицируется строкой, переданной в качестве параметра в конструктор. Не забудьте, что у функции должен быть аргумент (которым может быть любое выражение Expression)!

```
#include <string> // std::string
#include <cassert>
#include <cmath> // sqrt и fabs
```

```
// эти классы из предыдущего упражнения

struct Expression;
struct BinaryOperation;
struct Number;

struct FunctionCall : Expression

{

      /**

      * @name – это имя функции, возможные варианты
      * "sqrt" и "abs".
      *

      * Объекты, std::string можно
      * сравнивать с C-строками используя
      * обычный синтаксис ==.
      *

      * @arg – выражение-аргумент функции
      */


FunctionCall(/*аргументы*/)

      {

      //реализация

      }

// реализуйте оставшиеся методы из

// интерфейса структуры Expression и не забудьте

// удалить arg_, как это сделано в классе BinaryOperation

//также реализуйте предложенные ниже методы


std::string const & name() const

      {

      // реализация

      }


Expression const *arg() const

      {

      //реализация

      }


~FunctionCall(){/*реализация*/}
```

```
private:

      //а тут должны быть поля для FunctionCall

};
```

Проверьте полученную иерархию на следующем фрагменте кода:

```
//.....
Expression* n32 = new Number(32.0);

Expression* n16 = new Number(16.0);

Expression* minus = new BinaryOperation(n32, BinaryOperation::MINUS, n16);
Expression* callSqrt = new FunctionCall("sqrt", minus);

Expression* n2= new Number(2.0);

Expression* mult = new BinaryOperation(n2, BinaryOperation::MUL, callSqrt);
Expression* callAbs = new FunctionCall("abs", mult);

cout<<callAbs->evaluate()<<endl;

//.....
```

Часть 3 (до 7 баллов)

Создайте иерархию классов, используя наследование от абстрактного базового класса. В каждой программе необходимо соблюсти принцип разделения интерфейса и реализации класса. В каждом варианте необходимо написать программу, иллюстрирующую применение всех методов ваших классов. Прежде чем приступить к написанию программ, продумайте (или
посоветуйтесь с преподавателем), какие необходимы функции в каждом из классов. Также продумайте, что следует поместить в закрытые или защищенные переменные. Особенно обратите внимание на то, какие функции следует сделать чистыми виртуальными. Обязательно предусмотрите виртуальный деструктор! В основной программе обязательно используйте
динамическое связывание (без него работа не принимается).

Варианты

1. Координаты точки на плоскости – это способ описания ее положения в двумерном пространстве. Способов описания может быть много, но мы ограничимся декартовыми, полярными и естественными координатами. Рассмотрите класс «Координата» как базовый, а перечисленные способы описания положения на плоскости реализуйте конкретными классами. Можете также поэкспериментировать с функциями – друзьями классов, которые могут переводить один способ описания в другой.
2. Усложним задачу №1 – перейдем в пространство. Здесь остановимся на декартовых, сферических и цилиндрических координатах. Далее сделайте то же, что и в первой задаче.
3. Пока с утра вы едете в университет, вы являетесь пассажиром. Реализуйте абстрактный базовый класс «Пассажир» и конкретные классы – пассажир с билетом, льготной сезонкой, транспортной картой (помните, что их бывает несколько видов), «заяц». Попробуйте подсчитать, сколько пассажиров каждого «вида» едет в час пик и какова выручка кондуктора с них.
4. Реализуйте иерархию Форма -> Точка -> Круг -> Сфера. Можете попробовать написать для этих классов функции рисования, а не только расчета площади, периметра и т.п.
5. То же, что и в задаче №5, но иерархия Форма -> Точка -> Многоугольник -> Многогранник.

6. Все мы любим отдыхать. Но абстрактное понятие «нерабочий день» может на самом деле оказаться конкретным выходным, праздником или отпуском. Реализуйте такую иерархию. Разумеется, самый большой приоритет имеют выходные – на них могут попастьи праздники, и отпуск.

7. Вспомним основы механики. Наиболее общее понятие «движение» конкретизировалось сначала понятиями прямолинейное и криволинейное движение. Прямолинейное движение в свою очередь еще более сильно конкретизировали: равномерное и неравномерное движение. А уж из неравномерного движения выделяли еще равноускоренное движение. С криволинейным все немного проще – достаточно рассмотреть движение по окружности как частный случай более общего криволинейного движения. Реализуйте иерархию классов, используя абстрактный базовый класс «Движение». Не забудьте про основные характеристики движения!

8. Напишите абстрактный класс «Студент», которому наследуют конкретные классы отличник, хорошист, троечник, должник. Не забудьте успевающим студентам начислить стипендию, а неуспевающим – дату пересдачи долгов. Естественно, количество пересдач ограничено – не более 3. Предусмотрите возможность отчисления неуспевающих студентов или ухода их в академический от- пуск.

9. При слове «Авиабилет» у вас наверняка возникают самые разные ассоциации – ведь это может быть билет на простой междугородний рейс или на международный. Это может быть билет первого класса, бизнес-класса, второго класса… Реализуйте иерархию «Авиабилет» -> «Междугородний» -> «Международный». Помните, что требования на приобретение международного авиабилета более жесткие, значит данных в этом классе (или функций) будет больше.

10. Представьте, что вы только что выиграли чемпионат по программированию. Вы стали абстрактным победителем, чтобы о вас знали как о конкретном человеке, победившем в конкретном чемпионате, надо определить несколько вещей. Во-первых, свои персональные данные (хотя бы ФИО). Во-вторых, вид и тур соревнования: университетский тур, городской, областной, федеральный или международный. Начиная с абстрактного класса «Победитель чемпионата», реализуйте приведенную выше иерархию классов.

11. Попробуйте немного себя в роли бухгалтера, начисляющего зарплату. Само по себе понятие «Зарплата» не особенно конкретное: оно включает и почасовую, и ставочную зарплату, и комиссионные,

12. и процент с продаж (если работа связана с продажей). Реализуйте данные классы.

13. Вычислительное средство само по себе является понятием абстрактным. Оно может представлять из себя бухгалтерские счёты, арифмометр, калькулятор или современную ПЭВМ. Реализуйте такую иерархию.

14. Поговорим о транспортных средствах. На дороге можно встретить грузовой, легковой и пассажирский транспорт. Каждый из этих видов транспорта обладает общими, а также специфическими характеристиками (подумайте, какими). Напишите иерархию классов «Автомобиль» (абстрактный базовый) и далее 3 производных конкретных класса. Подумайте, какие характеристики нужны для вывода на печать, попробуйте также рассчитать средний тормозной путь, грузоподъемность, вместимость соответствующих транспортных средств.

15. Вся компьютерная графика делится на векторную и растровую. Так как сильно отличаются принципы построения рисунков этих видов, можно выделить программы, поддерживающие тот или иной вид графики и сохраняющие рисунки в том или ином формате. Попробуйте написать программу с использованием полиморфизма и абстрактного класса, реализующую данную схему.

16. Не за горами сессия, и скоро такое абстрактное понятие, как «Контроль успеваемости», превратится во вполне реальное понятие зачет или экзамен (в устной или письменной форме, или, быть может,в форме теста). Подумайте, какие элементы и функции входят в абстрактный базовый класс «Контроль успеваемости» и как их реализовать для конкретных производных классов. Напишите про- грамму.
17. Попробуйте выстроить логическую цепочку: Студент университета N -> Студент факультета N -> Студент специальности N -> Студент курса N. Вместо N каждый раз будет нужно подставить название вуза, факультета, специальности, номер курса. Реализуйте цепочку через абстрактные и конкретные классы. Возможно, для конкретной специальности придется ввести новые предметы, а для конкретного курса – количество часов на изучение определенных дисциплин.
18. Вы уже не первый год изучаете языки программирования. Наверняка вы знаете, что их можно подразделить на языки структурного программирования и языки объектно-ориентированного программирования. К первым можно отнести, например, C и Pascal, ко вторым – C++, Java... Реализуй- те эту структуру через абстрактные и конкретные классы.
19. Понятие "здание" относится к абстрактным. Чтобы его конкретизировать, необходимо знать, является ли оно жилым домом, или торговым центром, или административным зданием, либо спортивным (крытый каток, бассейн) или культурным (театр) объектом... Разумеется, у всех них будут ка- кие-то общие характеристики (например, адрес), и у каждого из них – свои особенные характеристики (скажем, у жилого дома – этажность, количество подъездов, наличие газовых коммуникаций, ГВС, наличие и количество лифтов, наличие мусоропровода и т.д., у бассейна – количество акваторий и их размер (например, две 50-метровые акватории по 9 дорожек), количество мест в раздевалке, душ...). Реализуйте иерархию классов.
20. Все основные неорганические вещества на нашей планете можно подразделить на соли, кислоты, основания и оксиды. Естественно, все категории веществ состоят из химических элементов. Сделав класс «Химический элемент» абстрактным базовым, опишите приведенную здесь иерархию. Вспомните, что все эти вещества могут взаимодействовать друг с другом, превращаясь друг в друга по определенным законам.
21. Напишите абстрактный класс «Учащийся». Его конкретными реализациями будут ученик, студент, аспирант. Подумайте, как выстроить подобную иерархию и какими общими и специфическими характеристиками будет обладать каждый класс. Напишите реализацию данной иерархии.

Лабораторная работа № 8. Обработка исключений

В данной лабораторной работе вам нужно написать класс исключения и программу, способную генерировать и обрабатывать определенный вид исключения (программа должна содержать блоки try, catch, точку throw). Позаботьтесь о том, чтобы исключение действительно могло возникнуть, продемонстрируйте работу перехватчика и обработчика исключений.

Варианты

Обработать следующие исключения:

1. Взятие квадратного корня из отрицательного числа.
2. Нехватка памяти при динамическом ее выделении.
3. Попытка записи в файл, открытый только для чтения.
4. Ввод пользователем вещественного числа вместо целого.
5. Ввод пользователем строки вместо числа.

6. Ввод пользователем специального символа вместо целого числа.
7. Взятие натурального логарифма от нуля.
8. Взятие натурального логарифма от отрицательного числа.
9. Ввод пользователем несуществующей даты, например, «31 февраля».
10. Ввод несуществующего времени, например, 56 час. 335 мин. 98 сек.
11. Ввод пользователем отрицательного возраста.
12. Ввод пользователем отрицательной зарплаты.
13. Попытка чтения из несуществующего файла.
14. Попытка записи в несуществующий файл.
15. Неопределенность вида «0/0».
16. Неопределенность вида «0*□».
17. Ввод числа, спецсимволов или русских букв вместо символов латинского алфавита.
18. Ввод числа, спецсимволов или латинских букв вместо символов русского алфавита.
19. Превышение при вводе максимально возможной длины строки.
20. Выход индекса за пределы массива.

