

г. Челябинск, 2022

Содержание

Раздел I. «Комплекс основных характеристик программы»	3
1.1 Пояснительная записка	3
1.2 Цель и задачи программы	6
1.3 Учебный план	8
1.4 Содержание программы	9
1.5 Планируемые результаты	10
Раздел II. «Комплекс организационно-педагогических условий»	12
2.1 Календарный учебный график	12
2.2 Условия реализации программы	13
2.3 Формы аттестации	13
2.4 Оценочные материалы	13
2.5 Методы обучения	14
2.6 Список литературы	16
Раздел III. Приложение	17
3.1 Приложение 1. Карта достижений	17
3.2 Приложение 2. Практические занятия	18
3.3 Приложение 3. Анкета по профессиональному самоопределению	43
3.4 Приложение 4. Мониторинговая карта результатов обучения	45

Раздел I. «Комплекс основных характеристик программы»

1.1 Пояснительная записка

Дополнительная общеобразовательная общеразвивающая программа «Основы профессии «Основы программирования» представлена в рамках приоритетного ведомственного проекта Министерства образования и науки Челябинской области «Образовательная индустрия будущего».

Правовыми основами реализации дополнительной общеобразовательной общеразвивающей программы «Основы профессии «Основы программирования» являются:

1. Конституция Российской Федерации
2. Федеральный закон Российской Федерации от 29 декабря 2012 г. N 273-ФЗ «Об образовании в Российской Федерации» (ст. 5ч.6; ст.28; ст.30 ч.2)
3. Концепция развития дополнительного образования детей (Распоряжение Правительства Российской Федерации от 04.09.2014 года № 1726 – р)
4. Приказ Министерства образования и науки Российской Федерации от 09.11.2018 № 196 «Об утверждении Порядка организации и осуществления образовательной деятельности по дополнительным общеобразовательным программам»
5. Письмо Департамента молодежной политики, воспитания и социальной поддержки детей Минобрнауки России от 11.12.2006 № 06-1844 № «О примерных требованиях к программам дополнительного образования детей»
6. Письмо Минобрнауки России от 29 марта 2016 г. №ВК-641/09 «О направлении методических рекомендаций по реализации адаптированных дополнительных общеобразовательных программ, способствующих социально-психологической реабилитации, профессиональному самоопределению детей с ограниченными возможностями здоровья, включая детей-инвалидов, с учетом их особых образовательных потребностей»
7. Письмо Минобрнауки России от 18.11.2015 года № 09-3242 «Методические рекомендации по проектированию дополнительных общеразвивающих программ (включая разноуровневые программы)»
8. Постановление Главного санитарного врача РФ от 04.07.2014 № 41 «Санитарно-эпидемиологические требования к устройству, содержанию и организации режима работы образовательных организаций дополнительного образования детей (СанПиН 2.4.4.3172-14)»

9. Устав ГБПОУ «Южно-Уральский многопрофильный колледж»

Программа «Основы профессии «Основы программирования» относится к программам *естественнонаучной направленности*. Программа предназначена для занятий с обучающимися основной школы в очной форме с применением сетевой формы реализации

Современному выпускнику необходимо быстро включаться в экономические и общественные процессы. Рынок труда остро нуждается в подготовленной к выбору профессии и адаптированной к трудовой деятельности молодежи. Это означает, что выпускники школ должны быть подготовлены к осознанному выбору профессии. Осознанный выбор профессии предполагает наличие знаний о предмете труда, условиях работы, средствах труда, трудовых функциях, профессионально необходимых качествах, уровне заработной платы, медицинских противопоказаниях, путях получения профессии и востребованности профессий на рынке труда. Кроме того, обучающиеся должны иметь представление о собственных интеллектуальных и физических ресурсах, необходимых для получения профессий в сфере IT-технологий.

Актуальность данной программы определяется развитием мотивации на профессиональное самоопределение и воспитание позитивной социализации у школьников. Кроме этого, программа имеет **практическую** направленность, которая позволяет сформировать первоначальные профессиональные компетенции у обучающихся данной категории по профессиям в сфере IT-технологий.

Программа носит развивающий, мотивирующий характер первоначального профессионального образования.

Особенность данной программы: реализация практико-ориентированного обучения, направленного на формирование основ инновационной культуры у обучающихся данной возрастной категории с целью развития имеющихся способностей, формирования компетенций, связанных с определенным видом профессиональной деятельности.

Адресат программы

Программа рассчитана на обучающихся от 13 до 16 лет. В процессе реализации программы учитываются возрастные особенности детей.

Для успешной реализации программы целесообразно объединение обучающихся в учебные группы численностью от 10 до 15 человек. Задания по программе построены с учётом интересов, возможностей и предпочтений обучающихся.

Возрастные особенности детей:

Возрастные особенности 12-15 лет

Основным видом деятельности подростка, как и младшего школьника, является учение, но содержание и характер учебной деятельности в этом возрасте существенно изменяется. Подросток приступает к систематическому овладению основами наук. Обучение становится многопредметным и подростку предъявляются более высокие требования. Это приводит к изменению отношения к учению. Нередко происходит снижение успеваемости.

Подросток не всегда осознает роль теоретических знаний, чаще всего он связывает их с личными, узкопрактическими целями. В то же время подростки склонны к выполнению самостоятельных заданий и практических работ на уроках. Даже учащиеся с низкой успеваемостью и дисциплиной активно проявляют себя в подобной ситуации.

Особенно ярко проявляет себя подросток во внеучебной деятельности. Ярко проявляет себя подросток и в играх. Они любят подвижные игры, но такие, которые содержат в себе элемент соревнования. Особенно ярко в подростковом возрасте проявляются интеллектуальные игры, которые носят состязательный характер.

Подросток стремится к самостоятельности в умственной деятельности. Вместе с самостоятельностью мышления развивается и критичность. В отличие от младшего школьника, который все принимает на веру, подросток предъявляет более высокие требования к содержанию рассказа учителя, он ждет доказательности, убедительности.

В области эмоционально-волевой сферы для подростка характерны большая страстность, неумение сдерживать себя, слабость самоконтроля, резкость в поведении. Для подросткового возраста характерен активный поиск объекта для подражания.

Одной из существенных особенностей личности подростка является стремление быть и считаться взрослым. Подросток всеми средствами пытается утвердить свою взрослость, и в то же время ощущения полноценной взрослости у него еще нет. В связи с «чувством зрелости» у подростка появляется специфическая социальная активность, стремление приобщаться к разным сторонам жизни и деятельности взрослых, приобрести их качества, умения и привилегии.

Для подросткового возраста характерна потребность в общении с товарищами. Подростки не могут жить вне коллектива, мнение товарищей оказывает огромное влияние на формирование личности подростка. Он болезненнее и острее переживает неодобрение коллектива, чем неодобрение учителя. Формирование личности подростка будет зависеть от того, с кем он вступит в дружеские взаимоотношения. Главной основой дружбы подростков является общность интересов. При этом к дружбе предъявляются довольно высокие требования, и дружба носит более длительный характер. Она может сохраниться

на всю жизнь. У подростков начинают складываться относительно устойчивые и независимые от случайных влияний моральные взгляды, суждения, оценки, убеждения.

Объём и сроки освоения программы

Режим занятий:

Год обучения/ № группы	Дата начала обучения по программе	Дата окончания обучения по программе	Всего учебных недель	Количество учебных дней	Количество учебных часов	Режим занятий	Дата начала и окончания каникул	Сроки проведения аттестации
1 полугодие / группы 1	01.09.2022г	30.12.2022г	17	17	34	1р./нед. X 2 часа (1 час – 45 минут)	-	декабрь

Форма обучения – очная

Данная образовательная программа является модернизированной.

Уровень сложности – базовый.

1.2 Цель и задачи программы

Основная цель программы – развитие естественнонаучных способностей и формирование раннего профессионального самоопределения школьников в процессе освоения компетенций профессий в сфере IT-технологий.

Задачи:

Обучающие:

- способствовать получению базовых сведений о профессиональной деятельности программиста;
- научить правилам безопасной работы в кабинете Информационные технологии и с программным обеспечением;
- расширить и научить практическому применению знаний, полученных на уроках математики, информатики.

Развивающие:

- повышать мотивацию к личностному саморазвитию и профессиональному образованию;
- формировать умение соотносить личностные характеристики с профессионально важными качествами;
- способствовать формированию готовности к профессиональному самоопределению;
- развивать умение организовывать собственную деятельность, исходя из цели и способов ее достижения;
- формировать навыки индивидуальной работы и работы в команде.

Воспитательные:

- воспитывать трудолюбие, ответственность;
- прививать уважение к профессиональной деятельности и людям труда.

1.3 Учебный план

Неделя	Тема занятия и содержание	Количество часов			Формы контроля
		всего	теория	Практика	
	Диагностика	2	2		
1	Вводное занятие. Техника безопасности. Охрана труда и пожарная безопасность	2	2		
	Раздел 1. Основы программирования	28	2	26	
2	1.1 Особенности профессий в сфере IT-технологий	2		2	Устный опрос
3	1.2 Практическое занятие «Знакомство со средой программирования»	2		2	Наблюдение, оценка выполнения практической работы
4	1.3 Практическое занятие «Изучение структуры интерфейса приложения Visual Studio»	2		2	
5	1.4 Практическое занятие «Изучение технология создания интерфейса приложения»	2		2	
6	1.5 Практическое занятие «Изучение элементов интерфейса приложения, их свойств и событий»	2		2	
7	1.6 Практическое занятие «Запись выражений на языке программирования»	2		2	
8	1.7 Практическое занятие «Создание программ линейной структуры»	2		2	
9	1.8 Практическое занятие «Использование элементов управления»	2		2	
10	1.9 Практическое занятие «Использование не визуальных элементов управления»	2		2	
11	1.10 Практическое занятие «Простейший калькулятор»	2		2	
12	1.11 Практическое занятие «Создание меню в C# и платформе .NET»	2		2	
13	1.12 Практическое занятие «Создание простейшей анимации при помощи Timer компонента в C# и платформе .NET.»	2		2	
14	1.13 Практическое занятие «Использование окон диалога в формах»	2		2	
15	Итоговое занятие		2		
	Диагностика	2	2		
Всего часов		34	8	26	

1.4 Содержание программы

Диагностика. Входное анкетирование. Методика Л.А. Йовайши определение склонностей личности к различным сферам профессиональной деятельности.

Вводное занятие. Техника безопасности, охрана труда. Пожарная безопасность. Сформировать готовность учащихся к обоснованному выбору профессии, карьеры, жизненного пути, с учетом своих склонностей, способностей, состояния здоровья и потребностей рынка труда в специалистах. Инструктаж по охране труда и технике безопасности. Основные правила по безопасности труда, электробезопасности и поведения учащихся в учебной лаборатории при эксплуатации оборудования. Ознакомление с правилами пожарной безопасности, мерами по предупреждению пожаров. Первичные средства пожаротушения. Проведение вводного инструктажа по т/б, видами травматизма и его причинами.

Раздел 1. Программирование (28 часов)

1.1 Особенности профессий в сфере IT-технологий

Содержание занятия: Характеристика профессий, предмет труда, условия работы, техника безопасности, средства труда, трудовые функции, профессионально необходимые качества, уровень заработной платы, медицинские противопоказания, пути получения профессии, востребованность профессии на рынке труда.

1.2 Практическое занятие «Знакомство со средой программирования»

Содержание занятия: Создание и сохранение проекта в среде разработки. Запуск проекта на выполнение. Режимы работы в приложении.

1.3 Практическое занятие «Изучение структуры интерфейса приложения Visual Studio»

Содержание занятия: знакомство главным окном, окном обозреватель решений, запуск и отладка проекта.

1.4 Практическое занятие «Изучение технология создания интерфейса приложения»

Содержание занятия: знакомство с панелью элементов, настройка свойств и создание событий для элементов управления.

