Комплект контрольно-оценочных средств разработан на основе Федерального государственного образовательного стандарта среднего профессионального образования по специальности СПО**09.02.03 Программирование в компьютерных системах** программы учебной дисциплины: ПМ.01 Разработка программных модулей программного обеспечения для компьютерных систем

**Разработчик(и):**

\_СмК\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_преподаватель \_ \_Л.А. Хвалько

(место работы) (занимаемая должность) (инициалы, фамилия)

**Эксперт(ы) от работодателя:**

\_ ООО «Кибер-софт» \_\_\_\_\_\_\_\_\_\_\_\_\_ директор \_\_\_\_\_\_\_\_\_\_\_ Ю.В. Рокотов \_

(место работы) (занимаемая должность) (инициалы, фамилия)

**Общие положения**

Результатом освоения профессионального модуля является готовность обучающегося к выполнению вида профессиональной деятельности ПМ.01 Разработка программных модулей программного обеспечения для компьютерных систем

Формой аттестации по профессиональному модулю является экзамен (квалификационный).

Итогом экзамена является однозначное решение: «вид профессиональной деятельности освоен с оценкой / не освоен»

**1. Результаты освоения модуля, подлежащие проверке**

**1.1. Профессиональные и общие компетенции:**

|  |  |
| --- | --- |
| ПК 1.1 | Выполнять разработку спецификаций отдельных компонент. |
| ПК 1.2 | Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля. |
| ПК 1.3 | Выполнять отладку программных модулей с использованием специализированных программных средств. |
| ПК 1.4 | Выполнять тестирование программных модулей. |
| ПК 1.5 | Осуществлять оптимизацию программного кода модуля. |
| ПК 1.6 | Разрабатывать компоненты проектной и технической документации с использованием графических языков спецификаций. |
| ОК 1 | Понимать сущность и социальную значимость своей будущей профессии, проявлять к ней устойчивый интерес. |
| ОК 2 | Организовывать собственную деятельность, выбирать типовые методы и способы выполнения профессиональных задач, оценивать их эффективность и качество. |
| ОК 3 | Принимать решения в стандартных и нестандартных ситуациях и нести за них ответственность. |
| ОК 4 | Осуществлять поиск и использование информации, необходимой для эффективного выполнения профессиональных задач, профессионального и личностного развития. |
| ОК 5 | Использовать информационно-коммуникационные технологии в профессиональной деятельности. |
| ОК 6 | Работать в коллективе и в команде, эффективно общаться с коллегами, руководством, потребителями. |
| ОК 7 | Брать на себя ответственность за работу членов команды (подчиненных), за результат выполнения заданий. |
| ОК 8 | Самостоятельно определять задачи профессионального и личностного развития, заниматься самообразованием, осознанно планировать повышение квалификации. |
| ОК 9 | Ориентироваться в условиях частой смены технологий в профессиональной деятельности. |

Сформированность компетенций (в т. ч. частичная для общих) может быть подтверждена как изолированно, так и комплексно. В ходе экзамена (квалификационного) предпочтение следует отдавать комплексной оценке.

Показатели сформированности следует указывать для каждой компетенции из перечня.

Таблица 1

|  |  |
| --- | --- |
| **Профессиональные и общие компетенции** | **Показатели оценки результата** |
| ПК 1.1. Выполнять разработку спецификаций отдельных компонент. | Точность определения основных этапов разработки программного обеспечения;  Правильность применения основных принципов технологии структурного и объектно-ориентированного программирования;  Правильность оформления документации на программные средства;  Правильность и точность разработки алгоритма поставленной задачи |
| ПК 1.2. Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля. | Правильность применение основных принципов технологии структурного и объектно-ориентированного программирования;  Правильность и точность разработки кода программного модуля на современных языках программирования;  Точность создания программы по разработанному алгоритму как отдельного модуля;  Правильность разработки кода программного продукта на основе готовой спецификации на уровне модуля; |
| ПК 1.3. Выполнять отладку программных модулей с использованием специализированных программных средств. | Правильность применения основных принципов отладки и тестирования программных продуктов;  Точность использования инструментальных средств на этапе отладки программного продукта;  Правильность отладки и тестирование программы на  уровне модуля; |
| ПК 1.4. Выполнять тестирование программных модулей. | Проведения тестирования программного модуля по определенному сценарию;  Правильность выполнения отладки и тестирование программы на уровне модуля; |
| ПК 1.5. Осуществлять оптимизацию программного кода модуля. | Точность проведения оптимизации программного кода модуля по определенному сценарию;  Правильность выполнения отладки и тестирование программы на уровне модуля;  Правильность использования инструментальных средств на этапе отладки программного продукта |
| ПК 1.6. Разрабатывать компоненты проектной и технической документации с использованием графических языков спецификаций. | Правильность использования инструментальные средства для автоматизации оформления документации;  Правильность определения и использование методов и средств разработки технической документации |