5 семестр

Лабораторная работа № 1. Виртуальные функции и полиморфизм

Часть 1 (до 3 баллов)

Вам даны классы BinaryOperation (бинарный оператор) и Number (число), которые наследуются от базового класса Expression (выражение). Ваша задача реализовать базовый класс Expression так, чтобы не было утечек памяти. Кроме этого подумайте, какие методы стоит сделать виртуальными.

```
#include <cassert> // assert
using namespace std;


struct Expression

{

	// здесь должен быть ваш код

};


struct Number : Expression

{

	Number(double value) : value_(value) {}
	double value() const { return value_; }
	double evaluate() const { return value_; }

private:
```

```
      double value_;
};

struct BinaryOperation : Expression
{
enum {
      PLUS = '+',
      MINUS = '-',
      DIV = '/',
      MUL = '*'
      };

BinaryOperation(Expression const *left, int op,
                  Expression const *right):left_(left), op_(op), right_(right)
      {
      assert(left_ && right_);
      }

~BinaryOperation()
      {
      delete left_;
      delete right_;
      }

Expression const *left() const { return left_; }
Expression const *right() const { return right_; }
int operation() const { return op_; }

double evaluate() const
      {
      double left = left_->evaluate();
      double right = right_->evaluate();
      switch (op_)
            {
            case PLUS: return left + right;
            case MINUS: return left - right;
            case DIV: return left / right;
            case MUL: return left * right;
```

```
            }
      assert(0);
      return 0.0;
      }

private:
      Expression const *left_;
      Expression const *right_;
      int op_;
};
```

Проверьте полученную иерархию на следующем фрагменте кода:

```
//.....
Expression * e1 = new Number(1.234);
Expression * e2 = new Number(-1.234);
Expression * e3 = new   BinaryOperation(e1,
                              BinaryOperation::DIV, e2);
cout<<e3->evaluate()<<endl;
//.....
```

Часть 2 (до 5 баллов)

Добавьте к иерархии из предыдущего упражнения класс-наследник FunctionCall. FunctionCall должен представлять вызов одной из двух предопределенных математических функций: sqrt — извлечение квадратного корня и abs — вычисление модуля числа. Функция идентифицируется строкой, переданной в качестве параметра в конструктор. Не забудьте, что у функции должен быть аргумент (которым может быть любое выражение Expression)!

```
#include <string> // std::string
#include <cassert>
#include <cmath> // sqrt и fabs
// эти классы из предыдущего упражнения
struct Expression;
struct BinaryOperation;
struct Number;
struct FunctionCall : Expression
{
      /**
      * @name – это имя функции, возможные варианты
      * "sqrt" и "abs".
```

```
      *
      * Объекты, std::string можно
      * сравнивать с C-строками используя
      * обычный синтаксис ==.
      *
      * @arg – выражение-аргумент функции
      */

FunctionCall(/*аргументы*/)
      {
      //реализация
      }
// реализуйте оставшиеся методы из
// интерфейса структуры Expression и не забудьте
// удалить arg_, как это сделано в классе BinaryOperation
//также реализуйте предложенные ниже методы

std::string const & name() const
      {
      // реализация
      }

Expression const *arg() const
      {
      //реализация
      }

~FunctionCall(){/*реализация*/}

private:
      //а тут должны быть поля для FunctionCall
};
```

Проверьте полученную иерархию на следующем фрагменте кода:

```
//-----------------------------------------------------------------------------
Expression* n32 = new Number(32.0);
```

```
Expression* n16 = new Number(16.0);

Expression* minus = new BinaryOperation(n32, BinaryOperation::MINUS, n16);
Expression* callSqrt = new FunctionCall("sqrt", minus);

Expression* n2= new Number(2.0);

Expression* mult = new BinaryOperation(n2, BinaryOperation::MUL, callSqrt);
Expression* callAbs = new FunctionCall("abs", mult);

cout<<callAbs->evaluate()<<endl;

//.....
```

Часть 3 (до 7 баллов)

Создайте иерархию классов, используя наследование от абстрактного базового класса. В каждой программе необходимо соблюсти принцип разделения интерфейса и реализации класса. В каждом варианте необходимо написать программу, иллюстрирующую применение всех методов ваших классов. Прежде чем приступить к написанию программ, продумайте (или
посоветуйтесь с преподавателем), какие необходимы функции в каждом из классов. Также продумайте, что следует поместить в закрытые или защищенные переменные. Особенно обратите внимание на то, какие функции следует сделать чистыми виртуальными. Обязательно предусмотрите виртуальный деструктор! В основной программе обязательно используйте
динамическое связывание (без него работа не принимается).

Варианты

1. Координаты точки на плоскости – это способ описания ее положения в двумерном пространстве. Способов описания может быть много, но мы ограничимся декартовыми, полярными и естественными координатами. Рассмотрите класс «Координата» как базовый, а перечисленные способы описания положения на плоскости реализуйте конкретными классами. Можете также поэкспериментировать с функциями – друзьями классов, которые могут переводить один способ описания в дру- гой.
2. Усложним задачу №1 – перейдем в пространство. Здесь остановимся на декартовых, сферических ицилиндрических координатах. Далее сделайте то же, что и в первой задаче.
3. Пока с утра вы едете в университет, вы являетесь пассажиром. Реализуйте абстрактный базовый класс «Пассажир» и конкретные классы – пассажир с билетом, льготной сезонкой, транспортной картой (помните, что их бывает несколько видов), «заяц». Попробуйте подсчитать, сколько пассажиров каждого «вида» едет в час пик и какова выручка кондуктора с них.
4. Реализуйте иерархию Форма -> Точка -> Круг -> Сфера. Можете попробовать написать для этих классов функции рисования, а не только расчета площади, периметра и т.п.
5. То же, что и в задаче №5, но иерархия Форма -> Точка -> Многоугольник -> Многогранник.
6. Все мы любим отдыхать. Но абстрактное понятие «нерабочий день» может на самом деле оказаться конкретным выходным, праздником или отпуском. Реализуйте такую иерархию. Разумеется, самый большой приоритет имеют выходные – на них могут попастьи праздники, и отпуск.
7. Вспомним основы механики. Наиболее общее понятие «движение» конкретизировалось сначала понятиями прямолинейное и криволинейное движение. Прямолинейное движение в свою очередь еще более сильно конкретизировали: равномерное и неравномерное движение. А уж из неравно- мерного движения выделяли еще равноускоренное движение. С криволинейным все немного проще – достаточно рассмотреть движение по окружности как частный случай более общего криволинейного движения. Реализуйте иерархию классов, используя абстрактный базовый класс «Движение». Не забудьте про основные характеристики движения!
8. Напишите абстрактный класс «Студент», которому наследуют конкретные классы отличник, хорошист, троечник, должник. Не забудьте успевающим студентам начислить стипендию, а неуспевающим – дату пересдачи долгов. Естественно, количество пересдач ограничено – не более 3. Предусмотрите возможность отчисления неуспевающих студентов или ухода их в академический от- пуск.
9. При слове «Авиабилет» у вас наверняка возникают самые разные ассоциации – ведь это может быть билет на простой междугородний рейс или на международный. Это может быть билет первого класса, бизнес-класса, второго класса… Реализуйте иерархию «Авиабилет»
   -> «Междугородний» -> «Международный». Помните, что требования на приобретение международного авиабилета более жесткие, значит данных в этом классе (или функций) будет больше.

10. Представьте, что вы только что выиграли чемпионат по программированию. Вы стали абстрактным победителем, чтобы о вас знали как о конкретном человеке, победившем в конкретном чемпионате, надо определить несколько вещей. Во-первых, свои персональные данные (хотя бы ФИО). Во-вторых, вид и тур соревнования: университетский тур, городской, областной, федеральный или международный. Начиная с абстрактного класса «Победитель чемпионата», реализуйте приведенную выше иерархию классов.
11. Попробуйте немного себя в роли бухгалтера, начисляющего зарплату. Само по себе понятие «Зарплата» не особенно конкретное: оно включает и почасовую, и ставочную зарплату, и комиссионные, и процент с продаж (если работа связана с продажей). Реализуйте данные классы.
12. Вычислительное средство само по себе является понятием абстрактным. Оно может представлять из себя бухгалтерские счёты, арифмометр, калькулятор или современную ПЭВМ. Реализуйте такую иерархию.
13. Поговорим о транспортных средствах. На дороге можно встретить грузовой, легковой и пассажирский транспорт. Каждый из этих видов транспорта обладает общими, а также специфическими характеристиками (подумайте, какими). Напишите иерархию классов «Автомобиль» (абстрактный базовый) и далее 3 производных конкретных класса. Подумайте, какие характеристики нужны для вывода на печать, попробуйте также рассчитать средний тормозной путь, грузоподъемность, вместимость соответствующих транспортных средств.
14. Вся компьютерная графика делится на векторную и растровую. Так как сильно отличаются принципы построения рисунков этих видов, можно выделить программы, поддерживающие тот или иной вид графики и сохраняющие рисунки в том или ином формате. Попробуйте написать программу с использованием полиморфизма и абстрактного класса, реализующую данную схему.
15. Не за горами сессия, и скоро такое абстрактное понятие, как «Контроль успеваемости», превратится во вполне реальное понятие зачет или экзамен (в устной или письменной форме, или, быть может, в форме теста). Подумайте, какие элементы и функции входят в абстрактный базовый класс «Контроль успеваемости» и как их реализовать для конкретных производных классов. Напишите про- грамму.
16. Попробуйте выстроить логическую цепочку: Студент университета N -> Студент факультета N -> Студент специальности N -> Студент курса N. Вместо N каждый раз будет нужно подставить на- звание вуза, факультета, специальности, номер курса. Реализуйте цепочку через абстрактные и конкретные классы. Возможно, для конкретной специальности придется ввести новые предметы, а для конкретного курса – количество часов на изучение определенных дисциплин.
17. Вы уже не первый год изучаете языки программирования. Наверняка вы знаете, что их можно под- разделить на языки структурного программирования и языки объектно-ориентированного программирования. К первым можно отнести, например, C и Pascal, ко вторым – C++, Java... Реализуй- те эту структуру через абстрактные и конкретные классы.
18. Понятие "здание" относится к абстрактным. Чтобы его конкретизировать, необходимо

знать, является ли оно жилым домом, или торговым центром, или административным зданием, либо спортивным (крытый каток, бассейн) или культурным (театр) объектом... Разумеется, у всех них будут ка- кие-то общие характеристики (например, адрес), и у каждого из них – свои особенные характеристики (скажем, у жилого дома – этажность, количество подъездов, наличие газовых коммуникаций, ГВС, наличие и количество лифтов, наличие мусоропровода и т.д., у бассейна – количество акваторий и их размер (например, две 50-метровые акватории по 9 дорожек), количество мест в раздевалке, душ...). Реализуйте иерархию классов.

19. Все основные неорганические вещества на нашей планете можно подразделить на соли, кислоты, основания и оксиды. Естественно, все категории веществ состоят из химических элементов. Сделав класс «Химический элемент» абстрактным базовым, опишите приведенную здесь иерархию. Вспомните, что все эти вещества могут взаимодействовать друг с другом, превращаясь друг в друга по определенным законам.
20. Напишите абстрактный класс «Учащийся». Его конкретными реализациями будут ученик, студент, аспирант. Подумайте, как выстроить подобную иерархию и какими общими и специфическими характеристиками будет обладать каждый класс. Напишите реализацию данной иерархии.

Лабораторная работа № 2. Обработка исключений

В данной лабораторной работе вам нужно написать класс исключения и программу, способную генерировать и обрабатывать определенный вид исключения (программа должна содержать блоки try, catch, точку throw). Позаботьтесь о том, чтобы исключение действительно могло возникнуть, продемонстрируйте работу перехватчика и обработчика исключений.

Варианты

Обработать следующие исключения:

1. Взятие квадратного корня из отрицательного числа.
2. Нехватка памяти при динамическом ее выделении.
3. Попытка записи в файл, открытый только для чтения.
4. Ввод пользователем вещественного числа вместо целого.
5. Ввод пользователем строки вместо числа.
6. Ввод пользователем специального символа вместо целого числа.
7. Взятие натурального логарифма от нуля.
8. Взятие натурального логарифма от отрицательного числа.
9. Ввод пользователем несуществующей даты, например, «31 февраля».
10. Ввод несуществующего времени, например, 56 час. 335 мин. 98 сек.
11. Ввод пользователем отрицательного возраста.
12. Ввод пользователем отрицательной зарплаты.
13. Попытка чтения из несуществующего файла.
14. Попытка записи в несуществующий файл.
15. Неопределенность вида «0/0».
16. Неопределенность вида «0*□».
17. Ввод числа, спецсимволов или русских букв вместо символов латинского алфавита.
18. Ввод числа, спецсимволов или латинских букв вместо символов русского алфавита.
19. Превышение при вводе максимально возможной длины строки.
20. Выход индекса за пределы массива.

Лабораторная работа № 3. Шаблоны

Упражнение 1. (1 балл) Реализуйте шаблонную версию класса Array. Список всех операций, которые должен поддерживать класс Array, приведен в шаблоне кода. Проверьте написанный код на типах int, double, string, char*, пользовательском типе Student (для студента храните имя и номер зачетки, а также определите необходимые методы — подумайте над конструкторами и деструкторами). Шаблон класса Array должен успешно работать с данными любого типа.

```
#include <cstddef>

using namespace std;


template <typename T>

class Array

{ private:

      T* myArray;
      size_t n;

public:

      // Список операций:

explicit Array(size_t size = 0, const T& value = T())

      // конструктор класса, который создает

      // Array размера size, заполненный значениями

      // value типа T. Считайте что у типа T есть

      // конструктор, который можно вызвать без

      // без параметров, либо он ему не нужен.

      {}

Array(const Array & mas)

      // конструктор копирования, который создает

      // копию параметра. Считайте, что для типа

      // T определен оператор присваивания.

      { }

~Array()

      // деструктор, если он вам необходим.

      {}

Array& operator=(const Array& mas)

      // оператор присваивания.
```

```
      {}
      //
      // две версии оператора доступа по индексу.
T& operator[](size_t idx)
      {}
const T& operator[](size_t idx) const
      {}
size_t size() const {}
};
```

Упражнение 2. (1 балл) В предыдущей версии предполагается, что для типа T определен оператор присваивания или он ему не нужен (например, для примитивных типов он не нужен). При создании шаблонных классов контейнеров (вроде Array и не только) разумно стараться минимизировать требования к типам шаблонных параметров. Поэтому усложним задачу, реализуйте класс Array не полагаясь на то, что для типаT определен оператор присваивания.