1.5 Практическое занятие «Изучение элементов интерфейса приложения, их свойств и событий»

Содержание занятия: познакомится с основными визуальными и не визуальными компонентами, их свойствами и методами.

1.6 Практическое занятие «Запись выражений на языке программирования»

Содержание занятия: создание проекта, арифметическое выражение в программировании, изучение правил записи выражений.

1.7 Практическое занятие «Создание программ линейной структуры»

Содержание занятия: овладение практическими навыками программированию основных алгоритмических структур, применяемых в языке программирования

1.8 Практическое занятие «Использование элементов управления»

Содержание занятия: Элементы управления, настройка свойств и использование в программе.

1.9 Практическое занятие «Использование не визуальных элементов управления»

Содержание занятия: Знакомство с не визуальными компонентами, настройка свойств, использование в программе.

1.10 Практическое занятие «Простейший калькулятор»

Содержание занятия: закрепление навыков работы с не визуальными компонентами, настройка свойств, использование в программе.

1.11 Практическое занятие «Создание меню в C# и платформе .NET»

Содержание занятия: знакомство с технологией создания главного меню приложения.

1.12 Практическое занятие «Создание простейшей анимации при помощи компонента Timer в C# и платформе .NET.

Содержание занятия: знакомство с технологией работы по созданию движения с помощью компонента Timer

1.13 Практическое занятие «Использование окон диалога в формах»

Содержание занятия: знакомство с различными окнами, предназначенными для ведения диалога с пользователем.

Диагностика. Итоговое анкетирование. «Карта интересов» А.Е. Голомшток Изучения интересов и склонностей школьников старших классов в различных сферах деятельности: физика, математика, химия, астрономия, биология, медицина, сельское хозяйство, филология, журналистика, история, искусство, геология, география, общественная деятельность, право, транспорт, педагогика, рабочие специальности, сфера обслуживания, строительство, легкая промышленность, техника, электротехника.

1.5 Планируемые результаты

Предметные результаты:

По итогам выполнения профессиональных проб обучающиеся должны

знать:

- содержание, характер труда в изучаемой сфере деятельности, требования, предъявляемые к личности и профессиональным качествам;

- общие теоретические сведения по основным разделам учебно-тематического плана программы;

- правила безопасности труда, санитарии, гигиены;
- правила написания программного кода.

уметь:

- выполнять простейшие профессиональные операции; пользоваться программным обеспечением, оборудованием, документацией;
- выполнять санитарно-гигиенические требования и правила безопасности труда;
- соотносить свои индивидуальные особенности с профессиональными требованиями.

Личностные результаты	Метапредметные результаты		
	познавательные	регулятивные	коммуникативные
<ul style="list-style-type: none"> - критическое отношение к результатам собственной деятельности; - осмысление мотивов своих действий при выполнении заданий; - формирование профессионального самоопределения, ознакомление с миром профессий; - уважение к труду, трудолюбие. 	<ul style="list-style-type: none"> - умение использовать знаково-символические средства для выполнения практических задач. 	<ul style="list-style-type: none"> - способность обучающегося принимать и сохранять учебную цель и задачи; - умение планировать собственную деятельность в соответствии с поставленной задачей и условиями её реализации и искать средства её осуществления; - умение контролировать и оценивать свои действия, вносить коррективы в их выполнение на основе оценки и учёта характера ошибок, проявлять инициативу и самостоятельность в обучении 	<ul style="list-style-type: none"> - умение сотрудничать с педагогом и сверстниками при решении учебных задач.

Раздел II. «Комплекс организационно-педагогических условий»

2.1 Календарный учебный график

Год обучения/ № группы	Дата начала обучения по программе	Дата окончания обучения по программе	Всего учебных недель	Количество учебных дней	Количество учебных часов	Режим занятий	Дата начала и окончания каникул	Сроки проведения аттестации
1 полугодие / группы 1	01.09.2022г	30.12.2022г	17	17	34	1р./нед. X 2 часа (1 час – 45 минут)	-	декабрь

2.2 Условия реализации программы

1.1. Требования к минимальному материально-техническому обеспечению

Реализация учебной дисциплины требует наличия лаборатории прикладного программирования.

Оборудование учебного кабинета: Стол преподавателя, стол компьютерный (15 шт.), стул (18 шт.), доска аудиторная белая, одноэлементная, коммутатор D-Link, системный блок Intel Celeron 2.80 ГГц/ОЗУ 1.49 Гб/HDD 80Гб (6 шт.), системный блок Intel Pentium Dual E2200 2.20 ГГц/ ОЗУ 1Гб/HDD 80Гб (6 шт.), системный блок Intel Core i3 540 3.07 ГГц/ ОЗУ 2Гб/HDD 250Гб (1 шт.), Монитор LCD 17" (11 шт.), Монитор LCD 19" (2 шт.)

Кадровое обеспечение:

1) Воропанова И.О. преподаватель высшей категории.

Дополнительная профессиональная программа повышения квалификации преподавателей (мастеров п\о) «Формирование и развитие ИКТ- компетентности ФГОС и проф.стандартов в условиях ЦОС»(2020г.). Выдано удостоверение.

2.3. Формы аттестации

Результатом успешного усвоения программы является усвоение обучающимися знаний и умений, заложенных в программе.

Форма аттестации: практические работы

Формы отслеживания результатов: включенное педагогическое наблюдение, устный опрос, практическая работа для оценивания знаний, умений и навыков.

Формы фиксации результатов: сертификаты.

Формы представления результатов: практические занятия, конкурс профессионального мастерства.

2.4 Оценочные материалы.

Основой образовательного процесса является групповое обучение. Для успешной реализации программы целесообразно объединение обучающихся в учебные группы численностью от 10 до 15 человек. Состав группы постоянный.

Для отслеживания и анализа результатов обучения рекомендуется использование Карты достижений, где усвоение программного материала и развитие других качеств ребёнка определяются по 3 уровням: высокий, средний, достаточный (*Приложение 1*).

Предметные результаты оцениваются на итоговых Практических занятиях каждого

раздела (*Приложение 2*).

Для определения достижения личностных и метапредметных результатов используется педагогическое наблюдение в процессе освоения программы и Анкета по профессиональному самоопределению (*Приложение 3*), Результаты фиксируются в Мониторинговых картах результатов обучения (*Приложение 4*).

2.5 Методы обучения

Основными *методами* обучения являются:

- словесный – передача необходимой для дальнейшего обучения информации;
- демонстрационный – показ педагогом технологий выполнения работ;
- практический – отработка технологий выполнения работ;
- наглядный – просмотр фильмов и презентаций;

Формы проведения занятий: лекция, беседа, практическое занятие.

Задания по программе построены с учётом интересов, возможностей и предпочтений обучающихся. В процессе реализации программы учитываются возрастные особенности детей.

Программа состоит из вводного занятия, 1-го раздела на 28 часов, и заключительного занятия по подведению итогов. Общая продолжительность программы 30 часа.

Основа теоретических блоков программы – вводные беседы о предмете труда, условия работы, технике безопасности, средствах труда, трудовых функциях, профессионально необходимых качествах, уровне заработной платы, медицинских противопоказаниях, путях получения профессии, востребованности профессии на рынке труда.

На практических занятиях дети изучают технологии работы, пробуют выполнять работы вслед за преподавателем и самостоятельно, учатся пользоваться полученными знаниями на практике, получают умения и закрепляют навыки, развивают творческие способности.

Для более полного погружения в вид профессиональной деятельности обучающихся просматриваются презентации и видеоматериалы по профессии. Важным элементом работы является итоговое занятие в виде самостоятельного выполнения практического задания, которое показывает успешность освоения раздела, проверяет наличие способностей к определённому виду деятельности, демонстрирует уровень сформированности общепрофессиональных и профессиональных компетенций, способствует обеспечению профессионального и личностного самоопределения.

В основе программы лежит системно-деятельностный подход, который создает условия для самостоятельного успешного усвоения обучающимися новых знаний, умений, компетенций, видов и способов деятельности и обеспечивает соответствие деятельности обучающихся их возрасту и индивидуальным особенностям.

В этом качестве программа обеспечивает реализацию следующих принципов:

- Принцип деятельности: включение в активную созидательную деятельность; сочетание индивидуальных и коллективных форм работы; связь теории с практикой, приоритет практических занятий.

- Принцип индивидуализации и учета, возрастных психолого-педагогических особенностей развития детей: творческое развитие на различных возрастных этапах и в соответствии с личностным развитием.

- Принцип доступности, последовательности и систематичности: от простого к сложному, с учётом возврата к освоенному содержанию на новом, более сложном творческом уровне; интеграция с учебными программами.

- Принцип вариативности: развитие вариативного мышления – понимания возможности наличия различных вариантов решения задачи и умения осуществлять выбор вариантов.

- Принцип творчества: ориентация на творческое начало, приобретение и расширение собственного опыта творческой деятельности.

2.6 Список литературы

1.2. Информационное обеспечение обучения

Перечень рекомендуемых учебных изданий, Интернет-ресурсов, дополнительной литературы

1. Кудрина, Е. В. Основы алгоритмизации и программирования на языке с# : учеб. пособие для СПО / Е. В. Кудрина, М. В. Огнева. — М. : Издательство Юрайт, 2018. — 322 с. — (Серия : Профессиональное образование). <https://biblio-online.ru/book/osnovy-algoritmizacii-i-programmirovaniya-na-yazyke-c-431505>

2. Трофимов, В. В. Основы алгоритмизации и программирования: учебник для СПО / В. В. Трофимов, Т. А. Павловская ; под ред. В. В. Трофимова. — М. : Издательство Юрайт, 2018. — 137 с. — (Серия : Профессиональное образование) <https://biblio-online.ru/book/osnovy-algoritmizacii-i-programmirovaniya-441286>

Дополнительная литература

1. Демин, А. Ю. Информатика. Лабораторный практикум : учебное пособие для среднего профессионального образования / А. Ю. Демин, В. А. Дорофеев. — Москва : Издательство Юрайт, 2018. — 133 с. — (Профессиональное образование). — ISBN 978-5-534-07984-5. — Текст : электронный // ЭБС Юрайт [сайт].

2. Гниденко, И. Г. Технология разработки программного обеспечения : учебное пособие для среднего профессионального образования / И. Г. Гниденко, Ф. Ф. Павлов, Д. Ю. Федоров. — Москва : Издательство Юрайт, 2018. — 235 с. — (Профессиональное образование). — ISBN 978-5-534-05047-9. — Текст : электронный // ЭБС Юрайт [сайт].

Раздел III.

Приложение 1

Карта достижений

[illegible]

1.2 Практическое занятие «Знакомство со средой программирования»

Цель работы: Знакомство с программной средой Visual Studio. Изучение порядка создания Windows-приложения и структуры Windows-приложения.

Создание Windows-приложения

Запускаем среду Visual Studio и на начальной странице выбираем ссылку *Создать проект*. В разделе шаблонов выбираем *Приложение Windows Forms*. Имя проекта и приложения – *Проба*. Место размещения: *X:\C#*. Будет создана заготовка (рис. 4).