Таблица 3

|  |  |
| --- | --- |
| **Результаты**  **(освоенные общие компетенции)** | **Основные показатели оценки результата** |
| ОК 1. Понимать сущность и социальную значимость своей будущей профессии, проявлять к ней устойчивый интерес. | - ак­тив­ность и ини­циа­тив­ность в про­цес­се ос­вое­ния про­фес­сио­наль­ной дея­тель­но­сти;  - уча­стие в сту­ден­че­ских кон­фе­рен­ци­ях, кон­кур­сах и т.п. |
| ОК 2. Организовывать собственную деятельность, выбирать типовые методы и способы выполнения профессиональных задач, оценивать их эффективность и качество | * обос­но­ван­ность вы­бо­ра и при­ме­не­ния ме­то­дов и спо­со­бов ре­ше­ния про­фес­сио­наль­ных за­дач в об­лас­ти раз­ра­бот­ки и ад­ми­ни­ст­ри­ро­ва­ния баз дан­ных; * свое­вре­мен­ность вы­пол­не­ния ра­бот и оцен­ка их ка­че­ст­ва и точ­но­сти. |
| ОК 3. При­ни­мать ре­ше­ния в стан­дарт­ных и не­стан­дарт­ных си­туа­ци­ях и не­сти за них от­вет­ст­вен­ность. | * бы­ст­ро­та оцен­ки си­туа­ции и аде­к­ват­ность при­ня­тия ре­ше­ния при вы­пол­не­нии стан­дарт­ных и не­стан­дарт­ных про­фес­сио­наль­ных за­дач в об­лас­ти раз­ра­бот­ки и ад­ми­ни­ст­ри­ро­ва­ния баз дан­ных |
| ОК 4. Осу­ще­ст­в­лять по­иск и ис­поль­зо­ва­ние ин­фор­ма­ции, не­об­хо­ди­мой для эф­фек­тив­но­го вы­пол­не­ния про­фес­сио­наль­ных за­дач, про­фес­сио­наль­но­го и лич­но­ст­но­го раз­ви­тия. | - ре­зуль­та­тив­ность по­ис­ка ин­фор­ма­ции в раз­лич­ных ис­точ­ни­ках, в т.ч. се­ти Ин­тер­нет;  - аде­к­ват­ность от­бо­ра и ис­поль­зо­ва­ния по­лу­чен­ной ин­фор­ма­ции для ре­ше­ния про­фес­сио­наль­ных за­дач. |
| ОК 5. Ис­поль­зо­вать ин­фор­ма­ци­он­но-ком­му­ни­ка­ци­он­ные тех­но­ло­гии в про­фес­сио­наль­ной дея­тель­но­сти. | - ре­зуль­та­тив­ность по­ис­ка ин­фор­ма­ции в Ин­тер­не­те;  - аде­к­ват­ность от­бо­ра и ис­поль­зо­ва­ния ин­фор­ма­ции для ре­ше­ния про­фес­сио­наль­ных за­дач. |
| ОК 6. Ра­бо­тать в кол­лек­ти­ве и в ко­ман­де, эф­фек­тив­но об­щать­ся с кол­ле­га­ми, ру­ко­во­дством, по­тре­би­те­ля­ми. | - со­блю­де­ние эти­че­ских норм при взаи­мо­дей­ст­вии с обу­чаю­щи­ми­ся, пре­по­да­ва­те­ля­ми и ад­ми­ни­ст­ра­ци­ей, ком­му­ни­ка­тив­ная то­ле­рант­ность. |
| ОК 7. Брать на се­бя от­вет­ст­вен­ность за ра­бо­ту чле­нов ко­ман­ды (под­чи­нен­ных), за ре­зуль­тат вы­пол­не­ния за­да­ний. | - ре­зуль­та­тив­ность ис­пол­не­ния функ­ций ру­ко­во­ди­те­ля ра­бот, вы­пол­няе­мых груп­пой. |
| ОК 8. Са­мо­стоя­тель­но оп­ре­де­лять за­да­чи про­фес­сио­наль­но­го и лич­но­ст­но­го раз­ви­тия, за­ни­мать­ся са­мо­об­ра­зо­ва­ни­ем, осоз­нан­но пла­ни­ро­вать по­вы­ше­ние ква­ли­фи­ка­ции. | - по­зи­тив­ная ди­на­ми­ка учеб­ных дос­ти­же­ний;  - уча­стие в раз­лич­ных се­ми­на­рах и кон­фе­рен­ци­ях. |
| ОК 9. Ори­ен­ти­ро­вать­ся в ус­ло­ви­ях час­той сме­ны тех­но­ло­гий в про­фес­сио­наль­ной дея­тель­но­сти. | * про­яв­ле­ние ин­те­ре­са к ин­но­ва­ци­ям в об­лас­ти про­фес­сио­наль­ной дея­тель­но­сти. |