Подсказка: возможно, понадобится placement new и явный вызов деструктора, чтобы создавать и уничтожать объекты, аллоцировать правильно выровненную память можно с помощью `new char[N * sizeof(T)]`, где N – количество элементов массива.

```
#include <cstddef>

template <typename T>
class Array
{ private:
      T* myArray;
      size_t n;
public:
      // Список операций:
explicit Array(size_t size=0, const T& value=T())
      // конструктор класса, который создает
```

```
      // Array размера size, заполненный значениями
      // value типа T. Считайте что у типа T есть
      // конструктор, который можно вызвать без
      // без параметров, либо он ему не нужен.
      {}
Array(const Array & mas)
      // конструктор копирования, который создает
      // копию параметра. Считайте, что для типа
      // T определен оператор присваивания.
      { }
~Array()
      // деструктор, если он вам необходим.
      { }
Array& operator=(const Array& mas)
      // оператор присваивания.
      { }
      // две версии оператора доступа по индексу.
T& operator[](size_t idx)
      { }
const T& operator[](size_t idx) const
      { }
size_t size() const {}};
```

Упражнение 3. (1 балл) Шаблонные классы можно наследовать. Реализуйте шаблонную структуру ValueHolder с одним типовым параметром T, унаследованную от интерфейса ICloneable. Интерфейс ICloneable содержит всего один виртуальный метод ICloneable* clone() const, который должен вернуть указатель на копию объекта, на котором он был вызван (объект должен быть создан в куче). Шаблон ValueHolder, как говорит его название, хранит всего одно значение (назовите его data_) типа T (для типа T определен конструктор копирования). Не делайте поле data_ закрытым (поэтому в данном случае мы явно пишем, что ValueHolder должна быть структурой).

```
struct ICloneable
{
      virtual ICloneable* clone() const = 0;
      virtual ~ICloneable() { }
```

```
};

// Шаблон ValueHolder с типовым параметром T,
// должен содержать одно открытое поле data_
// типа T.

// В шаблоне ValueHolder должен быть определен
// конструктор от одного параметра типа T,
// который инициализирует поле data_.
//
// Шаблон ValueHolder должен реализовывать
// интерфейс ICloneable, и возвращать указатель
// на копию объекта созданную в куче из метода
// clone.
```

Упражнение 4. (1 балл) Реализуйте функцию копирования элементов copy_n из массива источника типа U* в целевой массив типа T*, где T и U произвольные типы, для которых определено преобразование из U в T. На вход функция принимает два указателя и количество элементов, которые необходимо скопировать.

Пример вызова функции copy_n:

```
int ints[] = {1, 2, 3, 4};
double doubles[4] = {};
copy_n(doubles, ints, 4);
// теперь в массиве doubles содержатся
// элементы 1.0, 2.0, 3.0 и 4.0

#include <cstddef>
      // Параметры функции copy_n идут в следующем
      // порядке:
      // 1. целевой массив
      // 2. массив источник
      // 3. количество элементов, которые нужно
      // скопировать
      //
      // Вам нужно реализовать только функцию copy_n,
```

```
    // чтобы ее можно было вызвать так, как показано
    // в примере.
```

Упражение 5. (1 балл) Реализуйте шаблонную функцию minimum, которая находит минимальный элемент, который хранится в экземпляре шаблонного класса Array, при этом типовой параметр шаблона Array может быть произвольным. Чтобы сравнивать объекты произвольного типа, на вход функции также будет передаваться компаратор, в качестве компаратора может выступать функция или объект класса с перегруженным оператором «круглые скобки» "()".

Примеры вызова функции minimum:

```
bool less(int a, int b) { return a < b; }

struct Greater
{ bool operator()(int a, int b) { return b < a; } };

Array<int> ints(3);
ints[0] = 10;
ints[1] = 2;
ints[2] = 15;

int min = minimum(ints, less); // в min должно попасть 2
int max = minimum(ints, Greater()); // в max должно попасть 15
```

```
#include <cstddef>
template <typename T>
class Array
{
public:
    explicit Array(size_t size = 0, const T& value = T());
    Array(const Array& other);
    ~Array();
    Array& operator=(Array other);
    void swap(Array &other);
    size_t size() const;
    T& operator[](size_t idx);
    const T& operator[](size_t idx) const;
private:
    size_t size_;
    T *data_;
```

```
};

// Ваш код
```

Упражнение 6. (2 балла) Шаблонный класс Array может хранить объекты любого типа, для которого определён конструктор копирования, в том числе и другой Array, например, Array< Array<int> >. Глубина вложенности может быть произвольной. Напишите шаблонную функцию (или несколько) flatten, которая принимает на вход такой "многомерный" Array неизвестной заранее глубины вложенности и выводит в поток out через пробел все элементы, хранящиеся на самом нижнем уровне.

Примеры работы функции flatten:

```
Array<int> ints(2, 0);

ints[0] = 10;

ints[1] = 20;

flatten(ints, std::cout); // выводит на экран строку "10 20"
Array< Array<int> > array_of_ints(2, ints);
flatten(array_of_ints, std::cout);

// выводит на экран строку "10 20 10 20"

Array<double> doubles(10, 0.0);
flatten(doubles, std::cout);

// работать должно не только для типа int
```

Примечание: лидирующие и завершающие пробельные символы будут игнорироваться, т. е. там где ожидается "10 20" будет так же принят, например, вариант " 10 20 ", но не вывод "1020".

Подсказка: шаблонные функции тоже можно перегружать, из нескольких шаблонных функций будет выбрана наиболее специфичная.

```
#include <iostream>

// Весь вывод должен осущствляться в поток out,

// переданный в качестве параметра.

//

// Можно заводить любые вспомогаетльные функции,

// структуры или даже изменять сигнатуру flatten,

// но при этом все примеры вызова из задания должны

// компилироваться и работать.
```

Упражнение 7. (1 балл) Выше вы реализовали простой шаблон ValueHolder, в этом задании мы используем его чтобы написать класс Any (интересно, что не шаблонный), который позволяет хранить значения любого типа! Например, вы сможете создать массив объектов типа Any, и сохранять в них int-ы, double-ы или даже объекты Array. Подробности в шаблоне кода.

Подсказка: в нешаблонном классе Any могут быть шаблонные методы, например, шаблонный конструктор.

```
struct ICloneable;

// Поле data_ типа T в классе ValueHolder

// открыто, к нему можно обращаться


template <typename T>

struct ValueHolder;

// Это класс, который вам нужно реализовать


class Any

{

public:

// В классе Any должен быть конструктор, который можно вызвать

// без параметров, чтобы работал следующий код:

// Any empty; // empty ничего не хранит


// В классе Any должен быть шаблонный конструктор от одного

// параметра, чтобы можно было создавать объекты типа Any,

// например, следующим образом:

// Any i(10); // i хранит значение 10


// Деструктор: выделенные ресурсы нужно освободить.


// В классе Any также должен быть конструктор копирования (вам

// поможет метод clone интерфейса ICloneable)


// В классе должен быть оператор присваивания и/или шаблонный

// оператор присваивания, чтобы работал следующий код:

// Any copy(i); // copy хранит 10, как и i

// empty = copy; // empty хранит 10, как и copy

// empty = 0; // а теперь empty хранит 0


// Чтобы получать хранимое значение, определите в классе Any
```

```
// шаблонный метод cast, который возвращает указатель на

// хранимое значение, или нулевой указатель в случае

// несоответствия типов или если объект Any ничего не хранит:

// int *iptr = i.cast<int>(); // *iptr == 10

// char *cptr = i.cast<char>(); // cptr == 0,

// // потому что i хранит int, а не char

// Any empty2;

// int *p = empty2.cast<int>(); // p == 0

// При реализации используйте dynamic_cast.

};
```

Упражнение 8. (1 балл) В качестве упражнения на частичную специализацию шаблонов классов вам предлагается реализовать простой шаблон SameType. Этот шаблон не содержит никаких методов, а только одно **статическое константное поле типа bool** с именем value. Шаблон принимает два типовых параметра, и если два типовых параметра шаблона являются одним и тем же типом, то статическое поле value должно хранить значение true, в противном случае – значение false.

Примеры:

```
struct Dummy { };

typedef int type;

std::cout << SameType<int, int>::value << std::endl;

// выведет 1, т. е. true

std::cout << SameType<int, type>::value << std::endl;

// 1, type == int

std::cout << SameType<int, int&>::value << std::endl;

// 0, int и ссылка на int - различные типы

std::cout << SameType<Dummy, Dummy>::value << std::endl; // 1
std::cout << SameType<int, const int>::value << std::endl;

// 0, const - часть типа


// Определите шаблон SameType с двумя типовыми параметрами.

// В шаблоне должна быть определена одна статическая константа // типа bool с именем
value
```

Упражнение 9. (1 балл) Реализуйте функцию array_size, которая возвращает размер массива, переданного в качестве параметра. Функция должна работать только для массивов! Т. е. если функции передать указатель, должна произойти ошибка компиляции.

Примеры:

```
int ints[] = {1, 2, 3, 4};

int *iptr = ints;

double doubles[] = {3.14};
array_size(ints); // вернет 4
array_size(doubles); // вернет 1

array_size(iptr); // тут должна произойти ошибка компиляции
```

Подсказка: можно организовать передачу в функцию массивов только заданного размера (передача массива по ссылке), совместите его с вашими знаниями о шаблонах.

Лабораторная работа № 4. Обобщенные контейнеры STL

Часть 1. Задачи с **acm.timus.ru (до 10 баллов)**

1563. Баяны (подсказка: удобно воспользоваться контейнером set<string>)

1496. Спамер (подсказка: контейнер map<string, int>)

1161. Stripies

Два полосатика объединяются в одного с весом $2\sqrt{m_1 m_2}$, если получившегося полосатика объединить с каким-то еще, то его вес будет $2\sqrt{m_3 * 2\sqrt{m_1 m_2}}$ и т.д. Чтобы получить минимальный результирующий вес полосатика, надо извлекать корень как можно большее количество раз, выбирая каждый раз двух самых тяжелых полосатиков.

Но для чисел m_1 и m_2 число $\overline{\sqrt{m}}_1 m_2$ является средним геометрическим, а его величина лежит в диапазоне между m_1 и m_2. Упорядочив полосатиков по весу и проводя слияние двух самых тяжелых, результирующий полосатик будет тяжелее, чем любой из оставшихся. А значит его можно поместить в конец списка и далее его объединение с самым тяжелым из оставшихся. Действия можно продолжать, пока не останется один полосатик.

Подсказка: задача легко решается с использованием multiset<double>. Multiset хранит данные в отсортированном виде. При стандартном способе сортировки самые тяжелые полосатики оказываются в конце multiset'a, далее останется вспомнить про реверсный итератор, который можно установить на последний элемент в мультисете... Будьте осторожны с итераторами, особенно если надумаете удалять полосатиков из multiset!

2002. Тестовое задание.

Удобно пользоваться map<string, string> для хранения логинов-паролей и set<string> для учета вошедших в сеть пользователей.

1795. Мужья в магазине.

Здесь придется воспользоваться несколькими структурами данных. Например, подойдет map<string, int> для хранения информации о товаре и его количестве, а мужей в магазине можно выстроить в очередь (или дек, или м.б. list... Только надо учесть, что в очереди надо хранить пару значений — наименование товара и количество, заказанное к покупке, т. е. будет queue <pair<string,int> >). Далее надо просто аккуратно запрограммировать и протестировать написанное.

Часть 2 (до 5 баллов)

Используя обобщенные контейнеры и итераторы STL C++, решите следующие задания. Приветствуется создание собственных контейнеров на базе обобщенных.