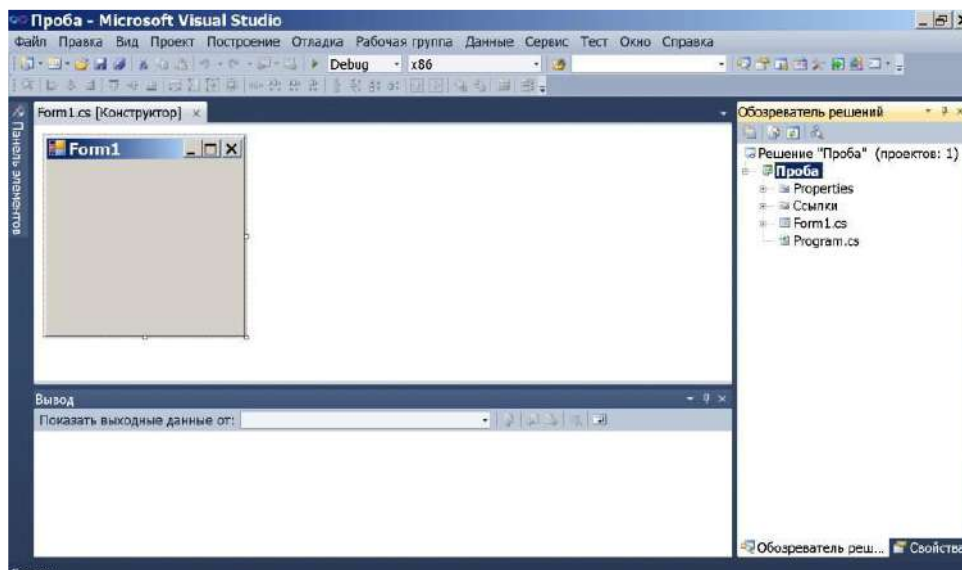


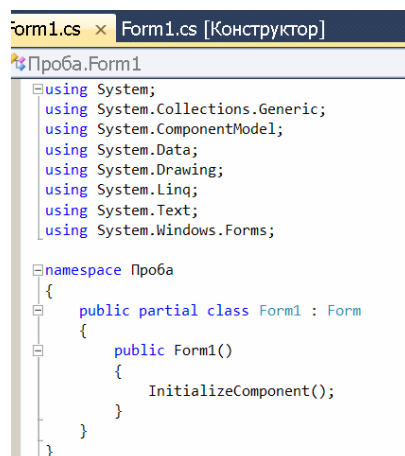
Рис. 4. Заготовка Windows-приложения

В окне *Обозреватель решений* виден состав нашего проекта. Он не пуст – в папках уже имеются компоненты. Именно те, которые нужны для организации Windows-приложения. Каждое решение в C# может содержать несколько проектов. В нашем случае – это один проект под названием *Проба*. В составе проекта есть два программных файла – *Program.cs* (такой файл мы уже видели в консольном приложении) и *Form1.cs*.

Содержание заготовки

На экране слева мы видим отображение содержимого файла *Form1.cs*. Обратите внимание на закладку, соответствующую этому отображению. Закладка помечена словом **[Конструктор]**. Это не случайно. Файл отображается *визуально*. Показано, какие объекты в нем используются. Правда, пока в нём нет никаких объектов, не считая самого окна, но само оно тоже объект.

Установим мышку на окно и нажмём правую кнопку. Отобразится контекстное меню, связанное с окном. Выберем режим *Перейти к коду*. Появится новая закладка с именем *Form1*, но без пометки *Конструктор* (рис. 5).



```

Form1.cs x Form1.cs [Конструктор]
Проба.Form1
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace Проба
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
    }
}

```

Программный код Form1

Отображено содержимое файла **Form1.cs**, но уже не в виде объектов, а в виде программного кода на языке C#. Операторов, подключающих к программе системные пространства имён, здесь значительно больше, чем в консольном приложении. Есть уже известное нам пространство имен **System**. Другие строки подключают подпространства имен пространства **System**. Есть среди них, в частности, и пространство **System.Windows.Forms**, которое содержит всё необходимое для разработки окон Windows-приложения. Имеется пространство имен нашей программы – **Проба**. Внутри него имеется описание класса с именем **Form1**. Мы помним, что класс – это структурная единица программы, ее отдельный модуль, где описываются какие-либо алгоритмы. Описания алгоритмов оформлены в виде функций (методов). Например, в составе класса **Form1** уже имеется метод **Form1()**. Это особый метод: его называют *конструктором объектов класса*. Конструктор объектов есть в любом классе. В данном случае – это конструктор окна **Form1**. Что он содержит и как именно он строит объекты – мы изучим позже. Кроме конструктора в составе класса будут и другие методы, которые напишет программист. Все эти методы будут иметь отношение именно к окну **Form1**.

Рассмотрим модуль **Program.cs**. Выберем его мышкой в окне **Обозреватель решения**. Структуру этого модуля мы уже изучали в предыдущей лабораторной работе. В этом модуле есть класс – **Program**. Он содержит метод **Main()**. Мы помним, что это главный метод программы. Однако, в отличие от консольного приложения здесь этот метод не пуст – в нем уже есть три оператора. Если последовательно навести указатель мышки на каждый оператор, то можно получить сведения о назначении каждого оператора. В целом смысл операторов таков. Два первых оператора разрешают использовать визуальные стили в процессе выполнения программы и задают некоторые стандартные режимы управления программой. Третий оператор обеспечивает запуск процесса, который связан с окном **Form1**. Для этого используется метод **Run** из класса **Application**. В круглых скобках указано имя того объекта, с которым связан процесс.

А теперь с помощью закладки **Form1.cs [Конструктор]** переключимся в окно визуального отображения формы **Form1**. Установим курсор мышки на окне, вызовем контекстное меню и выберем **Свойства**. Отобразится дополнительное окно свойств формы. Свойств много, они различны, знакомиться с ними нужно постепенно, по мере надобности. Для начала отметим свойство **Name**, которое определяет идентификатор окна. Фактически это имя переменной: **Form1**. Именно это имя будет использоваться в программном коде при различных обращениях к объекту окна **Form1**. Свойство **Text** содержит заголовок окна – сейчас заголовок совпадает с именем переменной. Это свойство можно изменить, что приведёт к изменению названия окна. А вот свойство **MaximizeBox** позволяет запретить развёртывание окна на полный экран, если установить значение **False**.

Задание для самостоятельной работы

Поэкспериментируйте с установкой разных значений в свойствах формы. Попробуйте экспериментально понять смысл разных значений. Что-то можно увидеть сразу при изменении значения свойства, а что-то – только после запуска программы. Попробуйте изменить значения у некоторых свойств и, запустив программу (Ctrl-F5), посмотрите, как разные значения свойств проявляются при выполнении программы. Исследуйте:

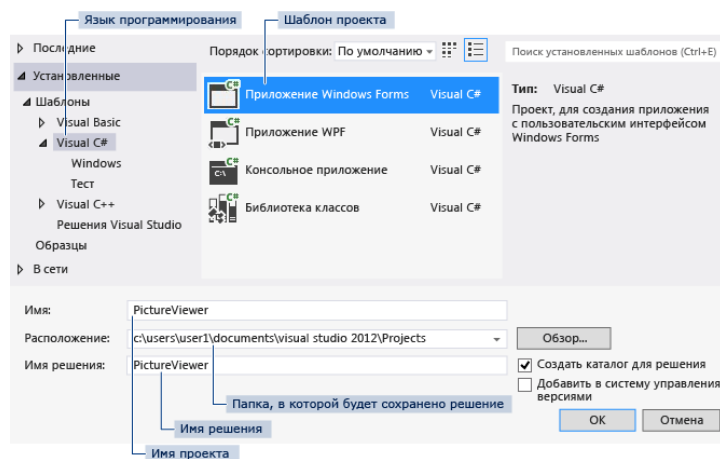
1. *AutoSizeMode* (убедитесь, что при одном из значений окно невозможно растянуть).
2. *BackColor* (устанавливается фоновый цвет).
3. *BackgroundImage* (устанавливается фоновая картинка).
4. *ControlBox* (при установке значения **false** у окна отсутствуют кнопки управления; прервать программу теперь можно только через диспетчер задач Windows).
5. *Cursor* (изменяется внешний вид курсора мышки).
6. *Enabled* (запрещает доступ к элементам окна – если они там есть).
7. *FormBorderStyle* (различные виды оформления границ окна).
8. *Icon* (изменение значка-иконки в заголовочной строке окна).
9. *ShowIcon* (показ или скрывание иконки).
10. *MaximizeBox* (активизация или блокировка кнопки развёртки окна в полный экран).
11. *MinimizeBox* (активизация или блокировка кнопки свёртки окна).
12. *Opacity* (степень прозрачности окна).
13. *Size* (установка размеров окна).
14. *Text* (установка надписи в заголовок окна).

1.3 Практическое занятие

«Изучение структуры интерфейса приложения Visual Studio»

Цель: изучить возможности настройки интегрированной среды Microsoft Visual Studio.

В строке меню выберите Файл, Создать, Проект. Диалоговое окно должно выглядеть следующим образом.

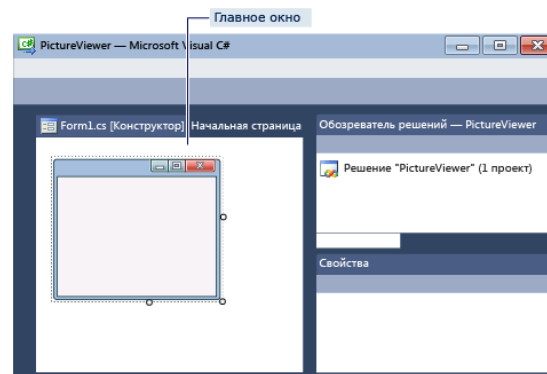


Диалоговое окно "Новый проект"

В списке Установленные шаблоны выберите Visual C# .

В списке шаблонов выберите значок Приложение Windows Forms. Назовите новую форму PictureViewer и нажмите кнопку ОК. Visual Studio создает решение для программы. Решение играет роль контейнера для всех проектов и файлов, необходимых программе. Более подробно эти термины поясняются далее в этом учебнике.

На следующем рисунке показано, как теперь должен выглядеть интерфейс Visual Studio.



Окно интегрированной среды разработки

Интерфейс содержит три окна: главное окно, Обозреватель решений и окно Свойства.

Если какое-либо из этих окон отсутствует, восстановите макет окон по умолчанию, выбрав в строке меню **Окно**, **Сброс макета окон**. Можно также отобразить окна с помощью команд меню. В строке меню выберите **Вид**, **Окно "Свойства"** или **Обозреватель решений**. Если открыты какие-либо другие окна, закройте их с помощью кнопки **Закрыть (x)** в верхнем правом углу.

На рисунке показаны следующие окна (по часовой стрелке от левого верхнего угла):


Главное окно В этом окне выполняется основная часть работы, например работа с формами и редактирование кода. На рисунке в окне показана форма в редакторе форм. В верхней части окна находятся две вкладки — вкладка **Начальная страница** и вкладка **Form1.cs [Design]**. (В Visual Basic имя вкладки заканчивается на **.vb**, а не на **.cs**.)

Окно Обозреватель решений В этом окне можно просматривать все элементы, входящие в решение, и переходить к ним. Если выбрать файл, содержимое в окне **Свойства** изменится. Если открыть файл кода (с расширением **.cs** в Visual C# и **.vb** в Visual Basic), откроется файл кода или конструктор для файла кода. Конструктор — это визуальная поверхность, на которую можно добавлять элементы управления, такие как кнопки и списки. При работе с формами Visual Studio он называется конструктор Windows Forms.

Окно Свойства В этом окне производится изменение свойств элементов, выбранных в других окнах. Например, выбрав форму **Form1**, можно изменить ее название путем задания свойства **Text**, а также можно изменить цвет фона путем задания свойства **BackColor**.

В строке меню выберите **Файл**, **Сохранить все**.

Другой вариант — нажать кнопку **Сохранить все** на панели инструментов, показанной на следующем рисунке.

 Кнопка "Сохранить все" на панели инструментов

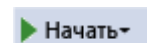
Visual Studio автоматически заполняет имя папки и имя проекта, а затем сохраняет проект в папке проектов.

Для запуска программы используйте один из следующих методов.

Нажмите клавишу **F5**.