**1.2. Иметь практический опыт – уметь – знать**

С целью овладения указанным видом профессиональной деятельности и соответствующими профессиональными компетенциями обучающийся в ходе освоения профессионального модуля должен:

**иметь практический опыт:**

* разработки алгоритма поставленной задачи и реализации его средствами автоматизированного проектирования;
* разработки кода программного продукта на основе готовой спецификации на уровне модуля;
* использования инструментальных средств на этапе отладки программного продукта;
* проведения тестирования программного модуля по определенному сценарию;

**уметь:**

* осуществлять разработку кода программного модуля на современных языках программирования;
* создавать программу по разработанному алгоритму как отдельный модуль;
* выполнять отладку и тестирование программы на уровне модуля;
* оформлять документацию на программные средства;
* использовать инструментальные средства для автоматизации оформления документации;

**знать:**

* основные этапы разработки программного обеспечения;
* основные принципы технологии структурного и объектно-ориентированного программирования;
* основные принципы отладки и тестирования программных продуктов;
* методы и средства разработки технической документации.

**2. Формы промежуточной аттестации по профессиональному модулю**

Таблица 3

|  |  |
| --- | --- |
| Элементы модуля, профессиональный модуль | Формы промежуточной аттестации |
| МДК 01.02. | Дифференцированный зачет |
| МДК 01.01 | Экзамен |
| УП.01.01 | Дифференцированный зачет |
| ПП.01.01 | Дифференцированный зачет |
| ПМ.01.ЭК | Экзамен квалификационный |

**3. Оценка освоения теоретического и практического курса профессионального модуля, типовые задания для оценки**

**3.1. Общие положения**

Основной целью оценки теоретического курса профессионального модуля является оценка умений и знаний.

Оценка теоретического курса профессионального модуля осуществляется с использованием следующих форм и методов контроля: Тестирование, самостоятельные работы, доклады, рефераты по междисциплинарному курсу. Экзамен квалификационный по профессиональному модулю.

**3.2. Типовые задания для оценки освоения МДК**

**3.2.1. Типовые задания для оценки освоения МДК 01.01.**

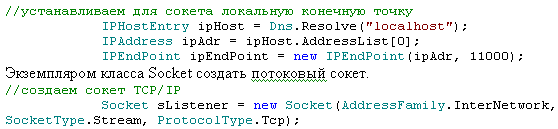
**Тема «Программирование сокетов»**

**Задание 1:** Создать приложение на потоковом сокете ТСР.

Сервер построен синхронно, выполнение потока блокируется, пока сервер не даст согласие на соединение с клиентом. Клиент завершает соединение с сервером, отправляя серверу сообщение <The End>.

Алгоритм:

Установить для сокета локальную конечную точку. Создать IPEndPoint для сервера, комбинируя первый IP адрес хост компьютера, полученный от метода Dns.Resolve (), и номер порта. Хост – компьютер –localhost, номер порта – 11000.



Перечисление AddressFamaly указывает схемы адресации для разрешения адреса.

Параметр SocketType поддерживает следующие параметры:

|  |  |
| --- | --- |
| Dgram | *Поддерживает дейтаграммы. Значение Dgram требует указать Udp для типа протокола и InterNetwork в параметре адресов.* |
| Raw | *Поддерживает доступ к базовому транспортному протоколу.* |
| Stream | *Поддерживает потоковые сокеты. Требует указать Тср для типа протокола и InterNetwork в параметре адресов.* |

Следующим шагом должно быть назначение сокета с помощью метода **Bind ().** Когда сокет открывается конструктором, ему не назначается имя, а резервируется дескриптор.

Чтобы сокет клиента мог идентифицировать потоковый сокет Тср, серверная программа должна дать имя своему сокету.

try

{

sListener.Bind(ipEndPoint);

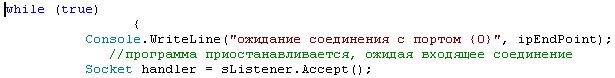
после создания и связывания имени, прослушиваем входящие сообщения.

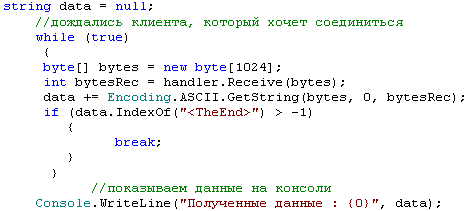
sListener.Listen(10);

в параметре определяется максимальное число соединений, ожидающих обработки в очереди.