Варианты

1. Подсчитайте в заданном файле text1.txt количество неповторяющихся слов и выведите их в файл text2.txt.
2. Подсчитайте в заданном файле text1.txt все повторяющиеся слова и выведите их в файл text2.txt.
3. Определите, какие буквы алфавита не встречаются ни разу в заданном текстовом файле text1.txt, и выведите их в файл text2.txt.
4. В заданном текстовом файле text1.txt хранятся пары значений ФИО – номер телефона. При этом ФИО может повторяться несколько раз. Организуйте text2.txt в следующем виде: ФИО – список всех его телефонов. Отсортируйте по ФИО по возрастанию.
5. Дан текстовый файл text1.txt. Требуется отсортировать все входящие в него слова по алфавиту и записать результат в файл text2.txt.
6. Дан текстовый файл text1.txt. Исключите из него все дубликаты слов. Результат запишите в text2.txt.
7. Дан текстовый файл text1.txt. Найдите самые часто повторяющиеся в нем слова и запишите их в файл text2.txt.
8. Даны текстовые файлы text1.txt и text2.txt. Найдите все общие слова.
9. Дан текстовый файл text1.txt. Требуется записать его содержимое в файл text2.txt в обратном по- рядке.
10. В файле text1.txt имеется список процессов Windows с указанием количества памяти, которую они занимают. В файле text2.txt имеется список процессов, которые мы дополнительно запускаем, так- же
    с указанием количества занимаемой памяти. Запишите в файл text3.txt общий список процессов, отсортированный по количеству занимаемой памяти в порядке убывания
    Имеется текстовый файл text1.txt со следующей структурой: в каждой строке находится запись вида: «фамилия студента» – «предмет». Названия предметов и фамилии могут повторяться. По заданной фамилии выведите все изучаемые им предметы.
11. Имеется текстовый файл text1.txt со следующей структурой: в каждой строке находится запись вида: «год» – «наименование статьи» – «вид публикации». Годы и виды публикации (тезисы, статья, монография…) могут повторяться в любом порядке. Отсортируйте информацию по принципу: самый ранний год, потом все монографии, затем статьи в реферируемых журналах, потом статьи и тезисы. Такую сортировку произведите для каждого года.
12. Дан текстовый файл text1.txt со следующей структурой: в каждой строке файла находится запись вида: «фамилия» – «имя» – «отчество» – «год рождения». Запишите в файл text2.txt количество однофамильцев, в text3.txt – всех ровесников (год задается с клавиатуры).
13. Дан файл text1.txt со следующей структурой: в каждой строке находится запись вида: «исполнитель» – «композиция» – «альбом» – «год выхода записи». В файл text2.txt запишите все композиции одного исполнителя (можно задать с клавиатуры какого). В файл text3.txt записать все композиции из одного альбома.

14. В заданном текстовом файле text1.txt хранятся строки вида: имя потока – приоритет – время выполнения. Определить, в каком порядке завершатся потоки.

15. Дан текстовый файл text1.txt со следующей структурой: в первой строке указано число n, меньшее числа строк в файле. В остальных строках записаны имена. Постройте программу-считалочку, ко- торая выбрасывает каждый раз по одному имени на позиции n, если доходит до конца списка, то продолжает счет с начала. В файл text2.txt записать последовательность «выброса» имен и оставшегося игрока.

16. Дан текстовый файл text1.txt со следующей структурой: в каждой строке находится запись вида:

17. «масть карты» – «старшинство». Количество строк в файле не превосходит 8. Определите, сколько карт каждой масти на руках у игрока. Найдите количество карт одного достоинства для каждого вида старшинства карты. Результат запишите в файл text2.txt.

18. Дан текстовый файл text1.txt со следующей структурой: в каждой строке находится запись вида:

    «масть карты» – «старшинство». В файле перечислена вся колода в произвольном порядке (52 карты…). Требуется отсортировать колоду по принципу следования мастей: червы, трефы, бубны, пики. Внутри каждой масти отсортировать карты по старшинству по возрастанию.

19. В текстовом файле text1.txt хранится информация следующего вида: количество человек в очереди в кассу, количество касс, время обслуживания в каждой кассе. Определить, за какое время очередь будет обслужена.

20. Имеется текстовый файл text1.txt со следующей структурой: в каждой строке файла записаны пары

21. «слово» – «синоним». Заменить в заданном текстовом файле text2.txt все слова, встречающиеся в text1.txt, на синонимы.

Лабораторная работа № 5. Программирование визуального интерфейса в C++

Примечение: При выполнении этой работы вам поможет книга «C/C++ и MS Visual C++ 2008 для начинающих» (Пахомов Б.И.) – там есть описание и примеры работы с элементами графического интерфейса. Также полезный ресурс https://msdn.microsoft.com/ru-ru/library/ms123401.aspx — библиотека MSDN (совет — если нужна подробная справка по некоторым классам или методам. Пространствам имен, наберите в поисковике имя этого класса/метода/пространства имен и слово msdn — поисковик отправит вас на русскоязычную страницу библиотеки MSDN).

Рассмотрим теперь, как можно создавать приложения с графическим интерфейсом пользователя. В Visual C++ такие приложения можно создавать, если самостоятельно программировать win API, работать с библиотекой MFC, либо создавать приложения windows forms. Первый подход на практике довольно сложен, хотя и предполагает детальное понимание «внутренней кухни» программы. Библиотека MFC является мощным средством для разработки гибких приложений, обладающих большими возможностями. Однако самый простой способ ознакомиться с программированием графического интерфейса предоставляют windows application forms.

Рассмотрим некоторые особенности программирования графического интерфейса на простом примере $sin^2(a+b^3)\sqrt{\frac{e^{a^2-9,4}}{(a+b)^3}}$: выполним расчет предложенной формулы в оконном приложении.

Создадим новый проект в Visual C++: **File -> New** -> **Project.** В появившемся диалоговом окне следует выбрать тип проекта CLR, установленный шаблон Windows Forms Application, указать имя проекту (которые, кстати, в VS называются решениями — solution) и назначить рабочую папку.

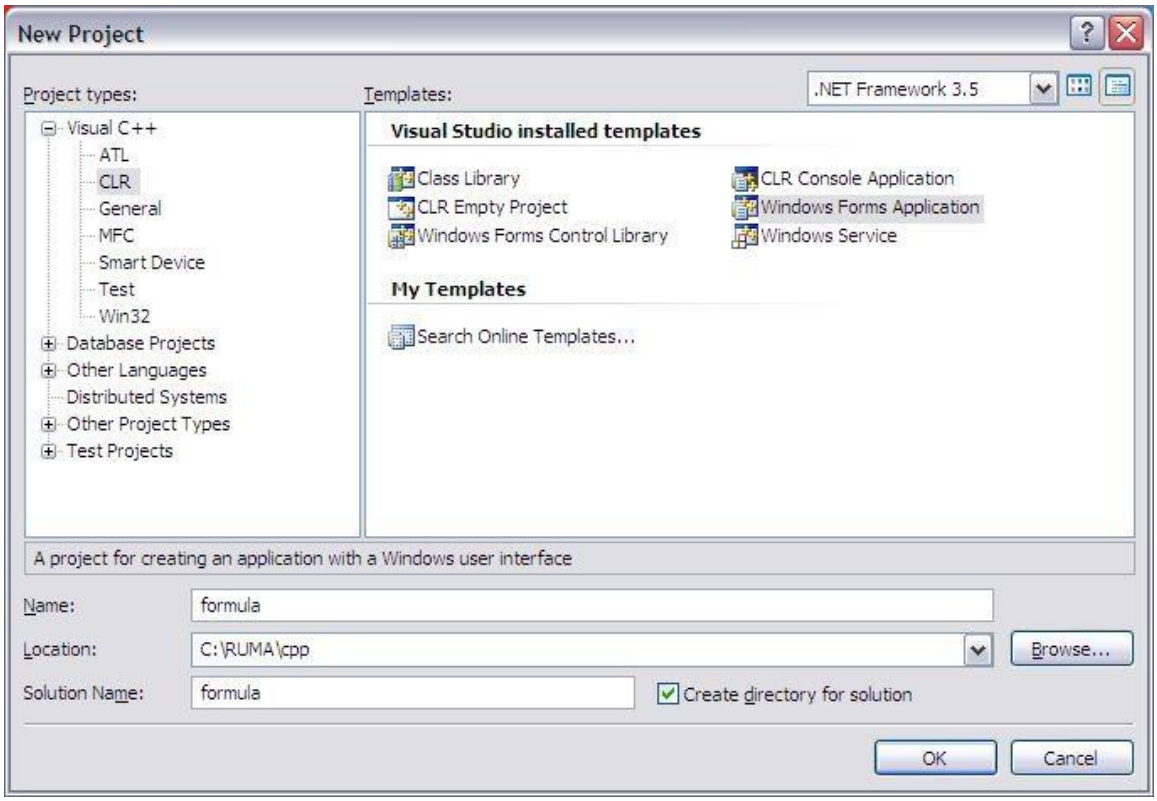


Рис. 1. Диалоговое окно создания нового проекта MS VC++

После этого будет доступно окно редактора с заготовкой формы:

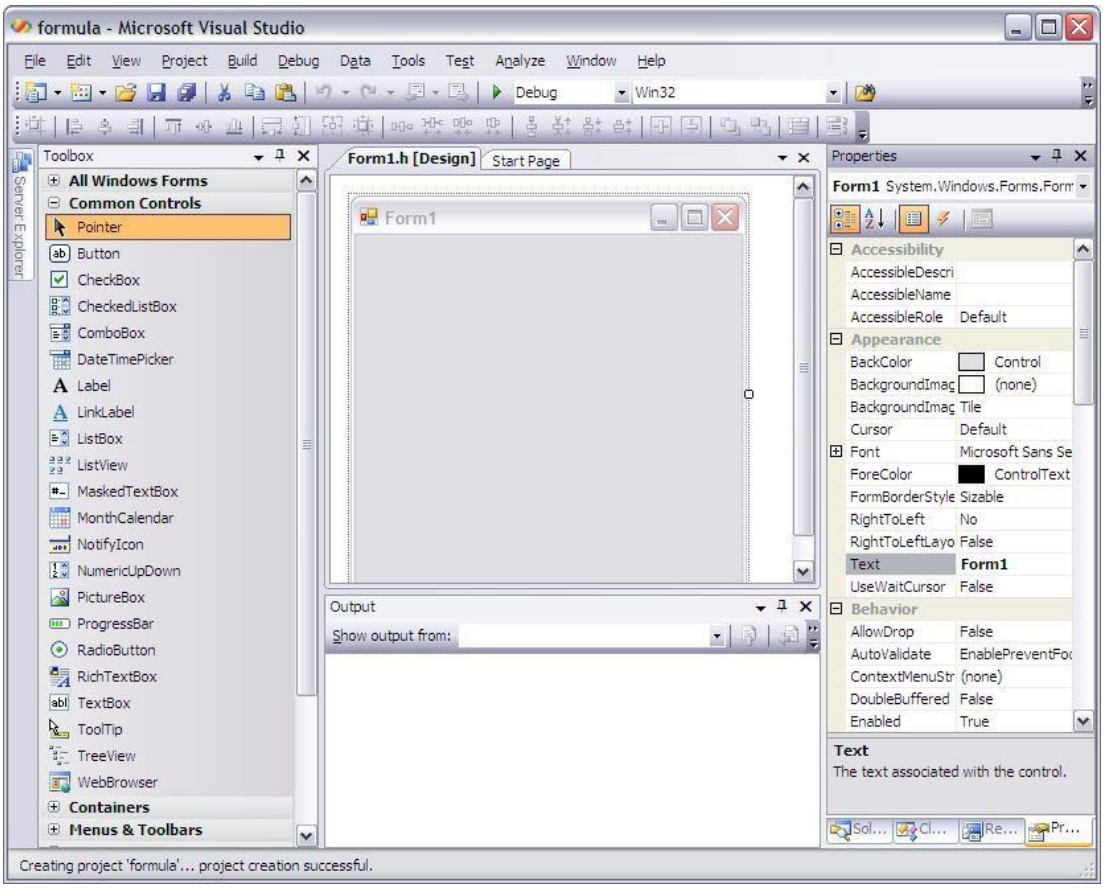


Рис. 2. Внешний вид проекта в IDE MS VC++ в режиме дизайна

Форма, собственно говоря,и будет содержать все необходимые элементы графического интерфейса — сюда можно помещать кнопки, текстовые поля, выпадающие списки, радиокнопки, флажки, метки, компоненты для графического содержимого... Изначально форма открыта в режиме дизайна, т.е. можно формировать интерфейс и сразу же его просматривать.