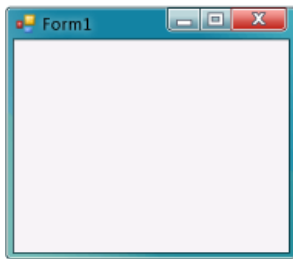
В строке меню выберите **Отладка**, **Начать отладку**.

На панели инструментов нажмите кнопку **Начать отладку**, которая показана на рисунке ниже.



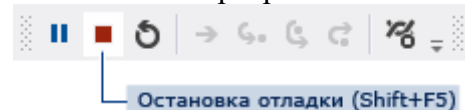
Кнопка панели инструментов "Начать отладку"

Visual Studio запускает программу, и открывается окно **Form1**. На следующей диаграмме показана только что созданная программа. Программа выполняется и скоро она будет дополнена.



Выполнение программы приложения Windows Form

Вернитесь в интегрированную среду разработки Visual Studio и посмотрите на новую панель инструментов. При запуске программы на панели инструментов появляются дополнительные кнопки. Эти кнопки позволяют выполнять такие действия, как остановка и запуск программы, а также помогают отслеживать все ошибки. В этом примере мы просто используем его для запуска и остановки программы.



Панель инструментов "Отладка"

Для остановки программы используйте один из следующих методов.

На панели инструментов нажмите кнопку Остановить отладку.

В строке меню выберите Отладка, Остановить отладку.

Нажмите кнопку X в правом верхнем углу окна Form1.

Убедитесь, что вы смотрите на конструктор Windows Forms. В интегрированной среде разработки Visual Studio откройте вкладку Form1.cs [Design] (или вкладку Form1.vb [Design] в Visual Basic).

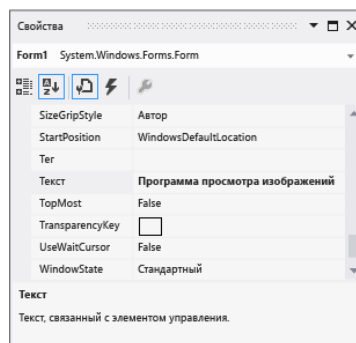
1.4 Практическое занятие

«Изучение технология создания интерфейса приложения»

Цель: познакомиться с возможностями среды программирования по созданию различных интерфейсов приложения.

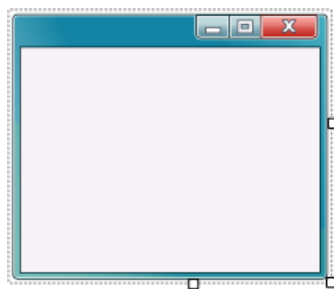
Чтобы выделить форму Form1, щелкните в любом ее месте. Посмотрите на окно Свойства. Теперь в нем должны отображаться свойства формы. У формы есть различные свойства. Например, можно установить цвет переднего плана и фона, текст заголовка, который отображается в верхней части формы, размер формы и другие свойства.

Когда форма будет выбрана, найдите свойство Text в окне Свойства. В зависимости от того, как отсортирован список, может потребоваться прокрутить вниз. Выберите Text, введите "Программа просмотра изображений", затем нажмите клавишу ВВОД. Теперь форма в заголовке окна должна содержать текст Программа просмотра изображений. Окно Свойства должно выглядеть так, как показано на рисунке ниже.



Окно \Свойства\

Вернитесь к конструктору Windows Forms. Нажмите нижний правый маркер перетаскивания формы, который представляет собой небольшой белый квадрат в нижнем правом углу формы и показан на рисунке ниже.



Маркер перетаскивания

Перетащите маркер, чтобы изменить размер формы — она должна стать шире и немного выше.

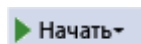
Посмотрите в окно Свойства и обратите внимание, что изменилось значение свойства Size. Свойство Size меняется каждый раз при изменении формы. Перетащите маркер, чтобы форма имела размер около 550, 350 (не обязательно точно такие значения). Такой размер вполне подходит для данного проекта. В качестве альтернативы можно вводить значения непосредственно в свойстве Size и затем нажимать клавишу ВВОД.

Снова выполните программу. Помните, что можно использовать любой из следующих методов для выполнения программы.

Нажмите клавишу F5.

В строке меню выберите Отладка, Начать отладку.

На панели инструментов нажмите кнопку Начать отладку, которая показана на рисунке ниже.



Кнопка панели инструментов "Начать отладку"

Как и ранее, интегрированная среда разработки выполняет построение программы и запускает ее, открывается окно.

Перед переходом к следующему шагу, остановите программу, так как интегрированная среда разработки не позволяет изменять программу при ее выполнении. Помните, что можно использовать любой из следующих методов для остановки программы.

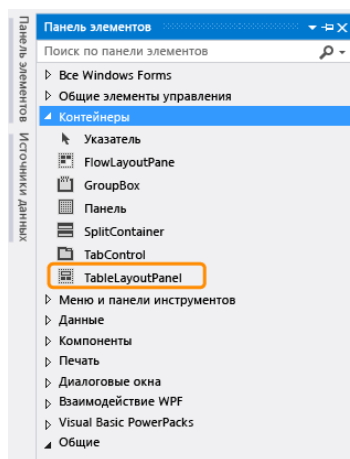
На панели инструментов нажмите кнопку Остановить отладку.

В строке меню выберите Отладка, Остановить отладку.

Нажмите кнопку X в правом верхнем углу окна Form1.

На левой стороне среды разработки Visual Studio найдите вкладку Панель элементов. Выберите вкладку Панель элементов; появится панель элементов. (Или выберите в строке меню Вид, Панель элементов.)

Выберите маленький треугольник рядом с группой Контейнеры, чтобы открыть ее, как показано на рисунке ниже.



Группа "Контейнеры"

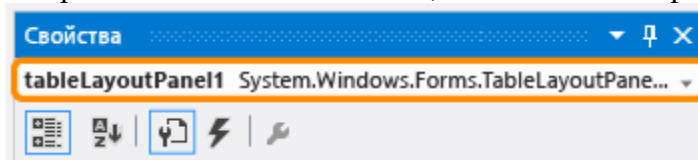
В форму можно добавить такие элементы управления, как кнопки, флажки и метки. Дважды щелкните элемент управления `TableLayoutPanel` в панели элементов. (Можно также перетащить элемент управления с панели элементов в форму.) В результате этого действия интегрированная среда разработки добавляет в форму элемент управления `TableLayoutPanel`, как показано на рисунке ниже. Элемент управления `TableLayoutPanel`

Обратите внимание, как разворачивается панель элементов, чтобы закрыть форму при нажатии на ее вкладку, и закрывается, после щелчка за ее пределами. Это функция автоматического скрывания интегрированной среды разработки. Ее можно включить или выключить для любого из окон путем нажатия значка канцелярской кнопки в правом верхнем углу окна, чтобы включить автоматическое скрывание и закрепление на месте. Появляется значок канцелярской кнопки, как показано на рисунке ниже.



Значок канцелярской кнопки

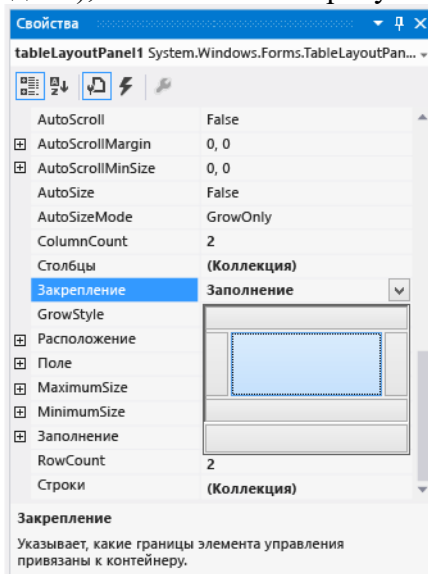
Убедитесь, что выделен элемент управления `TableLayoutPanel`, для этого щелкните по нему. Чтобы проверить, что элемент управления выделен, необходимо посмотреть в раскрывающийся список в верхней части окна Свойства, как показано на рисунке ниже.



Окно "Свойства", в котором показан элемент управления `TableLayoutPanel`

Нажмите кнопку В алфавитном порядке на панели инструментов в окне Свойства. Это приведет к отображению списка свойств в окне Свойства в алфавитном порядке, что упрощает поиск свойств в этом учебнике.

Селектор элементов управления представляет раскрывающийся список в верхней части окна Свойства. В данном примере он показывает, что выделен элемент управления с именем `tableLayoutPanel1`. Элементы управления можно выбирать, либо выделяя область в конструкторе Windows Forms, либо выбирая их в селекторе элементов управления. Теперь, когда выделен элемент управления `TableLayoutPanel`, найдите свойство `Dock` и выберите `Dock`, значение которого должно быть равным `None`. Обратите внимание, что рядом со значением появляется стрелка раскрывающегося списка. Нажмите стрелку, затем нажмите кнопку Заполнение (большая кнопка в середине), как показано на рисунке ниже.



Окно "Свойства", в котором нажата кнопка "Заполнение"

Под закреплением в Visual Studio понимается прикрепление окна к другому окну или области в интегрированной среде разработки. Например, окно "Свойства" можно открепить, т. е. отсоединить и оставить свободно плавающим в пределах Visual Studio или же закрепить в области Обзорщик решений.

После того, как у элемента управления TableLayoutPanel свойству Dock присвоено значение Fill, панель заполняет всю форму. Если снова изменить размер формы, элемент управления TableLayoutPanel останется закрепленным и сам изменит свой размер для заполнения формы.

В данный момент элемент управления TableLayoutPanel содержит две одинаковые по размеру строки и два одинаковых по размеру столбца. Нужно изменить их размер, чтобы верхняя строка и правый столбец были намного больше. В конструкторе Windows Forms выберите элемент управления TableLayoutPanel. В правом верхнем углу расположена маленькая кнопка с черным треугольником, как показано на рисунке ниже.



Кнопка с треугольником

Это кнопка указывает, что элемент управления содержит задачи, которые помогут автоматически задать его свойства.

1.5 Практическое занятие

«Изучение элементов интерфейса приложения, их свойств и событий»

Цель: научиться создавать проект с использованием компонентов для работы с текстом. Разработка проекта «Тук-тук»

Теоретические сведения

Для выполнения настоящего проекта необходимо использовать следующие компоненты: командная кнопка Button, окно редактирования Edit и метка Label;

а также следующие свойства этих компонентов:

Caption (заголовок), Color (цвет), Font (шрифт), Visible (видимость).

В программе используется условный оператор:

if условие then действия1 else действия2

где условие - равенство или неравенство;

действия1 и действия2 - оператор или операторы.

При выполнении условного оператора проверяется условие, если оно истинно (верно), то выполняются действия1, а действия2 - нет (игнорируются). Если же условие ложно, то напротив, выполняются действия2, а действия1 - нет.

Постановка задачи

Компьютер должен запросить имя пользователя (пароль). Если пользователь дает правильный ответ (вводит знакомое имя, например, Вася), то компьютер приветствует его (рис.20). В случае ввода любого другого слова, компьютер должен реагировать иначе, например, выводить запрос: "А где Вася?"

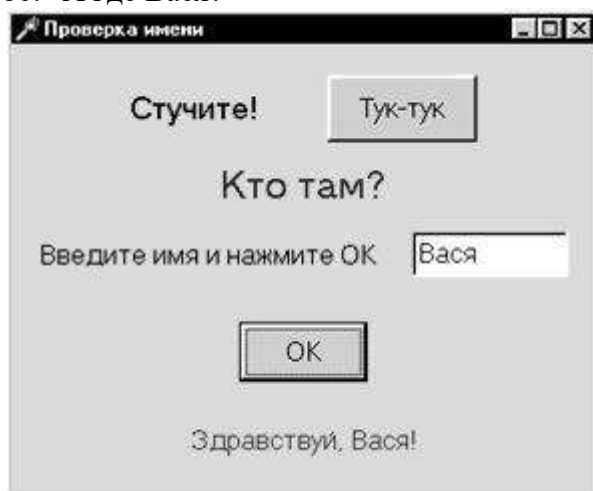


Рис. 20

Практическая часть

Ход работы

1. Откройте новый проект.
2. На новой форме Form1 разместите необходимые компоненты: 2 командные кнопки Button, окно редактирования Edit и 4 метки Label (страница Standard) (рис. 21).