В состоянии прослушивания надо быть готовым дать согласие на соединение с клиентом, для чего используется метод **Accept().** Он блокирует поток вызывающей программы до поступления соединения. Этот метод извлекает из очереди ожидающих запросов первый запрос и создает для его обработки новый сокет. Хотя новый сокет создан, первоначальный сокет продолжает слушать и может быть использован с многопоточной обработкой для приема нескольких запросов на соединение от клиентов. Серверное приложение не должно закрывать слушающий сокет. Он должен продолжать работать на ряду с сокетами, созданными методом Accept().

//начинаем слушать соединения

Когда клиент и сервер установили соединение, можно отправлять и получать сообщения, используя методы **Send ()** и **Receive()**.



Метод **Receive()** получает данные от сокета и заполняет массив байтов, переданный в качестве аргумента. Возвращаемое методом значение – фактически считанное количество байтов.

Если символы конца сообщения в строке не найдены, продолжается ожидание входящих данных, иначе сообщение отображается на консоли.

При выходе из цикла создаем новый массив байтов для ответа, который нужно послать клиенту.



При завершении обмена данными закрывается соединение методом **Close ()** . для гарантии, что никаких не обработанных данных не осталось перед этим методом, вызывается метод **Shutdown().**

Для каждого экземпляра сокета успешно выполнившего свою работу, обязательно нужно вызывать метод Close ().



SocketShutdown – это перечисление, содержащее три значения для остановки сокета.

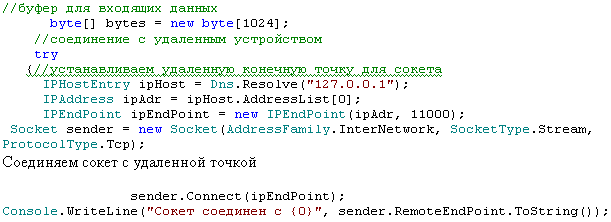
|  |  |
| --- | --- |
| Both | *Останавливает отправку и получение данных* |
| Receive | *Останавливает получение данных сокетом* |
| Send | *Останавливает отправку данных сокетом* |

Сокет закрывается при вызове метода Close (), который также устанавливает в свойстве **Connected** сокета значение **false**.

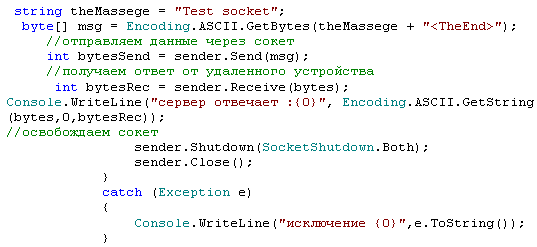
**Алгоритм построения Клиента на базе ТСР.**

Как и для сервера используются те же методы для определения конечной точки, создания экземпляра сокета, отправки и получения данных, закрытия сокета.

Сначала нужно установить удаленную конечную точку.



Получив сокет, метод Connect () устанавливает соединение между сокетом и удаленной точкой, заданной в параметре. Установив соединение, можно отправить данные и получить ответ.



**Задание 2**: Создать программу сканирования портов.

Программа пытается соединится с localhost по каждому порту, сообщается об успешности соединения, если соединение установить не удается перехватывается исключение SocketException.

Алгоритм:

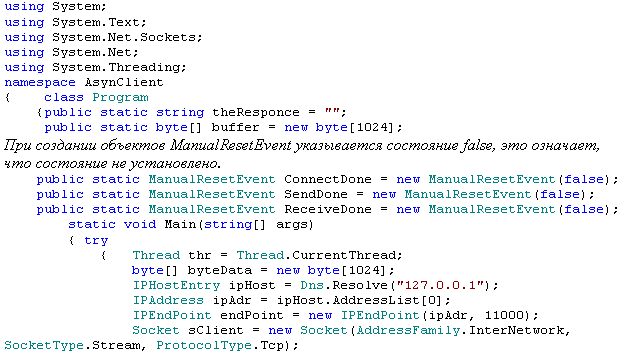
* Получение IP адреса локальной машины.
* Перебираем в цикле номера портов.
* Установление конечной точки. (в try)
* Создание потокового сокета.
* Соединение сокета с конечной точкой.
* Вывод сообщения о прослушивании порта.
* Обработка исключений (catch), объявить переменную типа SocketException
* Если код сообщения не равен 10061, что соответсвует сообщению о попытке соединения с закрытым портом, то выводится текст сообщения исключения

**Задание 3**: Создать клиент – серверное приложение, которое переданную информацию со стороны клиента считывает в файл.

**Задание 4**: Добавить в приложение подтверждения от сервера в ответ на переданные ему пакеты.

**Тема: «Асинхронное программирование сокетов»**

Задание 1: Создать асинхронное приложение- клиент.