Панель инструментов слева (**Toolbox**) содержит компоненты, которые можно поместить на форму, и при помощи которых можно моделировать визуальный интерфейс программы любого устраивающего вас уровня сложности. На первых порах нам понадобятся компоненты со вкладки **Common Controls** (обычные элементы управления). Если панель инструментов не видна, ее можно открыть, выбрав пункт меню **View → Toolbox** (или сочетание клавиш **Ctrl+Alt+X).**

Панель Свойств и Событий справа **(Properties)** содержит перечень возможных настраиваемых атрибутов для каждого компонента (например, цвет и размеры формы, текст в заголовке формы и т.п.). Если на- жать на панели **Properties** на значок с молнией, то можно переключиться на панель событий, доступных для данного компонента. Под событием будем понимать действие, производимое пользователем (или системой) над компонентом. Например, события — щелчок мыши по кнопке, выбор пункта в выпадающем списке, щелчок правой клавишей мыши и т.д. Если рядом с именем события в панели **Properties** дважды щелкнуть мышью (или позиционировать там курсор и нажать **Enter**), Visual C++ создаст обработчик события — метод, содержимое которого надо написать самостоятельно. Этот метод будет выполняться каждый раз, когда наступает событие, обработчик для которого мы только что создали. При этом вам будет доступен не только режим работы с дизайном формы, но и режим редактирования кода в h-файле. Отметим, что при создании приложения с графическим интерфейсом часть кода заполняется автоматически средой. Итак,

если создать обработчик события **Click** для формы, то в заголовочном файле формы атоматически появится заготовка вида

```
private: System::Void Form1_Click(System::Object^ sender, System::EventArgs^ e) {

}
```

Form1_Click – название обработчика события, метод принимает два аргумента — указатель на источник события (в данном случае, им будет форма), и вспомогательные параметры события. В обработчике, например, можно написать, чтобы по щелчку мыши в заголовок формы выводился некоторый текст:

```
private: System::Void Form1_Click(System::Object^ sender, System::EventArgs^ e) {

this -> Text = "Обработчик события";

}
```

Если теперь запустить сборку и выполнение решения (**F5**), то щелкая мышью по форме, можно убедиться, что заголовок окна изменяет свое название.

Вернемся к простому проекту вычисления значения по формуле. Параметры a и b должны откуда-то вводиться, результат надо куда-то выводить, считать результат надо «по команде» (т.е. в обработчике события), а еще неплохо было бы показать пользователю формулу, по которой идет расчет. Очевидно, что параметры a и b удобно будет задавать в текстовых полях (2 компонента **TextBox**), эти поля надо подписать (2 метки — **Label**), результат можно выводить либо в метку (**Label**, чем мы и воспользуемся), либо еще в одно подписанное текстовое поле. Вычисление должно производиться по щелчку по кнопке (**Button**), а картинка с формулой будет отображаться в соответствующей области **PictureBox**.

Компоненты на форму устанавливаются очень просто — надо выбрать соответствующий компонент в палитре и «растянуть» его на форме (если надо установить пользовательский размер), либо просто щелкнуть мышью по форме. Установим компоненты в форму. Получится что-то наподобие:

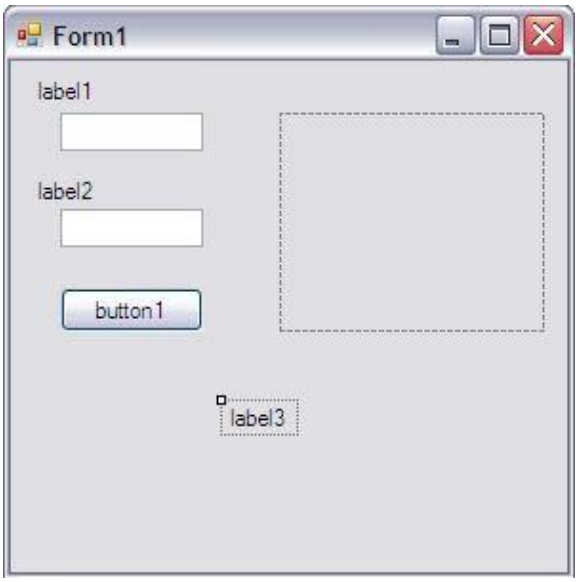


Рис. 3. Внешний вид разрабатываемой формы

Теперь поработаем над изменением некоторых свойств компонентов. Заголовок формы должен быть более информативным (например, расчет значения формулы), метки label1, label2, label3 должны содержатьтекст «Параметр a», «Параметр b», «Результат». Кнопка должна называться «Вычислить», а в PictureBoxнадо отобразить картинку с формулой.

Итак, устанавливаем значения свойств, как показано в таблице 1.

Значения свойств компонентов

Компонент	Свойство	Значение
label1	Text	Параметр a
label2	Text	Параметр b
label3	Text	Результат
button1	Text	Вычислить
pictureBox1	Image	Выбираем файл с картинкой, например, formula_pict.jpg
Form1	Text	Вычисление значения

Теперь, если запустить проект, вам будет доступна форма, в которой уже можно вводить различные значения параметров, но вычисление пока еще не происходит.

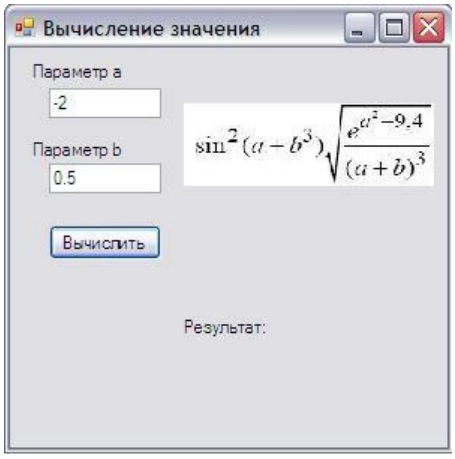


Рис. 4. Пример работы приложения с графическим интерфейсом

Вернемся в режим дизайна формы и добавим кнопке обработчик события Click аналогично тому, как добавляли обработчик событию щелчок мышью по форме. В код добавится заготовка:

```
private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e) {

}
```

Заполним ее, как показано ниже. Нам понадобится две вещественные переменные (чтобы считать туда значения параметров a и b), возможно, также вспомогательные переменные для промежуточных расчетов. Еще один важный момент связан с тем, что на форме данные отображаются в текстовом виде, т.е. имеют

тип, «понятный» операционной системе. Чтобы производить вычисления, понадобится преобразовать типы в стандартные типы C++, понятные компилятору.

```
   private: System::Void button1_Click(System::Object^ sender, System::EventArgs^ e)
{double a,b, res=0;
      a= System::Convert::ToDouble(textBox1->Text);
      b= System::Double::Parse(textBox2->Text);


      if (a+b>0)

      {

      res=sqrt(exp(a*a-9.4)/((a+b)*(a+b)*(a+b)))*sin(a+b*b*b)*sin(a+b*b*b);
      label3->Text = "Результат" + res.ToString();

      }

      else

      label3->Text= "Результат не определен";

   }
```

Поясним некоторые фрагменты кода. В переменные a и b считываем значения текстовых полей, обращаясь к свойству **Text** соответствующего компонента **textBox**. А далее преобразование типов можно про- вести различным образом. Например, для параметра «a» можно воспользоваться методом **ToDouble** класса **Convert** из пространства имен **System**. При работе этого метода могут возникнуть исключительные ситуации. Использование метода **ToDouble(String)** эквивалентно передаче value методу **Double::Parse(String)**. Подробнее о классе **Convert** и структуре **Double** можно прочесть здесь: https://msdn.microsoft.com/ru-ru/library/system.convert(v=vs.110).aspx?cs-save-lang=1&cs-lang=cpp#code-snippet-1, https://msdn.microsoft.com/ru-ru/library/system.double(v=vs.110).aspx?cs-save-lang=1&cs-lang=cpp#code-snippet-1.

Для вывода значения в метку значение надо преобразовать к строковому типу, для чего можно воспользоваться методом **ToString()**.

Примечание: не забудьте подключить заголовочный файл math.h для работы с арифметическими функциями!

Теперь при запуске программы можно задавать значения параметров a и b и вычислять значение.

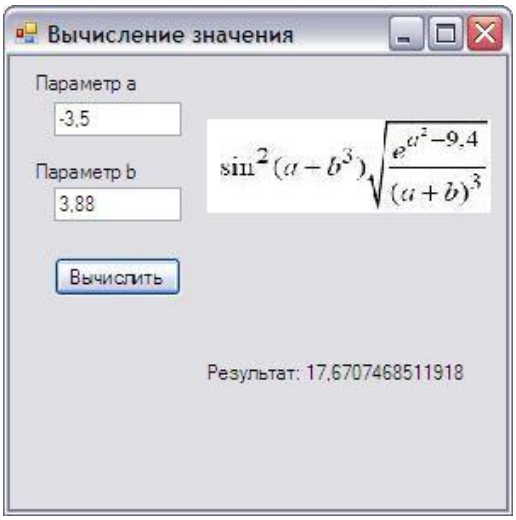


Рис. 5. Пример работы приложения с графическим интерфейсом

Задания для самостоятельной работыЧасть 1. Работа с готовым проектом (вычисление формулы)

1. Попробуйте вводить в текстовые поля неверные значения (например, отделять дробную часть числа точкой, а не запятой, ввести русскоязычный текст, оставить поля пустыми и т.п.). Посмотрите, как «ведет» себя программа. Модифицируйте программу так, чтобы обработать различные исключительные ситуации, связанные с неверным заданием значений параметров.
2. Доработайте программу так, чтобы можно было задавать точность вычисления результата (например, 3 знака после запятой), и вывод значения сделайте с этой точностью.

Часть 2. Самостоятельная разработка проекта

Упражнение 1. Работа с простыми компонентами (задания разработаны Рогачевой Е.В.).

1. В левом верхнем углу окна приложения расположена кнопка. При одинарном щелчке на кнопке левой клавишей мыши она (кнопка) смещается на 10 пикселей вправо и 10 пикселей вниз.
2. В окне приложения выводится надпись «Hello!». При одинарном щелчке левой кнопкой мыши на этой надписи осуществляется ее «перевод» на русский: «Привет!».
3. В окне приложения расположены кнопка и метка. Метка имеет заголовок: «Это мой проект». Одинарный щелчок на кнопке левой клавишей мыши приводит к замене существующего заголовка формы заголовком метки.
4. В окне приложения расположена метка. Одинарный щелчок левой клавишей мыши на метке приводит к смещению метки на 30 пикселей вниз, а щелчок на форме –смещает метку на 15 пикселей вверх и 5 пикселей вправо.
5. В окне приложения расположены метка и кнопка. Одинарный щелчок на метке приводит к тому, что кнопка перемещается и закрывает собой метку. Щелчок на кнопке возвращает кнопку в исходное положение.
6. В окне приложения расположены метка и кнопка. Метка имеет заголовок «Прыгай!», заголовок у кнопки отсутствует. Одинарный щелчок левой клавишей мыши на метке приводит к исчезновению заголовка метки и появлению этого заголовка у кнопки, щелчок на кнопке восстанавливает исходную ситуацию.
7. В окне приложения расположены метка и кнопка. Одинарный щелчок левой клавишей мыши на метке приводит к смещению метки и кнопки на 20 пикселей вниз, а щелчок на кнопке – к смещению метки и кнопки на 20 пикселей влево.
8. В окне приложения расположена кнопка. Одинарный щелчок левой клавишей мыши на кнопке приводит к увеличению ее во всех направлениях (вверх, вниз, вправо и влево) на 10 пикселей.
9. В окне приложения расположена кнопка. Одинарный щелчок левой клавишей мыши на кнопке приводит к уменьшению высоты формы на 20 пикселей, а щелчок на форме – к увеличению высоты формы на 10 пикселей.
10. В окне приложения расположены метка и кнопка. Щелчок на кнопке приводит к ее перемещению в правый нижний угол формы, а метки в левый верхний угол. Щелчок на форме меняет их местами.
11. В окне приложения расположена метка. Щелчок по форме помещает метку точно по центру формы.

12. В окне приложения расположены метка и кнопка. Щелчок по кнопке приводит к тому, что заголовок проекта становится заголовком метки, а заголовок метки становится заголовком кнопки.
13. В окне приложения расположена кнопка. Щелчок по кнопке делает ее квадратной, щелчок по форме уменьшает длину и высоту кнопки в 1,5 раза.
14. В окне приложения расположены кнопка и метка. Щелчок по кнопке перемещает метку в угол формы (сначала левый верхний, потом правый верхний, правый нижний, левый нижний). Щелчок по метке перекрашивает кнопку в случайный цвет.
15. В окне приложения расположены метка и кнопка. Щелчок по кнопке сворачивает кнопку, щелчок по метке разворачивает.