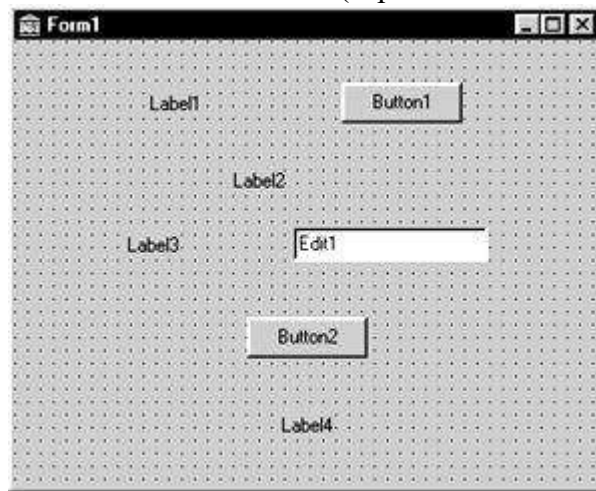


Рис. 21

3. Используя ObjectInspector, задайте необходимые заголовки (свойство Caption) форме, всем меткам и командным кнопкам, а также подберите для них подходящие цвета и размер (см. рис.20). Заголовок четвертой метки, а также текст в окне редактирования (свойство Text) можно сделать пустыми. Выполните необходимое выравнивание всех элементов по горизонтали и по вертикали (Edit @ Align...).

4. Сразу после запуска программы метки Label2 и Label3, окно редактирования Edit1 и кнопка Button2 должны оставаться невидимыми до нажатия на кнопку Button1. Для этого свойство Visible каждого из этих компонентов должно иметь значение False (ложь).

5. Сохраните проект (File Save @ Project As...) в отдельной папке Name, заменив предлагаемое название модуля unit1.pas на nam.pas, а предлагаемое название проекта Project1.dpr - на Name.dpr. Проверьте работоспособность программы на этом этапе и только после этого продолжайте ее разработку. Далее обязательно проверяйте правильность своих действий пробным запуском программы (в конце каждого этапа). При отсутствии ошибок сохраняйте все файлы проекта (File @ Save All).

6. Теперь нужно сделать так, чтобы нажатие на кнопку Button1 приводило к появлению невидимых до этого компонентов (меток Label2 и Label3, окна редактирования Edit1 и кнопки Button2). Для этого выполните двойной щелчок на кнопке Button1. В тексте программы будет создана процедура TForm1.Button1Click(Sender: TObject), в которую впишите следующие строки:

```
Label2.Visible := True;  
Label3.Visible := True;  
Edit1.Visible := True;  
Button2.Visible := True;
```

7. Сначала упростим нашу задачу. Пусть компьютер приветствует по имени любого пользователя. Для этого пустой заголовок метки Label4 после нажатия на кнопку Button2 должен принять значение 'Здравствуй, ' + имя, набранное в строке редактирования Edit1.

Дважды щелкните на кнопке Button2 и впишите следующую строку в созданную процедуру TForm1.Button2Click(Sender: TObject):

```
Label4.Caption := 'Здравствуй, ' + Edit1.Text + '!';
```

8. Теперь сделаем так, чтобы заголовок метки Label4 принимал значение 'Здравствуй, Вася!', если в строку редактирования введено имя Вася, а в любом другом случае заголовком метки Label4 становилось значение 'А где Вася?'. Для этого из процедуры TForm1.Button2Click(Sender: TObject) удалите ранее написанную строку и впишите вместо нее следующий условный оператор:

```
if Edit1.Text = 'Вася' then Label4.Caption := 'Здравствуй, Вася!'
else Label4.Caption := 'А где Вася!';
```

9. Еще раз сохраните все файлы проекта и запустите вашу программу.

Упражнения. Совершенствование проекта

1. Обеспечьте удаление кнопки Button1 (а также и метки Label1) после ее нажатия. Для этого добавьте в процедуру TForm1.Button1Click(Sender: TObject)

строки, задающие значение False свойствам Visible этих компонентов.

2. Засекретим текст, вводимый в строку редактирования Edit1. Для этого в окне ObjectInspector укажите для этого компонента символ (свойство PasswordChar), который будет печататься в строке при наборе имени вне зависимости от набираемых букв (обычно используется *).

3. Сделайте так, чтобы цвет формы менялся в зависимости от правильности введенного имени. Для этого добавьте в условный оператор выражения вида

```
Form1.Color:=clBlack;
```

1.6 Практическое занятие

«Запись выражений на языке программирования»

Цель работы: Научиться записывать арифметические выражения различной степени сложности на языке программирования.

Краткие теоретические сведения:

Арифметические выражения определяет порядок получения некоторого значения. Оно строится из операндов, знаков операций и круглых скобок. Константы, переменные и функции, называемые операндами, должны быть обязательно либо описаны в программе, либо иметь стандартные имена.

Порядок выполнения операций в арифметическом выражении подчиняется трем правилам:

1. Правилу скобок:

Оно гласит, что первыми выполняются операции в скобках. Если несколько пар скобок, вычисления начинаются с самых внутренних скобок.

2. Правилу учета приоритета операций:

Вначале вычисляются значения функций, затем выполняются операции умножения и деления и в последнюю очередь – операции сложения и вычитания.

3. Правилу следования

Операции одинакового старшинства (приоритета) выполняются слева направо в порядке их следования.

Задание

Записать выражения на языке программирования

1	$1+x+\frac{x^2}{2}$	5	$\frac{x^2+y^2}{1-\frac{x^2-y^2}{2}}$
2	$\frac{-b+\sqrt{b^2-4ac}}{2a}$	5	$\frac{\sqrt{ax+bx+c^2}}{15x-\sin x}$
3	$2x+4-\sqrt{\frac{x^2}{2}}$	7	$x-\sqrt{\frac{y^2+2}{2}}+2y$

4	$\sqrt{p(p-a)^2(p-b)(p-c)^2}$	8	$y = -12\sqrt{xy^2 + \cos \frac{a}{2}}$
---	-------------------------------	---	---

1.7 Практическое занятие «Создание программ линейной структуры»

Цели работы: овладение практическими навыками программированию основных алгоритмических структур, применяемых в языке программирования, приобретение дальнейших навыков по организации программ циклической, линейной, ветвящейся структуры с использованием приемов программирования.

Задание:

Необходимо разработать программу позволяющую производить простейшие арифметические вычисления с натуральными числами в соответствии с вариантом задания. Научится строить приложение с использованием простейших визуальных компонентов C#: EditText, Label, Button.

Изучить основные свойства визуальных компонентов: Align, BorderStyle, Caption, Color, Font, Visible, Enabled, Left, Top, Height, Width и т.д..

Варианты заданий.

1. Вычислить площадь и периметр прямоугольника, если задана длина одной стороны (a) и коэффициент n (%), позволяющий вычислить длину второй стороны (b=n*a).
2. Дана длина ребра куба. Найти площадь грани, площадь полной поверхности и объем этого куба.
3. Вычислить периметр и площадь прямоугольного треугольника по заданным длинам двух катетов a и b.
4. Вычислить площади геометрических фигур: прямоугольника и треугольника по заданным сторонам.
5. По известному радиусу вычислить объем и площадь поверхности шара.
6. Даны два числа. Найти среднее арифметическое кубов этих чисел и среднее геометрическое модулей этих чисел.
7. Даны два числа. Вычислить их сумму, разность, произведение и частное.
8. Известен объем информации в байтах. Выразить его в мегабайтах и гигабайтах.
9. Длина выражена в сантиметрах. Выразить ее в дюймах. (1 дюйм=2.5 см)

1.8 Практическое занятие «Использование элементов управления»

Цели работы: овладение практическими навыками программированию основных алгоритмических структур, применяемых в языке программирования, приобретение дальнейших навыков по организации программ циклической, линейной, ветвящейся структуры с использованием приемов программирования.

Задание:

Для этого раздела выбран чисто учебный пример для вычисления тригонометрических функций синус, косинус и тангенс. Значение переменной (угол в радианах) задается в режиме диалога с программой. Также в режиме диалога задается имя вычисляемой функции и количество разрядов формата вывода функции на экран монитора – точность вычисления. Для реализации этой задачи в проекте использованы следующие элементы управления: Label, Button, Panel, RadioButton, ListBox и TextBox.

Разместите элементы управления как показано в примере.

Рисунок 2.1 – Работа программы вычисления функции

Код программы:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        private void button1_Click(object sender, EventArgs e)
        {
            //Значения по умолчанию
            string ff = "F3";
            string fu = "sin";
            Double x=0;
            //Ввод значения угла
            x = Convert.ToDouble(textBox2.Text);
            //Выбор функции
            if (listBox1.SelectedIndex == 1) fu = "cos";
            if (listBox1.SelectedIndex == 2) fu = "tn";
            // точность вычислений
            if (radioButton1.Checked)
            {
```

```

ff = "F3";
}
else
    if (radioButton2.Checked)
    {
ff = "F4";

} else
    if (radioButton3.Checked)
    {
        ff = "F5";
    };
switch (fu)
{
    case "sin": textBox1.Text = " sin= " + Math.Sin(x).ToString(ff); break;
    case "cos": textBox1.Text = " cos= " + Math.Cos(x).ToString(ff); break;
    case "tn": textBox1.Text = " tn= " + (Math.Sin(x) / Math.Cos(x)).ToString(ff); break;
}
}
}
}
}

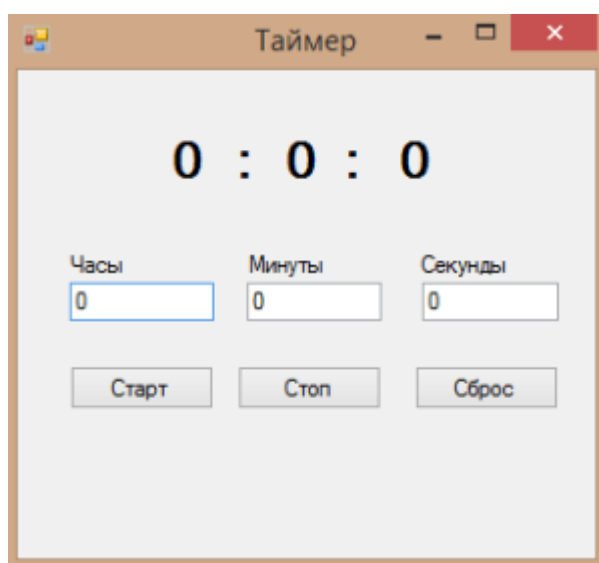
```

1.9 Практическое занятие


«Использование не визуальных элементов управления»»

Цели работы: овладение практическими навыками программированию основных алгоритмических структур, применяемых в языке программирования, приобретение дальнейших навыков по организации программ циклической, линейной, ветвящейся структуры с использованием приемов программирования.

Задание. Для начала в Windows Forms создаём внешнюю оболочку программы. У нас она выглядит вот так:




Здесь у нас 8 Label'ов, 3 TextBox'а, 3 Button'а и сам Timer.

Щёлкнем на значок таймера  timer1 и в окне “Свойства” в группе “Поведение” устанавливаем значение параметра Interval равным 1000. Данный параметр определяет длину тика таймера в миллисекундах, указав 1000, мы сделали один тик равным одной секунде.

После оформления и настройки приступаем к коду. Вводим целочисленные переменные h – часы, m- минуты, s – секунды.

Затем дважды щёлкаем мышью на кнопке “Старт” и переходим на участок кода, отвечающий за клик на эту кнопку.