*Метод BeginConnect начинает асинхронно устанавливать соединение с удаленным хостом. Для него требуется метод обратного вызова, в котором реализован представитель AsyncCallBack. Метод обратного вызова должен вызвать метод EndConnect(), который завершит запрос на соединение и вернет соединенный сокет.*

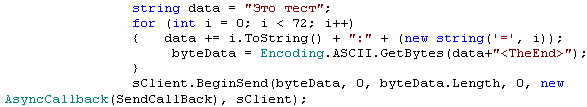
sClient.BeginConnect(endPoint, new AsyncCallback(ConnectCallBack), sClient);

*Этот метод блокирует текущий поток, пока объект не перейдет в сигнальное состояние.*

ConnectDone.WaitOne();

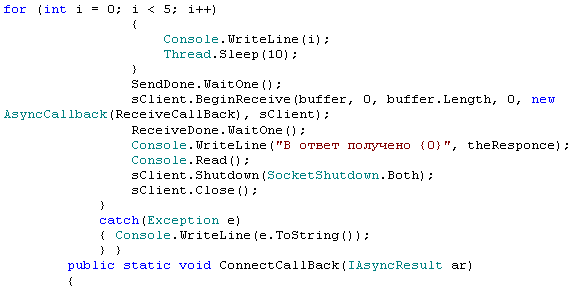
*Соединившись с сервером, приступаем к отправке данных.*

*Определим сообщение, которое пошлем серверу.*

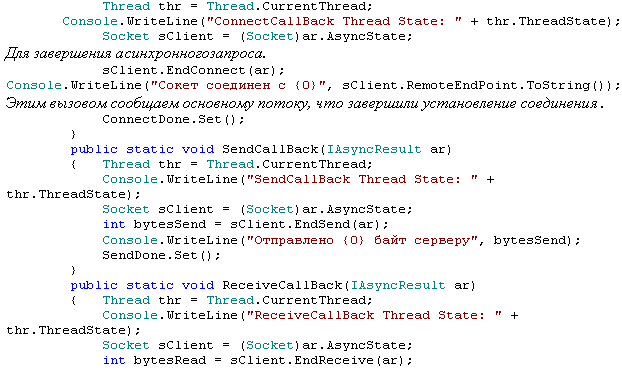


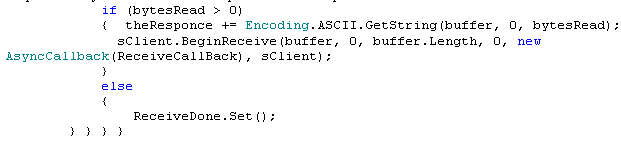
*Первый параметр – массив байтов, содержащий данные для отправки, второй –позиция в буффере, от которой нужно начать посылать данные, третий – размер буфера, последний используется для сохранения информации о состоянии. Метод BeginSend() вызывает функцию обратного вызова, переданную посредством делегата AsyncCallback.*

*Для иллюстрации асинхронной работы, выполним другую обработку. В цикле переводим текущий поток в ожидание на на сотую долю секунды – имитация вычислений тратящих процессорное время.*

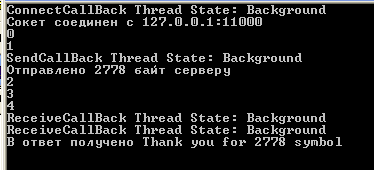


*Инспектируем поток, в котором выполняется асинхронный метод, он покажет, что выполнение метода идет в фоновом потоке. Параметр IAsyncResult содержит информацию о состоянии асинхронной операции.*

*Возвращает число отправленных байтов, поэтому проверяем, остались ли еще данные в очереди. Полученные данные сохраняем в theResponce.*



Вывод на консоль асинхронного – клиента, когда он запускается вместе с синхронным сервером, разработанным в предыдущей лабораторной работе, имеет следующий вид.

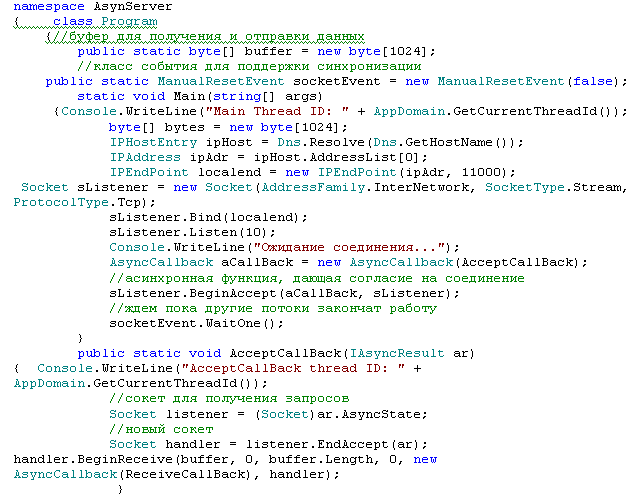
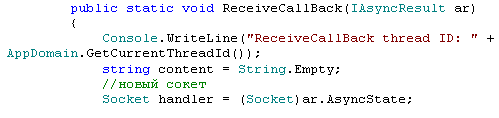


Задание -2 : Создать асинхронное приложение – сервер.