Упражнение 2. Самостоятельно изучите работу со свойством, управляющим видимостью компонентов

1. В окне приложения расположены метка с заголовком «Visual Studio приветствует Вас!» и кнопка. Одинарный щелчок левой клавишей мыши на кнопке приводит к исчезновению метки с экрана, щелчок в любом месте формы –к возникновению надписи.
2. В окне приложения расположены две кнопки, но изначально видна только одна из них. Одинарный щелчок левой клавишей мыши на видимой кнопке приводит к ее исчезновению, и появлению ранее невидимой. Щелчок на форме делает видимыми одновременно обе кнопки.
3. В окне приложения расположена кнопка. Одинарный щелчок левой клавишей мыши на кнопке при-водит к исчезновению кнопки с экрана, щелчок на форме – к ее появлению правее на 5 пикселей исходного положения.
4. В окне приложения расположены две метки с заголовками «Мышка 1» и «Мышка 2». Одинарный щелчок левой клавишей мыши на форме приводит к появлению кнопки с заголовком «Кошка». Щелчок на кнопке приводит к исчезновению меток.
5. В окне приложения расположены две кнопки с заголовками «День» и «Ночь». Одинарный щелчок левой клавишей мыши на кнопке «День» приводит к появлению метки с заголовком «Солнце», щелчок на кнопке «Ночь» приводит к исчезновению «Солнца» и к появлению такой же «Луны» и двух «Звезд».
6. В окне приложения расположены по кругу несколько меток. Изначально видима только одна из них. Одинарный щелчок левой клавишей мыши на видимой метке приводит к ее исчезновению с экрана, но делает видимой следующую.
7. В окне приложения расположены кнопка и метка, причем кнопка изначально закрывает метку. Одинарный щелчок левой клавишей мыши на кнопке приводит к ее исчезновению с экрана, щелчокна форме вновь показывает ее.
8. В окне приложения расположены три кнопки и метка. Изначально видимы только кнопки. Одинарный щелчок левой клавишей мыши по любой из кнопок

приводит к появлению метки с заголовком
«Выравнивание». Щелчок по метке располагает кнопки, выравнивая их по левому краю. Щелчок на форме вновь делает метку невидимой.

9. В окне приложения расположены две кнопки. Изначально видима только одна из них. Одинарный щелчок левой клавишей мыши на видимой кнопке приводит к обмену заголовками по следующей схеме. Заголовок видимой кнопки становится заголовком формы, а заголовок формы – заголовком

   как видимой, так и невидимой кнопок. После этого видимая кнопка исчезает с экрана, а невидимая появляется на форме.

10. В окне приложения расположены две кнопки (с заголовками «Включить» и «Выключить») и метка. Изначально видимы кнопка «Выключить» и метка. Одинарный щелчок левой клавишей мыши на кнопке «Выключить» приводит к исчезновению этой кнопки и метки и появлению кнопки «Включить». Щелчок на кнопке «Включить» приводит к появлению метки и кнопки «Выключить» и исчезновению кнопки «Включить».

11. В окне приложения расположены две кнопки: «Щука» и «Карась» и три метки – «Водоросли». Щелчок по кнопке «Карась» перемещает «Щуку» на 25 пикселей ближе к «Карасю», а «Карась» прячется в «Водорослях»

12. В окне приложения расположены три кнопки: «Дворник», «Ветер» и «Тротуар» (эту кнопку поместите внизу формы и сделайте ее достаточно длинной) и несколько меток – «Листьев». Нажатие на кнопку «Дворник» приводит к исчезновению «Листьев», нажатие на кнопку «Ветер» возвращает
    «Листья» на прежние места.

13. В окне приложения расположены две кнопки: «Кошка» и «Собака» и три метки: «Рыба», «Молоко»,
    «Кость». Нажатие на кнопку «Кошка» приводит к исчезновению «Рыбы» и «Молока», нажатие накнопку «Собака» приводит к исчезновению «Кошки», «Молока» и «Кости».

14. В окне приложения расположены две кнопки: «Кошка» и «Хозяйка» и метка «Сосиска». Кнопка
    «Хозяйка» изначально невидима. Нажатие на кнопку «Кошка» приводит к исчезновению «Сосиски»и появлению «Хозяйки», нажатие на кнопку «Хозяйка» приводит к исчезновению «Кошки».

15. В окне приложения расположены пять кнопок: «Семечко», «Росток», «Бутон», «Цветок», «Плод». Изначально видима только кнопка «Семечко». Нажатие на эту кнопку приводит к ее исчезновению и появлению «Ростка», нажатие на кноп-ку «Росток» делает невидимой ее и видимой кнопку «Бу- тон», и так далее. Нажатие на кнопку «Плод» делает видимой кнопку «Семечко».

Упражнение 3.

Самостоятельно изучите другие управляющие элементы формы.

1. На форме расположены три флажка и кнопка. Флажки задают red, green, blue компоненты цвета (установите их равными 255). Щелчок по кнопке устанавливает цвет кнопки в соответствии с выбранными флажками компонентами цвета.
2. Поместите на форму флажки и кнопку. Нажатие на кнопку включает все флажки. Повторное нажатие на кнопку снимает все флажки. Рядом с флажками располагается текст, нажатие на текст или флажок должно также включать соответствующий флажок.
3. На форму поместите бегущую строку и список с размерами шрифта. Выбор размера шрифта в списке меняет на таковой размер текста в бегущей строке.
4. На форму поместите флажки и текстовые поля. Например, флажки будут отмечать список экзаменов в зимнюю сессию, а текстовые поля - оценки за экзамены. Поставьте еще один флажок "Сессия сдана". Как только вы проставите все оценки - автоматически появится галочка для флажка "Сессия сдана" и появится окошко с сообщением об итогах сессии (отлично, хорошо, удовлетворительно или неудовлетворительно). Если хотя бы по одному экзамену выставлена неудовлетворительная оценка - флажок "Сессия сдана" очищается и флажком отмечаем экзамен, сданный на 2.На форме располагается текстовая область (RichTextBox), текстовое поле (TextBox), кнопка и радиоигруппа из 2 радиокнопок. При активной одной радиокнопке в поле вводится текст, после нажатия на кнопку он оказывается в текстовой области. В текстовую область непосредственно ничего впечатать нельзя, при попытке это сделать должно выдаваться сообщение-предупреждение. При активной другой радиокнопке в текстовую область вводится текст, после нажатия на кнопку он оказывается в текстовом поле. В текстовое поле непосредственно ничего впечатать нельзя, при попытке сделать это должно появляться предупреждение.
5. Поместите на форму флажки и 4 кнопки. Нажатие на первую кнопку поднимает все флажки, нажа- тие на вторую - снимает все флажки. Нажатие на 3-ю кнопку позволяет поставить все флажки, если они сняты, и снять все флажки, если они подняты. Нажатие на 4 кнопку делает все флажки неактивными, повторное нажатие на эту кнопку делает флажки активными.
6. Поместите на форму флажок, радиогруппу, список, и кнопку. Щелчок мышкой по тексту рядом с флажком должен включать-выключать его. Значения из выпадающего списка должны включать со- ответствующие радиокнопки. Щалчок по кнопке меняет местами надписи на кнопке и рядом с флажком.
7. Поместите на форму флажки и примечание, что выбрать можно ограниченное количество пунктов. Попытка выбора большего числа флажков вызывает предупреждающее сообщение. При этом лишние флажки снимаются. Повторная попытка выбрать лишний флажок должна привести к появлению метки-предупреждения, которая будет "убегать" от курсора мыши, не выходя за пределы страницы.
8. Поместите на форму флажки, несколько радиокнопок, несколько текстовых полей и кнопку. Флажки можно отмечать, а по нажатию на кнопку должно выводиться сообщение о количестве выбранных флажков, радиокнопок и заполненных текстовых полей. После этого заполненные поля становятся неактивными, поднятые флажки и радиокнопки становятся неактивными.
9. Заполняем регистрационную карточку (например, на конференцию). Необходимо

отметить имя, фамилию, университет (из списка), должность (из списка), страну (последнюю можно выбрать из выпадающего списка) . Данные надо продублировать. Для этого поместите на форму все необходимые элементы и кнопку. Заполняется только одна карточка, нажатием на кнопку данные копируются во вторую карточку. Если какое-либо поле не заполнено, либо заполнено неверно (например, од- ни цифры в фамилии), об этом должно быть выдано сообщение и предложено данное поле заполнить еще раз. Приветствуется возможность сохранить данные в файл.

10. Поместите на форму радиокнопки и список. Выбор радиокнопки формирует значения в списке. На- пример, радионопки могут указывать страны, а в выпадающем списке будут присутствовать международные аэропорты данных стран.
11. Поместите на форму несколько радиокнопок с обозначением цветов и выпадающий список с обо- значением цветов. Выбор радиокнопки делает этот цвет цветом фона формы, выбор элемента в списке делает цвет цветом текста на форме.
12. Установите на форму кнопку и список. Выбранное значение из списка должно отображаться на кнопке. Щелчок по кнопке запускает бегущую строку на кнопке. Повторный щелчок останавливает бегущую строку. Бегущая строка должна менять цвет в процессе своего движения. Указание 1: воспользуйтесь компонентом Таймер. Указание 2: компоненты цвета RGB лучше сделать целыми числами и объявить их в секции protected в файле описания формы. Задать начальные значения полям можно в методе InitializeComponent(), либо в конструкторе формы (который все равно вызывает InitializeComponent).
13. Делаем «планировщик заданий». Поместите на форму компонент для выборы даты, прокручивающиеся списки для выбора часов и минут, текстовое поле, кнопку, текстовую область. В текстовое поле записываем задание, выбираем для него дату, устанавливаем время. Щелчок по кнопке добавляет выбранные данные в текстовую область.

    Поместите на форму компонент для выбора даты, текстовую область, метку и кнопку. После выбора даты в метку выводится, какой это день недели, а в текстовой области появляется расписание на этот день. Щелчок по кнопке очищает текстовую область, значение метки, выбранную дату.

Вопросы

1. Что такое обработчик события?
2. Как создать обработчик события для компонента?
3. Поясните, что означает следующая запись: private: System::Void button3_Click(System::Object^ sender, System::EventArgs^ e)?
4. Из каких разделов состоит файл проекта? Покажите эти разделы в файле Вашего первого проекта.
5. Из каких файлов состоит проект? За что отвечает каждый?
6. Как получить текстовое представление файла формы?
7. Что такое режим дизайна и режим разработки? Как между ними можно переключиться?
8. Как открыть панель свойств/событий компонента?
9. Перечислите основные свойства и события компонентов, которые вам понадобились при выполнении упражнений. Поясните особенности работы с ними.

Тестовое задание 1 по программированию в C++

В каждом вопросе выберите один вариант ответа.

1. Идентификатор не может содержать ...
   a) символов подчеркивания;
   b) пробельных символов;
   c) цифр;
   d) латинских букв;
   e) прописных латинских букв.
2. Первым символом идентификатора в C++ можетбыть ...
   a) только буква;
   b) только цифра;
   c) буква или символ подчеркивания;
   d) только символ подчеркивания;
   e) только прописная буква.
3. Длина идентификатора ...
   a) ограничена 255 символами;
   b) ограничена 256 символами;
   c) не может быть длиннее 512 символов;
   d) не может быть короче 512 символов;
   e) не ограничена и зависит от реализациисистемы.
4. Регистр букв в идентификаторах ...
   a) не различается;
   b) не различается, если идентификатор начинается с символа подчеркивания;
   c) различается, если это название функции;
   d) различается;
   e) первая буква всегда должна быть прописной.
5. Область видимости идентификатора - это ...
   a) часть программы, которая видна в редакторе кода;
   b) область программы, в которой данныйидентификатор можно скрыть;
   c) область программы, в которой на данныйидентификатор можно сослаться;
   d) область программы, не доступная из другого модуля;
   e) область программы, доступная из другогомодуля.
6. Комментарии в C++ обозначаются ...
   a) символами // текст комментария // и немогут быть вложенными;
   b) символами /* текст комментария */ и немогут быть вложенными;
   c) символами /* текст комментария */ и могут быть вложенными;
   d) символами { текст комментария } и могутбыть вложенными;
   e) символами */ текст комментария /* и могут быть вложенными.
7. Модификатор signed указывает, что ...
   a) переменная может принимать только отрицательные значения;

b) переменная может принимать неотрицательные значения;
c) переменная может принимать только положительные значения;
d) переменная может принимать как положительные, так и неотрицательные значения;
e) переменная может принимать как положительные, так и отрицательные значения.