```
h = Convert.ToInt32(textBox1.Text);  
m = Convert.ToInt32(textBox2.Text);  
s = Convert.ToInt32(textBox3.Text);  
timer1.Start();
```

Также нам надо настроить счёт времени самого таймера. Для этого дважды кликаем на элементе  timer1 и внутри тела кода, в который нас отправило, пишем:

Здесь мы настраиваем таймер таким образом, чтобы каждую секунду переменная s уменьшалась на единицу. Если s становится меньше нуля, значит прошла минута, следовательно, m должна уменьшаться на единицу, а отсчёт с секундами s снова начнётся с 59.

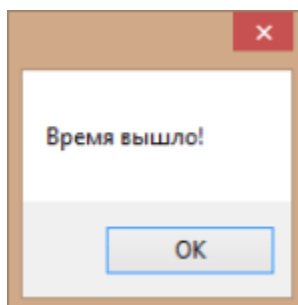
То же самое мы делаем с часами и минутами:

```
if (m == -1)  
{  
    h = h - 1;  
    m = 59;  
}
```

Теперь позаботимся о том, что случится, когда время, указанное пользователем, выйдет:

```
if (h == 0 && m == 0 && s == 0)  
{  
    timer1.Stop();  
    MessageBox.Show("Время вышло!");  
}
```

Как только часы, минуты и секунды будут вместе равняться нулю, мы выведем пользователю окно с предупреждением об этом.



А чтобы пользователь мог видеть, как идёт время, и как отсчитываются часы, минуты и секунды, мы вынесем всё выше происходящее на экран при помощи label'ов:

```
label1.Text = Convert.ToString(h);  
label3.Text = Convert.ToString(m);  
label5.Text = Convert.ToString(s);
```

Теперь надо разобраться с кнопками “Стоп” и “Сброс”. В первом случае при нажатии на кнопку пользователем, таймер просто останавливается и может быть возобновлён после нажатия на кнопку “Старт”. При нажатии на вторую кнопку счётчики сбрасываются и при нажатии на “Старт”, отсчёт начнётся заново.

Код кнопки “Стоп”:

```
private void button2_Click(object sender, EventArgs e)
{
    timer1.Stop();
}
```

Тут всё просто и понятно.

В кнопке “Сброс” нам надо помимо остановки сбросить значения переменных до нулей:

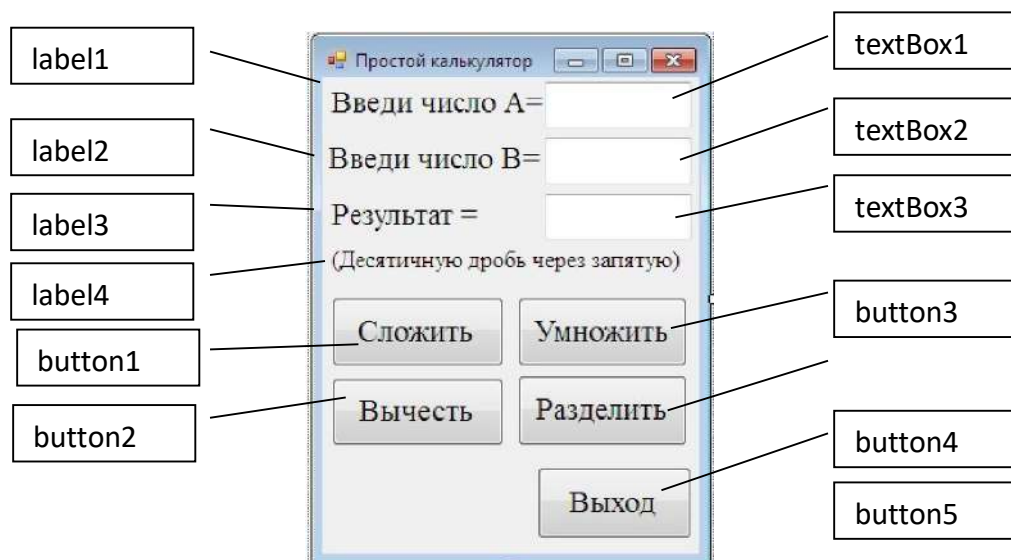
```
private void button3_Click(object sender, EventArgs e)
{
    timer1.Stop();
    label1.Text = "0";
    label3.Text = "0";
    label5.Text = "0";
}
```

1.10 Практическое занятие «Простейшие калькуляторы»

Цели работы: овладение практическими навыками программированию основных алгоритмических структур, применяемых в языке программирования, приобретение дальнейших навыков по организации программ циклической, линейной, ветвящейся структуры с использованием приемов программирования.

1. Калькулятор с кнопками.

Расположите на форме следующие элементы. В свойстве Font формы установите шрифт Times New Roman и 16 размера. Для того что бы форма открывалась в центре экрана свойство StartPosition установите CenterScreen.



Двойным щелчком мышки по кнопке «Сложить» создайте событие и запишите в него следующий код. Для остальных кнопок будет все так же, только меняется знак операции.


```
private void button1_Click(object sender, EventArgs e)
{
    double a = Convert.ToDouble(textBox1.Text);
    double b = Convert.ToDouble(textBox2.Text);
    double c = a + b;
    textBox3.Text = Convert.ToString(c);
}
```

Для кнопки

«Выход» двойным щелчком мыши создайте событие и запишите в него команду закрывающую форму.

2. Калькулятор с выпадающим списком

```
private void button1_Click(object sender, EventArgs e)
{
    label1.Text = textBox1.Text;
    label2.Text = textBox2.Text;
    label3.Text = comboBox1.SelectedItem.ToString();
}
```

Расположите на форме три этикетки label1, label2, label3. Шрифт Times New Roman, свойство StartPosition установлено в Center.

```
private void button1_Click(object sender, EventArgs e)
```

Свойство Font формы установите на открытие формы, чтобы она открывалась в центре экрана.

- textBox1
- textBox2
- GroupBox2
- Combobox1
- button1
- textBox3
- button2

Для Combobox1 в инспекторе объектов для свойства Items нажмите на кнопку с тремя

точками.

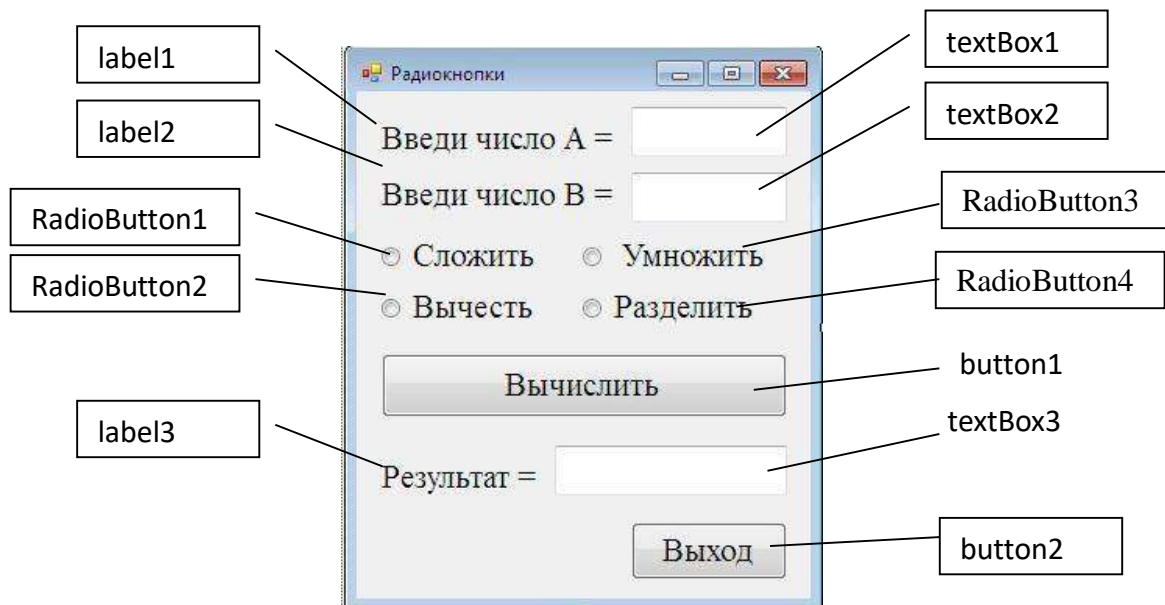
В открывшемся окне запишите названия операций, которые будет выполнять калькулятор.

Для кнопки «Вычислить» двойным щелчком мыши создайте процедуру и добавьте в нее следующий код.

```
private void button1_Click(object sender, EventArgs e)
{
    double a = Convert.ToSingle(textBox1.Text);
    double b = Convert.ToSingle(textBox2.Text);
    double c = 0;
    int k = comboBox1.SelectedIndex;
    if (k == 0) c = a + b;
    if (k == 1) c = a - b;
    if (k == 2) c = a * b;
    if (k == 3) c = a / b;
    textBox3.Text = Convert.ToString(c);
}
```

3. Калькулятор с радиокнопками.

Расположите на форме следующие элементы. В свойстве Font формы установите шрифт Times New Roman и 16 размера. Для того что бы форма открывалась в центре экрана свойство StartPosition установите CenterScreen.



Для кнопки «Вычислить» двойным щелчком мыши создайте процедуру и добавьте в нее следующий код.

```
private void button1_Click(object sender, EventArgs e)
{
    double a = Convert.ToSingle(textBox1.Text);
    double b = Convert.ToSingle(textBox2.Text);
    double c=0;
    if (radioButton1.Checked == true) c = a + b;
    if (radioButton2.Checked == true) c = a - b;
    if (radioButton3.Checked == true) c = a * b;
    if (radioButton4.Checked == true) c = a / b;
    textBox3.Text = Convert.ToString(c);
}
```

Процедуру описания формы добавим строку, которая делает активным первый переключатель:

```
public Form1()
{
    InitializeComponent();
    this.StartPosition=FormStartPosition.CenterScreen;
    radioButton1.Checked = true;
}
```

1.11 Практическое занятие

«Создание меню в C# и платформе .NET

Цели работы: овладение практическими навыками программированию основных алгоритмических структур, применяемых в языке программирования, приобретение дальнейших навыков по организации программ циклической, линейной, ветвящейся структуры с использованием приемов программирования.

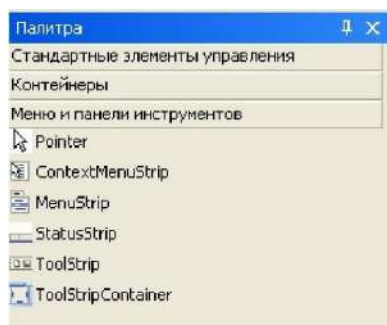
Для создания меню в Windows Forms применяется элемент **MenuStrip**. Данный класс унаследован от ToolStrip и поэтому наследует его функциональность.

Наиболее важные свойства компонента MenuStrip:

- Dock: прикрепляет меню к одной из сторон формы
- LayoutStyle: задает ориентацию панели меню на форме. Может также, как и с ToolStrip, принимать следующие значения
 - HorizontalStackWithOverflow: расположение по горизонтали с переполнением - если длина меню превышает длину контейнера, то новые элементы, выходящие за границы контейнера, не отображаются, то есть панель переполняется элементами
 - StackWithOverflow: элементы располагаются автоматически с переполнением
 - VerticalStackWithOverflow: элементы располагаются вертикально с переполнением
 - Flow: элементы размещаются автоматически, но без переполнения - если длина панели меню меньше длины контейнера, то выходящие за границы элементы переносятся
 - Table: элементы позиционируются в виде таблицы
- ShowItemToolTips: указывает, будут ли отображаться всплывающие подсказки для отдельных элементов меню
- Stretch: позволяет растянуть панель по всей длине контейнера
- TextDirection: задает направление текста в пунктах меню

MenuStrip выступает своего рода контейнером для отдельных пунктов меню, которые представлены объектом **ToolStripMenuItem**.

Создадим простое многоуровневое меню. Для этого выберем в Палитре элемент **MenuStrip**.



Добавим его на форму. В верхней части формы появится заготовка нашего будущего меню, а под формой будет отображаться сам элемент.

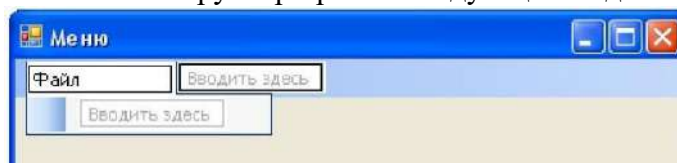


Для добавления доступно три вида элементов: MenuItem (объект ToolStripMenuItem), ComboBox и TextBox. Таким образом, в меню мы можем использовать выпадающие списки и текстовые поля, однако, как правило, эти элементы

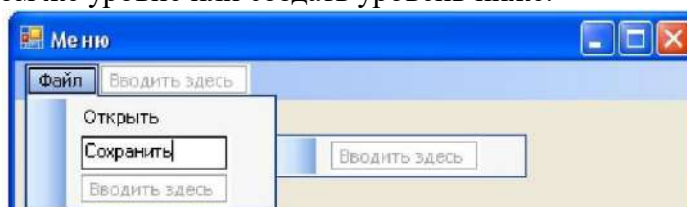
применяются в основном на панели инструментов. Меню же обычно содержит набор объектов ToolStripMenuItem. (Обычные текстовые надписи)



Если щелкнуть мышкой не по выпадающему списку, а по надписи «Вводить здесь» Можно начать создание меню. Конструктор примет следующий вид.



При вводе каждого последующего пункта, конструктор будет предлагать или добавить пункт на этом же уровне или создать уровень ниже.



Когда меню будет готово, двойным щелчком мышки по надписи можно будет создать события соответствующие этим пунктам. Они будут иметь вид:

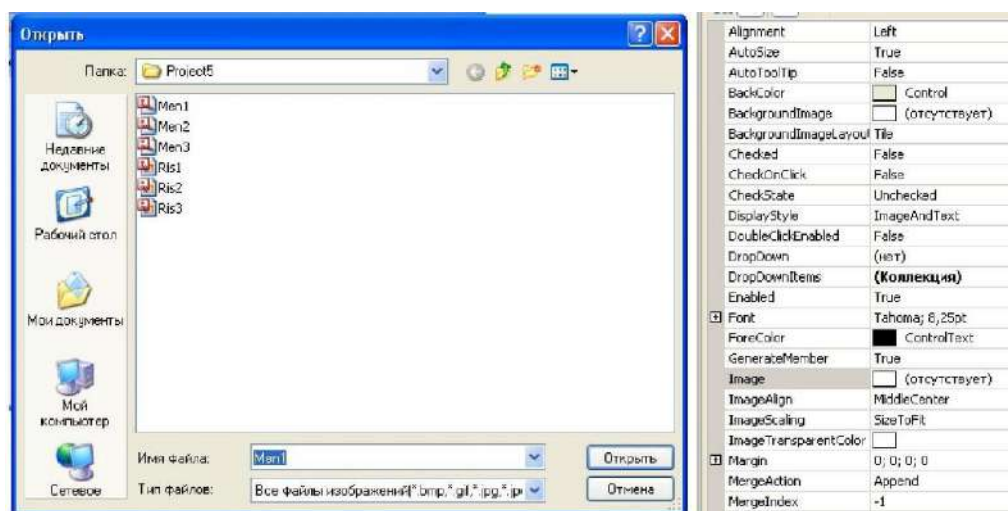
```
private void открытьToolStripMenuItem_Click(object sender, EventArgs e)
{
}
}
```

Сколько будет пунктов в меню, столько будет и процедур (событий).

С помощью изображений мы можем разнообразить внешний вид пунктов меню. Для этого мы можем использовать следующие свойства:

- **DisplayStyle:** определяет, будет ли отображаться на элементе текст, или изображение, или и то и другое.
- **Image:** указывает на само изображение
- **ImageAlign:** устанавливает выравнивание изображения относительно элемента
- **ImageScaling:** указывает, будет ли изображение растягиваться, чтобы заполнить все пространство элемента
- **ImageTransparentColor:** указывает, будет ли цвет изображения прозрачным

Если изображение для пункта меню устанавливается в режиме дизайнера, то нам надо выбрать в окне свойство пункт **Image**, после чего откроется окно для импорта ресурса изображения в проект



Для демонстрации работы пунктов меню добавим компонент **PictureBox**. Растянем его на всю форму. В папку с проектом добавим изображения котиков, переименуем файлы в **Ris1**, **Ris2**, **Ris3**.



В событиях меню про котэ вставим следующие строки:

```
private void рисунокN1ToolStripMenuItem_Click(object sender, EventArgs e)
{
    pictureBox1.Image = Image.FromFile("Ris1.jpg");
}
```

Что бы разные по размеру картинки, отображались точно по размеру окна, свойство **SizeMode** для **PictureBox** изменим на **StretchImage**: изображение растягивается или сжимается таким образом, чтобы вписаться по всей ширине и высоте элемента **PictureBox**.

Так как картинки подгружаются из файла, то сами файлы изображений должны храниться в папке с исполняемым файлом.

1.12 Практическое занятие

«Создание простейшей анимации при помощи компонента **Timer** в **C#**»

Цели работы: овладение практическими навыками программированию основных алгоритмических структур, применяемых в языке программирования, приобретение дальнейших навыков по организации программ циклической, линейной, ветвящейся структуры с использованием приемов программирования.

Для создания этой программы нам потребуются два компонента: **Timer** и **TrackBar**.

Timer является компонентом для запуска действий, повторяющихся через определенный промежуток времени. Хотя он не является визуальным элементом, но его также можно перетащить с Панели Инструментов на форму:

Наиболее важные свойства и методы таймера:

- Свойство **Enabled**: при значении **true** указывает, что таймер будет запускаться вместе с запуском формы
- Свойство **Interval**: указывает интервал в миллисекундах, через который будет срабатывать обработчик события **Tick**, которое есть у таймера
- Метод **Start()**: запускает таймер
- Метод **Stop()**: останавливает таймер

TrackBar представляет собой элемент, который с помощью перемещения ползунка позволяет вводить числовые значения.

Некоторые важные свойства **TrackBar**:

- **Orientation**: задает ориентацию ползунка - расположение по горизонтали или по вертикали
- **TickStyle**: задает расположение делений на ползунке
- **TickFrequency**: задает частоту делений на ползунке
- **Minimum**: минимальное возможное значение на ползунке (по умолчанию 0)
- **Maximum**: максимальное возможное значение на ползунке (по умолчанию 10)
- **Value**: текущее значение ползунка. Должно находиться между **Minimum** и **Maximum**

Свойство **TickStyle** может принимать ряд значений:

- **None**: деления отсутствуют
- **Both**: деления расположены по обеим сторонам ползунка
- **BottomRight**: у вертикального ползунка деления находятся справа, а у горизонтального - снизу
- **TopLeft**: у вертикального ползунка деления находятся слева, а у горизонтального - сверху (применяется по умолчанию)

К наиболее важным событиям элемента следует отнести событие **Scroll**, которое позволяет обработать перемещение ползунка от одного деления к другому. Что может быть полезно, если нам надо, например, устанавливать соответствующую громкость звука в зависимости от значения ползунка, либо какие-нибудь другие настройки.





Разместим на форме объекты как показано на рисунке 1, фон формы (BackColor) сделаем белого цвета под фон картинки. Для создания анимации нам потребуются две картинки, человечка, бегущего влево и вправо. Их можно сделать из одного рисунка, отразив в графическом редакторе картинку слева на право и сохранив как новый рисунок. Первую картинку загрузим непосредственно в PictureBox. Так как движение у нас начнется слева на право. Что бы разные по размеру картинки, отображались точно по размеру окна, свойство **SizeMode** для PictureBox изменим на **StretchImage**: изображение растягивается или сжимается таким образом, чтобы вписаться по всей ширине и высоте элемента PictureBox.

Разместим на форме не визуальный компонент Timer. Он будет отображаться внизу формы в отдельном окне. Его свойство Interval установим равное 10.



Щелкнув по нему двойным щелчком мыши, получим событие, связанное с таймером. В разделе глобальных переменных опишем целочисленную переменную d, отвечающую за скорость и направление движения.

```
public partial class Form1 : Form
{
    int d = 1;
    public Form1 ()
```

Само событие должно выглядеть следующим образом:

```
private void timer1_Tick(object sender, EventArgs e)
{
    //изменяем расстояние до края формы
    pictureBox1.Left = pictureBox1.Left + d;
    //если достигнут левый или правый край формы
    if (pictureBox1.Left>=640 || pictureBox1.Left<=0)
    {
        //в зависимости от того куда двигались,
        // меняем рисунок
        if (d>0) pictureBox1.Image = Image.FromFile("Ris2.jpg");
        if (d<0) pictureBox1.Image = Image.FromFile("Ris1.jpg");
        //изменяем направление движения
        d=-d;
    }
}
```

Рисунки загружаются с диска, поэтому они должны располагаться там, где находится исполняемый файл, или к ним должен быть прописан путь.

Для кнопки «Старт» создадим событие запускающее таймер.

```
private void button3_Click(object sender, EventArgs e)
{
    timer1.Enabled=true;
}
```

Для кнопки «Стоп» создадим событие останавливающее таймер.

```
private void button2_Click(object sender, EventArgs e)
{
    timer1.Enabled = false;
}
```

Элемент `TrackBar` будет отвечать за скорость движения. Настроим его свойства в инспекторе объектов. `Maximum` установим 50, `Minimum` равный 1, а `TickFrequency` (задает частоту делений на ползунке) равным 5.

Двойным щелчком мыши по `TrackBar` создадим событие, происходящее при изменении положения ползунка.

```
private void trackBar1_Scroll(object sender, EventArgs e)
{
    if (d>=0) d=trackBar1.Value;
    if (d<0) d=-trackBar1.Value;
}
```

Осталась кнопка «Выход»:

```
private void button1_Click(object sender, EventArgs e)
{
    this.Close();
}
```

1.13 Практическое занятие

«Использование окон диалога в формах»

Цели работы: овладение практическими навыками программированию основных алгоритмических структур, применяемых в языке программирования, приобретение дальнейших навыков по организации программ циклической, линейной, ветвящейся структуры с использованием приемов программирования.

Упражнение 1. Использование компонента `SaveFileDialog`

Чтобы пользователи могли сохранять файлы, можно использовать встроенный компонент `SaveFileDialog`. В этом упражнении Вы отобразите диалоговое окно, используя метод `ShowDialog`. Затем с помощью поля `DialogResult.OK` проверите, нажал ли пользователь кнопку `OK`.

Для реализации отображения диалогового окна обозревателя папок выполните:

1. Создайте приложение `Windows Forms`,
2. Добавьте элемент `MenuStrip`, задайте имя первого пункта меню `Файл` и команду `Сохранить как...`
3. Добавьте в форму элемент управления `richTextBox`, оставив имя по умолчанию `richTextBox1`. Свойству `Dock` установите `Fill`.
4. Добавьте в форму компонент `SaveFileDialog`. Проверьте, что в области компонентов появился компонент `saveFileDialog1`.
5. Дважды щелкните на пункте меню `Сохранить как`, чтобы добавить в редактор кода обработчик событий по умолчанию.
6. В обработчике событий добавьте следующий код для отображения диалогового окна `Сохранение файла`. Этот код сохраняет текст, введенный в элемент управления `richTextBox`, в текстовый файл в указанной папке.