Для асинхронного приема соединений асинхронному приложению -серверу необходимо использовать метод BeginAccept(), а после установления соединения для получения и отправки данных используются методы BeginSend() и BeginReceive().

После вызова метода BeginAccept(), устанавливаем событие в режим ожидания, чтобы другой поток приложения мог выполняться пока не будет установлено соединение.

Если этого не сделать, то из – за асинхронной природы сервера, приложение раньше завершится прежде клиента.

//если данные есть ...

if (bytesRead > 0)

{

//присоединяем их к основной строке

content += Encoding.ASCII.GetString(buffer, 0, bytesRead);

//если мы находим символ конца сообщения <TheEnd>

if (content.IndexOf("<TheEnd") > -1)

{

Console.WriteLine("Считано {0} байт из сокета\n Данные: {1}", content.Length, content);

byte[] bytedata = Encoding.ASCII.GetBytes(content);

//отправляем данные обратно клиенту

handler.BeginSend(bytedata, 0, bytedata.Length, 0, new AsyncCallback(SendCallBack), handler);

}

else

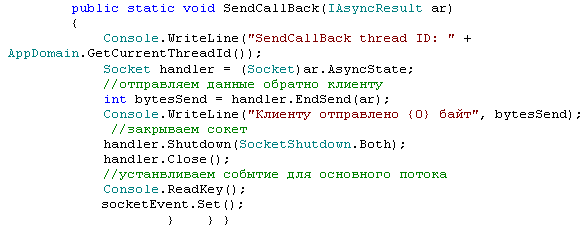
{//иначе получаем оставшиеся данные

handler.BeginReceive(buffer, 0, buffer.Length, 0, new AsyncCallback(ReceiveCallBack), handler);

}

}

}



Задание -3: Создать два клиента, передающих сообщения посредством сервера. Сервер ведет подсчет переданных данных и предоставляет их по требованию клиента.

Задание 4 :

Выполнить индивидуальное задание, полученное у преподавателя.

**Тема «Построение приложения на сокетах с использованием протокола ТСР»**

Задание 1: Создать многопоточное приложение. Сервер, используя потоки возвращает клиентам, получаемые от них сообщения.

Поддержку многопоточности обеспечивает пространство System.Threading, класс Thread представляет отдельный поток. Делегат ThreadStart представляет точку входа, с которой начнется выполнение потока. Метод Start() уведомляет операционную систему , что поток изменил состояние и начал выполняться.

Клиент –приложение.

using System;

using System.Collections.Generic;

using System.Text;

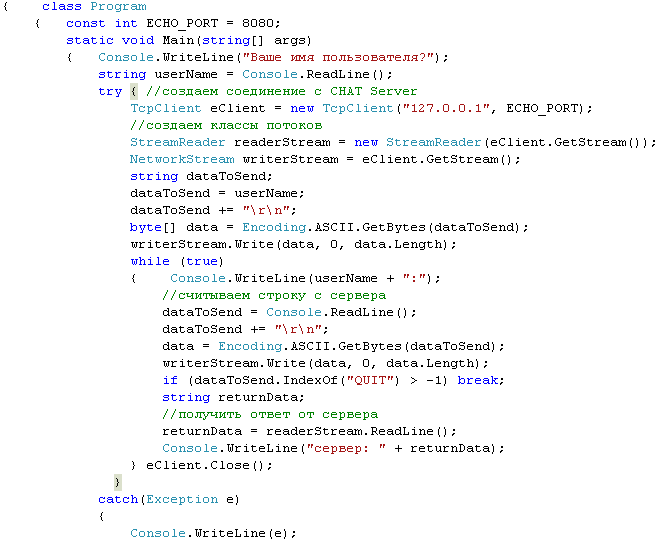
using System.Threading;

using System.Net;

using System.Net.Sockets;

using System.IO;

namespace ThreadKlient



} } }}

Приложение –сервер.

Класс TCPListener слушает входящие запросы от клиентов, принимает запрос и создает новый экземпляр от Socket или TCPClient для взаимодействия с клиентом. После создания сокета слушает клиентов, уже приступив к прослушиванию можно вызвать метод Pending(), чтобы проверять, нет ли ожидающих запросов о соединении.