8. Модификатор unsigned указывает, что ...
   a) переменная может принимать неотрицательные значения;
   b) переменная может принимать как положительные, так и отрицательные значения;
   c) переменная может принимать только положительные значения;
   d) переменная может принимать только отрицательные значения;
9. переменная может принимать как отрицательные, так и неотрицательные значения. Даны два числа с плавающей точкой a и b, необходимо сравнить их на равенство. Выберите наиболее корректный вариант сравнения ...
   a) a == b
   b) a = b
   c) a – b ==0
   d) a – b = 0
   e) a – b < 0.0001
10. Явное преобразование типов – это ...
   a) автоматическое преобразование всех переменных целого типа к переменным с плавающей точкой;
   b) автоматическое преобразование всех переменных с плавающей точкой к переменным целого типа;
   c) принудительное преобразование всех переменных целого типа к переменным с плавающей точкой с помощью операции приведения типов;
   d) принудительное преобразование типов, осуществляемое программистом с помощью операции приведения типов;
   e) автоматическое преобразование типов, осуществляемое компилятором в процессе вычислений.
11. Неявное преобразование типов – это ...
   a) автоматическое преобразование всех переменных с плавающей точкой к переменным целого типа;
   b) принудительное преобразование типов, осуществляемое программистом с помощью операции приведения типов;
   c) автоматическое преобразование типов, осуществляемое компилятором в процессе вычислений;
   d) автоматическое преобразование всех переменных целого типа к переменным с плавающей точкой;
   e) принудительное преобразование всех переменных целого типа к переменным с плавающей точкой с помощью операции приведения типов.

12. Пусть объявлены переменныеint a, b, c;
    double d;
    Выберите некорректный вид явного преобразования типа ...
    c = (double) a /(double) b;(double)c = a/b;
    d = (double) a /b;d = a /(double) b;c = (double)a/b;

13. Выберите некорректное объявление константы
    ...
    const int ten = 10;
    const int tenten = 2 * ten;const ten = 10;
    const tenten = 2 * ten;const int ten;
    ten = 10;
    const int ten = 10;const ten = 10;

14. Операция typedef ...
    a) не приводит к появлению нового типа, просто создает новое имя уже описанного типа
       typedef int integer;
    b) не приводит к появлению нового типа, просто создает новое имя уже описанного типа
       typedef integer int;
    c) позволяет образовать новый типtypedef int integer;
    d) позволяет образовать новый типtypedef integer int;
    e) позволяет образовать новый типtypedef integer;

15. Дана функция
    int Summa (int a=0, int b=10,int c=5)
    { return a+b+c; }
    и приведены примеры вызовов функции и результат. Выберите верный вариант:
    a) int Summa(1,2,3);
       a=1, b=2, c=3, результат 6;
    b) Summa();
       a=0, b=10, c=5, результат 15;Summa(2,9);
       a=0, b=2, c=9, результат 11;
    c) Summa(9);
       a=9, b=10, c=5, результат 24;
    d) Summa();
       a=0, b=0, c=0, результат 0.

16. Даны два числа с плавающей точкой и два целых числа:
    int a=5, b; float c=4.5, d;
    Выберите корректный вариант деления с на a (вскобках указан результат частного)...
    a) b:= c/a; (b=0.89999);
    b) d=c/a; (d=0.89999);
    c) b:=c/a; (b=0);
    d) b=c/a; (b=0.89999);
    e) d=c/a; (d=0).

17. Даны два массива: int Array1[5] = {0};
int Array2[] = {1, 3, 5, 7, 9};
Необходимо, чтобы массив Array1 получил значения массива Array2. Выберите корректныйвариант ...
    a) Array1 = Array2;
    b) Array1 == Array2;
    c) for (int i=1; i<n; i++)
    {Array1[i]=Array2[i];}
    d) for (int i=1, i<n, i++)
    {Array2[i]==Array1[i];}
    e) корректных вариантов нет.

18. Выберите некорректное объявление переменной
...
    a) int a;int b;int d;
    b) int a=b=c=6;
    c) int a, b, d;
    d) long int a, b = 1;float d = 2.5;
    e) int a=7, b=8, c;

19. Логические выражения в С++ задаются с помощью ...
    a) типа boolean;
    b) целых значений, где истина представляет-ся как 1, а ложь обозначается как 0;
    c) целых значений, где истина представляет- ся как любое ненулевое значение,а ложь обозначается как 0;
    d) целых значений, где ложь представляется как любое ненулевое значение, а истина обозначается через 0;
    e) целых значений, где ложь представляется как 1, а истина обозначается через 0.

20. Переменным x, y, z, sh необходимо задать следующие значения:
переменные x и y должны получить значение 6;переменная z должна получить значение 3; переменная sh должна получить значение 15.
Выберите некорректный вариант присваивания
...
    a) sh=(x=y=6)+(z=3)+6
    b) sh=(x=y=(z=3)+3)+9;
    c) sh=x=y=6+(z=3)+6;
    d) sh=6+3+(x=y=(z=3)+3);
    e) sh=(x=y=3+(z=3))+9;

21. Операция логического “или” ...
    a) обозначается && и возвращает значение0, когда выполняются все условия;
    b) обозначается && и возвращает значение0, когда одно из условий не выполняется;
    c) обозначается || и возвращает значение 0,когда ни одно из условий не выполняется;
    d) обозначается || и возвращает значение 0,когда выполняются оба условия;

e) обозначается || и возвращает значение 0,когда одно из условий не выполняется.

22. Операция логического “и” ...
    a) обозначается && и возвращает значение0, когда выполняются все условия;
    b) обозначается && и возвращает значение0, когда одно из условий не выполняется;
    c) обозначается || и возвращает значение 0,когда выполняются все условия;
    d) обозначается || и возвращает значение 0,когда ни одно из условий не выполняется;

23. обозначается || и возвращает значение 0,когда одно из условий не выполняется.

24. Операция отрицания ...
    является бинарной операцией и обозначается
    ?;
    является унарной операцией и обозначается !;является бинарной операцией и обозначается |
    ;
    является унарной операцией и обозначается ?;является унарной операцией и обозначается | .

25. В приведенных ниже операциях инкремента и декремента исключите некорректные выражение и соответствующее ему описание:
    a) b = b + ++a; величина a возрастает на 1 и это новое значение a используется в выражении, в котором оно встретилось;
    b) b = b + --a; величина a уменьшается на 1 иэто новое значение a используется в выражении, в котором оно встретилось;
    c) b = b + a--; в выражении используется текущее значение a, а затем величина a уменьшается на 1;
    d) b = b + a++; в выражении используется текущее значение a, а затем величина a возрастает на 1;
    e) b = b + a++; величина a возрастает на 1 и это новое значение a используется в выражении, в котором оно встретилось.

26. Выберите случай некорректного синтаксиса структуры if...
    a) if (условное_выражение)
       {оператор1;оператор2; оператор3;}
    b) if (условное_выражение) оператор1; else оператор2;
    c) if (условное_выражение) {оператор}
    d) if (условное_выражение); {оператор;}
    e) if (условное_выражение1) оператор1; elseif (условное_выражение2) оператор2;

27. Операторные скобки в C++ задаются с помощью
    ...
    a) фигурных скобок;
    b) ключевых слов begin и end;
    c) комбинации квадратных и круглых ско-бок;

d) ключевых слов finish и start;
e) квадратных скобок.

28. Выберите корректное описание условной операции ...
    a) условие_1 ? условие_2 : выражение; условие_1 содержит значение условного выражения в случае, если условие “лож- но”; а условие_2 равен значению условного выражения, если условие “истинно”;
    b) условие ? выражение_1 : выражение_2; выражение_1 содержит значение условного выражения в случае, если условие «истинно», а выражение_2 равен значению условного выражения, если условие «ложно»;
    c) выражение ? условие_1 : условие_2; условие_1 содержит значение условного выражения в случае, если условие «истинно», а условие_2 равен значению условного выражения, если условие «лож- но»;
    d) условие ? выражение_1 : выражение_2; выражение_1 содержит значение условного выражения в случае, если условие «ложно», а выражение_2 равен значению условного выражения, если условие «истинно»;
    e) выражение_1: выражение_2: условие ?; выражение_1 содержит значение условного выражения в случае, если условие «истинно», а выражение_2 равен значению условного выражения, если условие «ложно».

29. Выберите некорректное описание структуры switch ...
    a) switch (выражение)
       {case конст1: операторы; break; case конст2: операторы; break; case конст3: операторы; break;default: операторы; }
    b) switch (выражение)
       {case конст_1, конст_2: операторы; break;case конст: операторы; break; default: операторы; }
    c) switch (выражение)
       {case конст: операторы; break; case конст: операторы; break; case конст: операторы; break; }
    d) все указанные описания некорректны;switch (выражение)
       {default: операторы;
       case конст1: операторы; break; case конст2: операторы; break; case конст3: операторы; break; }

30. Метка default в структуре оператора switch ...
    a) является необязательной, может ставитьсякак в начале, так и в конце блока switch;
    b) является необязательной, может ставитьсятолько в конце блока switch;
    c) является необязательной, может ставитьсятолько в начале блока switch;
    d) является обязательной, может ставитьсятолько в конце блока switch;

e) является обязательной, может ставитьсятолько в начале блока switch.

31. Выберите корректное описание оператора goto
...
    a) goto L1:; L1 - метка, может располагатьсякак до оператора goto, так и после него;
    b) goto L1; L1 - метка, может располагатьсятолько после оператора goto;
    c) goto : L1; L1 - метка, может располагатьсякак до оператора goto, так и после него;
    d) goto L1; L1 - метка, может располагатьсякак до оператора goto, так и после него;
    e) goto :L1; L1 - метка, может располагатьсятолько до оператора goto.

32. Областью действия метки в операторе goto яв-ляется ...
    a) блок, в котором находится метка;
    b) блок, в котором находится оператор goto;
    c) функция;
    d) весь файл;
    e) вся программа и подгружаемые модули.

33. Выберите случай корректного синтаксисаструктуры for ...
    a) for (инициализация; условное_выражение; список_выражений); тело_цикла;
    b) for (инициализация, условное_выражение, список_выражений) тело_цикла;
    c) for (инициализация; условное_выражение; список_выражений) тело_цикла;
    d) for (список_выражений, инициализация, условное_выражение;)тело_цикла;
    e) for (условное_выражение;инициализация; список_выражений)тело_цикла;

34. Областью действия переменной, объявленной взаголовке структуры for, является ...
    структура for;файл; функция;
    подгружаемый модуль;прототип функции.

35. Список выражений в структуре for ...
    a) может быть частично пустым, порядоквыражений при этом произволен;
    b) всегда должен быть записан полностью;
    c) всегда должен быть записан полностью, но порядок выражений может быть про- извольным;
    d) всегда должен быть пустым;
    e) может быть пустым.

36. Выберите случай корректного синтаксисаструктуры while ...
    a) while (условие)тело_цикла;
    выполняется, когда условие ложно;
    b) while (условие);тело_цикла;

выполняется, когда условие истинно;

c) while (условие);тело_цикла;
выполняется, когда условие ложно;

d) while (условие)тело_цикла;
выполняется, когда условие истинно;

e) while (тело_цикла)условие;
выполняется, когда условие истинно.

37. Выберите случай корректной записи структурыdo while ...

a) do тело_цикла while (условие);
выполняется, когда условие ложно;

b) do тело_цикла while (условие);
выполняется, когда условие истинно;

c) do тело_цикла while; (условие);
выполняется, когда условие истинно;

d) do тело_цикла while; (условие);
выполняется, когда условие ложно;

e) do (условное_выражение) while; те-ло_цикла;
выполняется, когда условие истинно.

38. Оператор break нужен для ...

a) выхода из структуры;

b) выхода из цикла;

c) выхода из программы;

d) выхода из функции;

e) выхода из Windows.

39. Оператор return нужен для ...

a) выхода из структуры;

b) выхода из цикла;

c) выхода из программы;

d) выхода из функции;

e) выхода из Windows.

40. Оператор continue нужен для ...

a) пропуска функции;

b) пропуска итерации в цикле;

c) пропуска структуры;

d) досрочного завершения программы;

e) выхода из функции.

41. Для инициализации элементов массива нуляминеобходимо ...

a) каждому элементу массива присвоить ну-левое значение;

b) не нужно ничего делать – массив заполняется нулями автоматически;

c) обнулить первую строчку массива, еслимассив двумерный;

d) достаточно присвоить нулевое значение одному элементу массива, остальные эле- менты обнулятся автоматически;

e) названию массива присвоить нулевое значение.

42. Доступ к элементам двумерного массива IntArray осуществляется следующим образом:
    a) IntArray[i],[j];
    b) IntArray[[i],[j]];
    c) IntArray[i];[j];
    d) IntArray[i][j];
    e) IntArray [i,j];

43. Тип возвращаемого значения функции void указывает, что ...
    a) функция не возвращает значения;
    b) функция возвращает значение произвольного типа;
    c) функция может работать только с пара-метрами типа void;
    d) функция определена неверно;
    e) функция может работать только с типомvoid, определенным программистом.