```
saveFileDialog1.Filter = "txt files (*.txt)*.txt";
if(saveFileDialog1.ShowDialog() ==
System.Windows.Forms.DialogResult.OK && saveFileDialog1.FileName.Length > 0)
{
    richTextBox1.SaveFile(saveFileDialog1.FileName,
RichTextBoxStreamType.PlainText);
}
```


7. Постройте и протестируйте приложение.
8. В открывшейся форме введите какой-либо текст в текстовое поле.
9. Выберите команду Сохранить как... и сохраните файл (имя и место для сохранения файла выберите по своему усмотрению).
10. Убедитесь, что текстовый файл находится в указанном месте.

Упражнение 2. Использование компонента ColorDialog

Для отображения диалогового окна цветовой палитры можно использовать встроенный компонент ColorDialog вместо того, чтобы создавать свое собственное диалоговое окно.

Для реализации отображения диалогового окна цветовой палитры выполните:

1. Для элемента MenuItem задайте имя второго пункта меню – Формат и команду – Цвет фона.
2. Добавьте в форму компонент ColorDialog.
3. Проверьте, что в области компонентов появился компонент colorDialog1.
4. Дважды щелкните кнопку Цвет фона, чтобы создать обработчик событий по умолчанию в редакторе кода.
5. В обработчике событий добавьте следующий код для отображения диалогового окна выбора цвета и изменения фонового цвета в соответствии с выбором пользователя:

```
if (colorDialog1.ShowDialog() == DialogResult.OK)
{
    richTextBox1.BackColor = colorDialog1.Color;
}
```

6. Постройте и протестируйте приложение.

Упражнение 3. Использование компонента FontDialog

Для отображения диалогового окна выбора шрифтов можно использовать встроенный компонент FontDialog вместо того, чтобы создавать свое собственное диалоговое окно.

Для реализации отображения диалогового окна выбора шрифтов выполните:

1. Задайте в меню Формат новую команду – Шрифт.
2. Перетащите в форму компонент FontDialog.
3. Проверьте, что в области компонентов появится компонент fontDialog1.
4. Дважды щелкните команду Шрифт, чтобы создать в редакторе кода обработчик событий по умолчанию.
5. В обработчик событий добавьте следующий код для отображения диалогового окна выбора шрифта текста в окне и изменения шрифта текста в соответствии с выбором пользователя:

```
if (fontDialog1.ShowDialog() == DialogResult.OK)
{
    richTextBox1.Font = fontDialog1.Font;
}
```

6. Постройте и протестируйте приложение.

Упражнение 4. Использование компонента OpenFileDialog

Чтобы пользователи могли выбрать текстовый файл и загрузить его в элемент управления RichTextBox в форме Windows Forms, можно использовать компонент OpenFileDialog.

1. Задайте в меню Файл новую команду Открыть...
2. Перетащите в форму компонент OpenFileDialog. В области компонентов появился компонент openFileDialog1.

3. Подключите класс IO

Using System.IO;

4. Дважды щелкните команду Открыть..., чтобы создать в редакторе кода обработчик событий по умолчанию.

5. В обработчик событий добавьте следующий код для отображения диалогового окна открытия файла:

```
Stream myStream = null;
OpenFileDialog openFileDialog1 = new OpenFileDialog();
openFileDialog1.InitialDirectory = @"c:\";
openFileDialog1.Filter = "txt files (*.txt)*.txt|All
files (*.*)*.*";
openFileDialog1.FilterIndex = 2;
if (openFileDialog1.ShowDialog() == DialogResult.OK)
{
    try
    {
        if ((myStream = openFileDialog1.OpenFile()) != null)
        {
            using (myStream)

richTextBox1.LoadFile(openFileDialog1.FileName,
RichTextBoxStreamType.PlainText);
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show("Error: Could not read file from disk: "
+ ex.Message);
    }
}
```

6. Постройте и протестируйте приложение.

**Анкета
по профессиональному самоопределению**

Ответьте на вопросы предлагаемой анкеты по профессиональному самоопределению.

1. Выбрали ли вы свою будущую профессию?
2. Назовите выбранную вами профессию.
3. Если вы еще не выбрали профессию, то какие профессии вам нравятся?
4. Изменилось ли ваше отношение к профессиям, с которыми вы ознакомились, изучая данную программу?
5. Изменилось ли ваша оценка собственных способностей и возможностей после прохождения профессиональных проб?
6. Что вы можете сказать о людях, выбравших изученные вами профессии?
7. Каковы ваши намерения после окончания школы?
8. В каких профессиях вы хотели бы еще себя попробовать?
9. Предпринимаете ли вы что-нибудь для подготовки себя к будущей профессии?
10. Если вы готовитесь к будущей профессии, то укажите, каким образом вы это делаете.
11. Какими, по вашему мнению, основными качествами должен обладать специалист той профессии, которую вы хотите выбрать?
12. Какими из этих качеств вы обладаете?
13. Если вы не сможете после окончания школы сразу реализовать свои профессиональные намерения, что будете делать?

Мониторинговая карта результатов обучения

Показатели (оцениваемые параметры)	Критерии	Степень выраженности оцениваемого качества	Число баллов	Методы диагностики	Уро вень
П р е д м е т н ы е р е з у л ь т а т ы					
Теоретические знания по основным разделам учебно-тематического плана программы	Соответствие теоретических знаний программным требованиям	С трудом, помощью преподавателя объясняет содержание, характер труда в изучаемой сфере деятельности, требования, предъявляемые к личности и профессиональным качествам.	1	Устный опрос Практические занятия	Д
		Практически не усвоил теоретическое содержание программы; овладел менее чем 1/2 объема знаний, предусмотренных программой.			
		Практически не усвоил правила безопасности труда, санитарии, гигиены, нуждается в подсказке преподавателя.			
		Может рассказать менее 1/2 объема материала по правилам использования оборудования и инвентаря			
		С трудом, но самостоятельно объясняет содержание, характер труда в изучаемой сфере деятельности, требования, предъявляемые к личности и профессиональным качествам. Объем освоенного материала – более 1/2.	2	Устный опрос Практические занятия	С
		Владеет более 1/2 объема знаний по правилам безопасности труда, санитарии, гигиены, предусмотренных программой.			
		Самостоятельно, но с ошибками и не полностью, перечисляет правила безопасности труда, санитарии, гигиены.			
		Может рассказать более 1/2 объема изученного материала по правилам использования оборудования и инвентаря			
		Самостоятельно и в полном объеме объясняет содержание, характер труда в изучаемой сфере деятельности, требования, предъявляемые к личности и профессиональным качествам.	3	Устный опрос	В
		Отлично владеет знаниями по правилам безопасности труда, санитарии, гигиены, предусмотренными программой.			

		Самостоятельно и в полном объеме, перечисляет правила безопасности труда, санитарии, гигиены.		Практические занятия	
		Владеет правилами использования оборудования и инвентаря			
Практические умения и навыки, предусмотренные программой по основным разделам учебно-тематического плана программы	Соответствие практических умений и навыков программным требованиям	Самостоятельно и без ошибок не может выполнять простейшие профессиональные операции	1	Практические занятия Анкетирование	Д
		Нарушает санитарно-гигиенические требования и правила безопасности труда при выполнении практических работ			
		Нуждается в помощи преподавателя при пользовании инвентарем, оборудованием, документацией (технологической картой)			
		Не может в полной мере соотносить свои индивидуальные особенности с профессиональными требованиями			
		Самостоятельно, но с незначительными ошибками выполняет простейшие профессиональные операции.	2	Практические занятия Анкетирование	С
		Имеются незначительные ошибки в выполнении санитарно-гигиенических требований и правил безопасности труда при выполнении практических работ			
		Умеет пользоваться инвентарем, оборудованием, при работе с документацией нуждается в подсказке			
		В основном соотносит свои индивидуальные особенности с профессиональными требованиями			
		Самостоятельно и правильно выполняет простейшие профессиональные операции.	3	Практические занятия Анкетирование	В
		Выполняет санитарно-гигиенические требования и правила безопасности труда при выполнении практических работ			
		Умеет пользоваться инвентарем, оборудованием, с документацией.			
		Соотносит свои индивидуальные особенности с профессиональными требованиями			
Л и ч н о с т н ы е р е з у л ь т а т ы					
Осмысление мотивов своих действий при выполнении заданий;	Сформированность волевых качеств и критического	При выполнении заданий действует не уверенно, нуждается в постоянной помощи и поддержки преподавателя. Не способен адекватно оценить результаты собственной деятельности	1	Педагогическое наблюдение	Д

критическое отношение к результатам собственной деятельности	отношения к собственной деятельности.	При выполнении заданий действует с заминками, сомнением, но самостоятельно. При оценке собственных действий испытывает незначительные трудности.	2	ие при выполнен ии практичес ких работ	С
		Задания выполняет уверенно, в помощи не нуждается. Адекватно оценивает результат работы, указывает на достижения и ошибки.	3		В
Формирование профессионального самоопределения, ознакомление с миром профессий;	Сформированность профессионального самоопределения	Не определился в выборе профессии, не имеет профессиональных предпочтений, слабо владеет знаниями о мире профессий, нет интереса к самоопределению	1	Анкета	Д
		Не определился в выборе профессии, но имеет профессиональные предпочтения, самостоятельно интереса к самоопределению не проявляет, но к новым знаниям относится положительно, не игнорирует	2		С
		Выбрал будущую профессию либо имеет понимание, какие профессии нравятся; предпринимает действия для подготовки себя к будущей профессии	3		В
Уважение к труду, трудолюбие	Понимание ценности труда в жизни человека	Демонстрирует нежелание выполнять практические работы, ленится, отлынивает от занятий, пренебрежительно относится к физическому труду.	1	Педагоги ческое наблюден ие Практиче ские занятия Анкета	Д
		При выполнении практических работ неохотно берется за физическую работу, дифференцирует профессии на достойные и недостойные.	2		С
		Проявляет трудолюбие, демонстрирует уважение к любому труду и людям труда	3		В
М е т а п р е д м е т н ы е р е з у л ь т а т ы					
Познавательные: использовать знаково-символические средства для выполнения практических задач	Самостоятельность в использовании знаково-символических средств для выполнения практических задач	Демонстрирует слабое владение знаково-символическими средствами при выполнении практических работ, требуется помощь преподавателя	1	Педагоги ческое наблюден ие при выполнен ии	Д
		Имеются затруднения в выборе и использовании знаково-символических средств при выполнении практических работ	2		С
		Уверенно и самостоятельно пользуется знаково-символическими средствами при выполнении практических работ	3		В

Регулятивные: способность обучающегося принимать и сохранять учебную цель и задачи; умение планировать собственную деятельность в соответствии с поставленной задачей и условиями её реализации и искать средства её осуществления; умение контролировать и оценивать свои действия, вносить коррективы в их выполнение на основе оценки и учёта характера ошибок, проявлять инициативу и самостоятельность в обучении	Самостоятельность в выполнении практических работ	Требуется постоянной помощи и поддержки при выполнении практических работ со стороны преподавателя	1	практических работ	Д
		Иногда нуждается в незначительной помощи преподавателя при выполнении практических работ со стороны преподавателя	2		С
		Самостоятельно выполняет практические работы	3		В

Коммуникативные умение сотрудничать с педагогом и сверстниками при решении учебных задач	Адекватность восприятия информации идущей от преподавателя и сверстников	Объяснения преподавателя не слушает, учебную информацию не воспринимает; испытывает серьезные затруднения в концентрации внимания и работе в группе, с трудом воспринимает учебную информацию.	1	Педагогическое наблюдение	Д
		Слушает и слышит преподавателя, воспринимает учебную информацию при напоминании и контроле, иногда принимает во внимание мнение сверстников.	2		С
		Сосредоточен, внимателен, слушает и слышит преподавателя, адекватно воспринимает информацию, уважает мнения других, продуктивно работает в группе.	3		В