Типичная серверная программа оперирует двумя сокетами, один слушает запросы, другой – для взаимодействия с клиентом. Чтобы дать согласие на любой запрос – метод AcceptTcpClient(). В многопоточном приложении дав согласие на соединение, сервер запускает новый поток, предназначенный для введения обмена с клиентом.

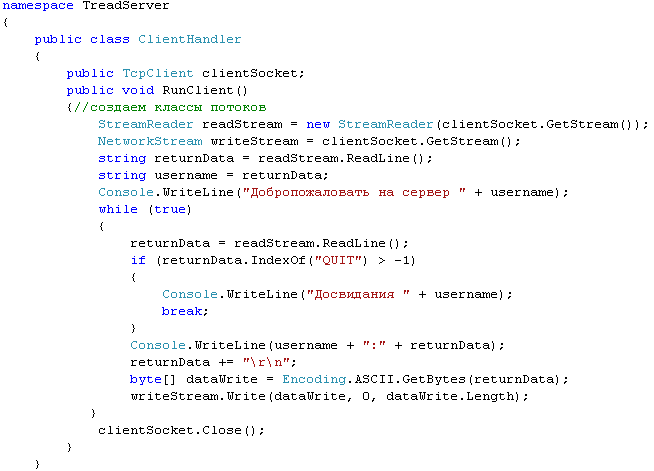
Основной поток сервера продолжает слушать следующие запросы.

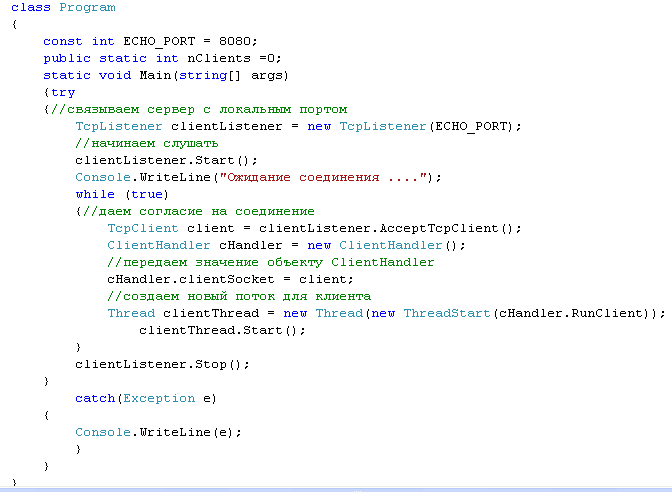
Выполняемый во вторичном потоке метод реализован в другом классе ClientHandler.

Вводим переменную, чтобы передавать данные между основным и вторичным потоком.

Метод RunClient() отвечает за весь обмен с одним клиентом. Он выполняет процедуру создания сокета, чтение и запись сообщений в сокет.

Листинг кода для серверного приложения.





Протестируйте многопоточное приложение, запустив сначала сервер, а затем несколько клиентов, введите разные имена пользователей и проследите взаимодействие клиентов и сервера.

**Лабораторная работа № 5  
Тема: «Создание приложений интерактивного форума, использующее UDP »**

Запустите два экземпляра приложения, для тестирования на локальной машине в качестве номера локального порта выберите 5001, а для удаленного 5002, IP- адрес 127.0.0.1, во втором приложении локальный порт – 5002, удаленный – 5001, IP- адрес 127.0.0.1.

**Приложение передачи файлов**

Создадим приложение для передачи файлов и сериализованного объекта. Программы отправителя и получателя разделены на две логические части. В первой части отправитель посылает получателю информацию о файле как сериализованный объект, а во второй части отправляет сам файл.

В получателе в первой части принимает сериализованный объект с соответсвующей информацией, а вторая часть создает файл на машине получателя. Сохраненный файл будем открывать соответствующей программой(например, \*. doc в Word, а \*.htm – браузером Internet Explorer ).

Файловый сервер.

Файловый сервер – это простое консольное приложение, реализованное в классе FileSender. В этом классе есть вложенный класс FileDetails, содержащий информацию о файле – тип и размер файла. Начнем с импорта необходимых пространств имен и объявления полей класса. В классе есть память закрытых полей: экземпляр класса

FileDetails, Объект UdpClient, а также информация о соединении с удаленным клиентом и объект FileStream для считывания файла, который отправляется клиенту.

Чтобы послать через сеть объект FileDetails, его требуется сериализовать, поэтому добавляем атрибут [Serializable].