44. Встраиваемая функция описывается с помощьюспецификатора...
    a) inline;
    b) in;
    c) into;
    d) inside;
    e) infunc.

45. Операция sizeof ...
    a) определяет размер, занимаемый программными файлами в байтах;
    b) определяет размер любого типа данных вбайтах;
    c) задает размер типов данных в гигабайтах;
    d) увеличивает размер типа int до 1 мегабайта;
    e) возвращает размер массива.

46. Операция new ...
    a) очищает значение переменной;
    b) добавляет в массив новую строку илистолбец;
    c) выделяет память для хранения перемен-ной;
    d) используется для создания нового типа; используется для описания прототипа функции.

47. Операция delete ...
    a) удаляет строку символьного массива;
    b) освобождает память;
    c) завершает работу программы при нажатии на кнопку Delete на клавиатуре;
    d) очищает файл программы;
    e) уменьшает размер файлов с кодом про-граммы.

48. Перегрузка функций ...
    a) нужна для выполнения одних и тех же действий с разными типами данных;
    b) нужна для загрузки в память функций, выполняющихся большое количество раз;

c) нужна для выполнения одних и тех же действий с разными типами данных, количество параметров должно быть разным;
d) нужна для использования функций с одинаковыми именами для различных действий, но параметры функций должны быть разными;
e) нужна для использования функций с одинаковыми именами для различных действий, но параметры функций должны полностью совпадать.

49. Выберите вариант неправильной записи прототипов двух перегруженных функций ...
a) int MyFunction (int, int);float MyFunction (float, float);
b) int MyFunction (int, int);float MyFunction (float, int);
c) int MyFunction (int a, int b);float MyFunction (float a);
d) int MyFunction (int, int);int MyFunction (int, float, int);
e) int MyFunction (int, int);float MyFunction (int, int);

50. Выберите случай некорректного объявленияодномерного массива:
a) int MyArray[3];
b) int MyArray[]={1,2,3};
c) #define Dim 3;
int MyArray[Dim];
d) const int Dim = 3; int MyArray[Dim];
e) int Dim = 3;
int MyArray[Dim];

51. Шаблоны функций ...
a) необходимы для выполнения одних и тех же действий с разными типами данных;
b) необходимы для загрузки в память функций, выполняющихся большое количество раз;
c) необходимы для использования функций с одинаковыми именами для различныхдействий, параметры функций должны быть разными;
d) необходимы для выполнения одних и тех же действий с разными типами данных, количество параметров должно быть разным;
e) необходимы для использования функций с одинаковыми именами для различныхдействий, параметры функций должны полностью совпадать.

52. С помощью блоков try... catch() в C++ обрабатываются
a) синхронные исключения;
b) асинхронные исключения;
c) пассивные ошибки;
d) активные ошибки;
e) прерывания.

53. Пассивный (реактивный) метод обработкиисключений подразумевает, что
a) исключение игнорируется;
b) исключение обрабатывается только врежиме отладки программы;
c) исключению не дают возникнуть в момент работы программы, заранее предусмотрев все ошибочные ситуации;
d) исключению позволяют возникнуть при выполнении программы, но

программа не завершается аварийно, т.к. исключение обрабатывается;

54. исключению позволяют возникнуть при выполнении программы, программа завершается аварийно. Выберите корректную форму записи блокаперехвата исключений:
    a) catch(){операторы;}
    b) catch {операторы;}
    c) catch (тип_исключения& имя_исключения) {операторы;}
    d) catch (имя_исключения){операторы;}
    e) catcth (спи- сок_исключений_через_запятую) {операторы;}
    f) catch (all) {операторы;}
55. Что происходит в программе, если для те- кущего блока try не находится подходящийобработчик?
    a) управление передается операторам,идущим после блоков catch;
    b) работа программы завершается аварий-но;
    c) генерируется обработчик для исключения unexpected;
    d) поиск обработчиков продолжается для вложенных блоков try, если подходящий catch не будет найден, работа программызавершается аварийно;
    e) поиск обработчиков продолжается для внешних блоков try, если подходящий catch не будет найден, работа программы завершается аварийно.
56. Повторная генерация исключений используется, чтобы
    a) сначала освободить ресурсы, потом об- работать ту же самую исключительную ситуацию;
    b) проверить, что исключительная ситуация была обработана корректно;
    c) гарантированно обработать исключи-тельную ситуацию;
    d) повторная генерация исключений не используется;
    e) сначала освободить ресурсы, потом об- работать остальные исключительные ситуации.
57. Дано следующее описание заголовка функции: void someFunction() throw ();
    Что означает такое описание?
    a) внутри функции обязательно присутствуют блоки try-catch;
    b) функция не генерирует никаких иключений;
    c) функция может сгенерировать любоеисключение;
    d) функцию обязательно надо заключить вблоки try-catch;
    e) в функции переопределяется исключение unexpected и впоследствии генерирует-ся;
    f) такое описание неверное, после заголовка функции не может идти ключевое слово throw.
58. Дано следующее описание заголовка функции:
    void someFunction() throw (myException1,myException2);
    Что означает такое описание?
    a) внутри функции обязательно присутствуют блоки try-catch;
    b) функция не генерирует никаких исключений;
    c) функция может сгенерировать любоеисключение;

d) функция гарантированно генерируеттолько исключения вида myException1, myException2;

e) функция может сгенерировать исключения вида myException1, myException2;

f) в функции переопределяется исключение unexpected и впоследствии генерируется;

такое описание неверное, после заголовка функции не может идти

4. Методические материалы, определяющие процедуру оценивания знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций

4.1. Описание процедур проведения текущего контроля успеваемости студентов

Методические указания для студентов

Текущий и промежуточный контроль результатов освоения дисциплины осуществляется с учетом балльно-рейтинговой системы.

По каждой теме предусмотрены задания из средств оценки результатов обучения, которые студент выполняет в процессе контактной работы с преподавателем либо в часы самостоятельной работы.

При подготовке к лекции и при выполнении самостоятельной работы необходимо прочитать материал предыдущей лекции, стремясь к пониманию всех понятий и утверждений. При изучении курса программирования в C++ лекционные занятия сопровождаются лабораторными занятиями. На лабораторных занятиях предлагается выполнить программирование основных классов задач в C++, желательно использовать для этого инструментальные среды Microsoft Visual Studio, CodeBlocks, Eclipse C++ (либо другие open-source средства по согласованию с преподавателем).

Знакомство с программированием на языке C++ начинается с базовых заданий. Обязательно выполнение лабораторной работы по отладке приложений для освоения режима работы debug в инструментальной среде и лучшего освоения синтаксических конструкций языка.

Следует обратить внимание на вопросы управления памятью в C++. На примере работы с динамическими массивами отрабатываются навыки по работе с указателями, грамотному выделению и освобождению памяти. Файловый ввод-вывод может быть предложен к самостоятельному изучению.

Особое внимание следует обратить на объектную модель в C++, такие моменты, как инкапсуляция данных, наследование и полиморфизм. Необходимо четко понимать различие абстрактных и конкретных классов, виртуальных и чистых виртуальных функций, разницу в применении статического и динамического связывания. По дисциплине предусмотрены задачи, требующие грамотного объектного проектирования, перегрузки операций. Дополнительно можно рекомендовать задания с автоматизированной проверкой с указанных в программе ресурсов.

При изучении динамических структур данных упор следует сделать на организацию структур при помощи классов и шаблонов класса. На заданных примерах необходимо проиллюстрировать умение строить основные динамические структуры данных и выполнять основные операции с ними, используя объектную модель в C++ и шаблоны классов,

желательно включить в программы обработку исключительных ситуаций.

Различные аспекты применения шаблонов в C++ следует закрепить решением небольших задач, иллюстрирующих возможности шаблонов, с указанных в программе ресурсов с автоматизированной проверкой.

Следует обратить особое внимание на стандартную библиотеку C++, структуры данных, описанные в ней. Для корректного освоения темы следует решить дополнительные задания с ресурса acm.timus.org (или другого рекомендованного преподавателем) с автоматизированной проверкой. Для лучшего понимания различий между контейнерами стандартной библиотеки рекомендуется одну и ту же задачу решить с применением различных контейнеров.

4.2. Описание процедур проведения промежуточной аттестации

Зачет

При определении уровня достижений обучающих на зачете учитывается:

- знание программного материла и структуры дисциплины;
- знания, необходимые для решения типовых задач, умение выполнять предусмотренные программой задания;
- владение методологией дисциплины, умение применять теоретические знания при решении задач, обосновывать свои действия.

Средняя оценка уровня сформированности компетенций по результатам текущего контроля	Оценка
Оценка не менее 3,0 и нет ни одной неудовлетворительной оценки по текущему контролю	«зачтено»
Оценка менее 3,0 или получена хотя бы одна неудовлетворительная оценка по текущему контролю	«не зачтено»

Если оценка уровня сформированности компетенций обучающегося не соответствует критериям получения зачета, то обучающийся сдает зачет. Зачет проводится в форме собеседования по перечню теоретических вопросов и решения типовых контрольных заданий. Перечень теоретических вопросов и типовых контрольных заданий обучающиеся получают в начале семестра.

Экзамен

При определении уровня достижений обучающих на экзамене обращается особое внимание на следующее:

- дан полный, развернутый ответ на поставленный вопрос;
- показана совокупность осознанных знаний об объекте, проявляющаяся в свободном оперировании понятиями, умении выделить существенные и несущественные признаки, причинно-следственные связи;
- знание об объекте демонстрируются на фоне понимания его в системе данной дисциплины и междисциплинарных связей;
- ответ формулируется в терминах дисциплины, изложен литературным языком, логичен, доказателен, демонстрирует авторскую позицию обучающегося;

- теоретические постулаты подтверждаются примерами из практики.

ВОПРОСЫ к экзамену

1. Структура программы на языке С++.
2. Простейший пример программы.
3. Встроенные типы переменных, их объявление, определение, инициализация.
4. Выражения в языке С++, приоритет операций.
5. Арифметические операции, явные и неявные преобразования типов.
6. Условные операторы и логические выражения.
7. Операторы цикла и логические выражения.
8. Вложенные условные операторы и оператор switch.
9. Условная операция, логические операции, целые величины в качестве логических.
10. Операторы переходов break, continue, goto.
11. Функции, объявление, определение, параметры и аргументы.
12. Передача аргументов по значению и по ссылке - примеры, основное различие.
13. Перегруженные функции. Аргументы по умолчанию.
14. Область видимости и классы памяти, время жизни переменных.
15. Классы и объекты. Определение класса.
16. . Инкапсуляция - один из основных принципов ООП. Примеры.
17. Методы класса. Объявление, определение и вызов методов.
18. Конструкторы и деструкторы класса. Перегрузка конструкторов.
19. Конструктор копирования.
20. Явная и неявная передача аргументов при вызове методов класса.
21. . Зачем нужны классы? Структурно-модульное программирование и ООП.
22. Массивы, объявление, инициализация, доступ к элементам массива.
23. Многомерные массивы,
24. Массивы объектов, объявление, инициализация, доступ к элементам массива.
25. Строковые переменные и константы, функции strcpy и strcat.
26. Перегрузка операций, преимущества использования.
27. Перегрузка арифметических операций - пример использования.
28. Перегрузка операций отношения - пример использования.
29. Наследование. Базовый и производный класс. Примеры использования.
30. Использование спецификаторов доступа при наследовании.
31. Конструкторы производного класса.
32. Перегрузка функций в производном классе.

33. Иерархия классов. Абстрактный базовый класс.
34. Множественное наследование. Пример использования.
35. Адреса и указатели. Операции & и *.
36. Переменные-указатели, объявление, инициализация.
37. Указатели и массивы. Доступ к элементам массива с помощью переменной-указателя.
38. Указатели и функции. Передача указателей в функции.
39. Передача массивов.
40. Указатели на строки. Копирование строк с использованием указателей.
41. Управление памятью с помощью операций new и delete, преимущества использования.
42. Указатели на объекты и ссылки на переменные и методы класса через указатели.
43. Виртуальные функции. Доступ к обычным и виртуальным функциям через указатели.
44. Пример использования виртуальных функций (класс PERSON).
45. Дружественные функции. Основное назначение. Пример дружественной функции.
46. Перегрузка операций с использованием дружественных функций.
47. Потоки ввода/вывода - операторы << и >>.
48. Потоки ввода/вывода - операторы open, close, read, write.
49. Использование аргументов командной строки в языке C++.
50. Шаблоны функций. Пример использования.
51. Шаблоны классов. Пример использования. 50. Исключения. Пример обработки исключений