Приглашаем пользователя ввести удаленный IP-адрес, по которому нужно отправить файл, путь и имя отправляемого файла. Открываем этот файл в объекте FileStream и определяем его длину. Если она больше максимально допустимой длины, равной 8192 байтам, закрываем UdpClient и FileStream и выходим из приложения. Иначе отправляем информацию о файле, выжидаем две секунды, вызвав метод Thread.Sleep(), и отправляем сам файл. Метод SendFileInfo() заполняет поля объекта FileDetails, а затем сериализует объект в MemoriStream, используя объект XmlSerializer. Этот объект считывается в массив байтов и передается методу Send() класса UdpClient, который отправляет информацию о файле клиенту. Метод SendFile() просто считывает содержимое файла из FileStream в массив байтов и отправляет его клиенту.

using System.IO;

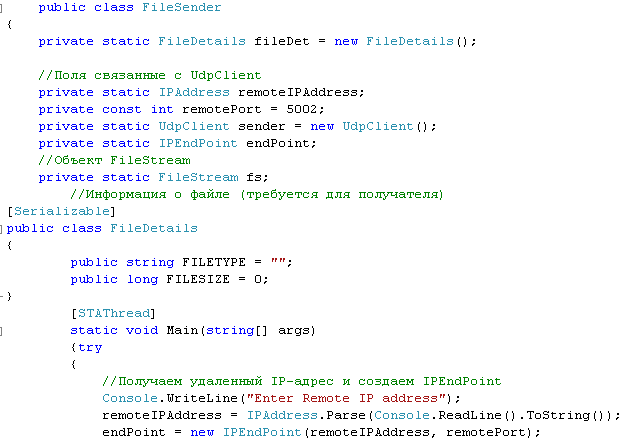
using System.Threading;

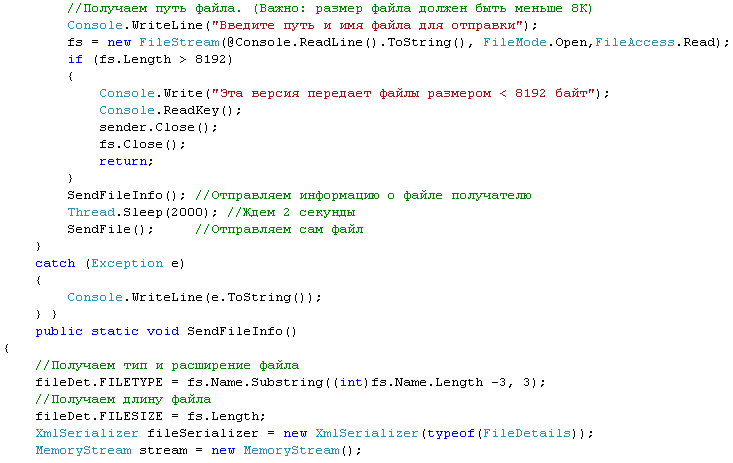
using System.Xml.Serialization;

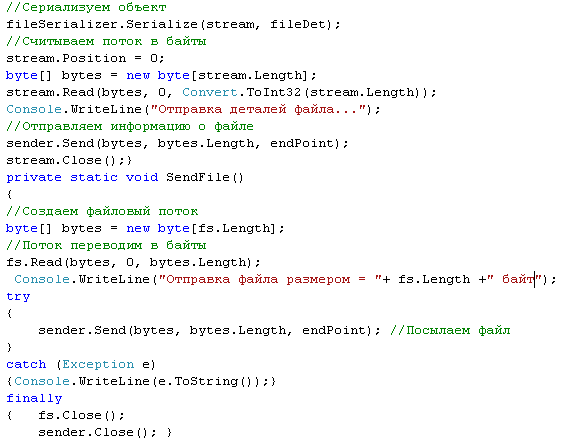
using System.Net;

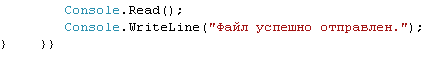
using System.Net.Sockets;

using System.Diagnostics;









Приемник файла

Приемник файла – тоже консольное приложение, реализованное в классе FileRecv. Здесь так же начинаем с экспорта необходимых пространств имен и объявление полей класса. Потребуется десериализовать информацию о файле, отправленную сервером, в объект FileDetals, поэтому нужно определить этот класс и в приложение – клиенте. Метод Main () этого приложение только называет два метода, чтобы получить, соответственно, информацию о файле и сам файл. Метод Main () этого приложение только называет два метода, чтобы получить, соответственно, информацию о файле и сам файл. Метод ReceiveFile() получает файл от сервера и сохраняет его на диске под именем temp, добавляя расширение, извлеченное из объекта FileDetails. Затем вызываем статический метод Process.Start() и открываем документ связанной с расширением программой.

using System.Xml.Serialization;

using System.Net;

using System.Net.Sockets;

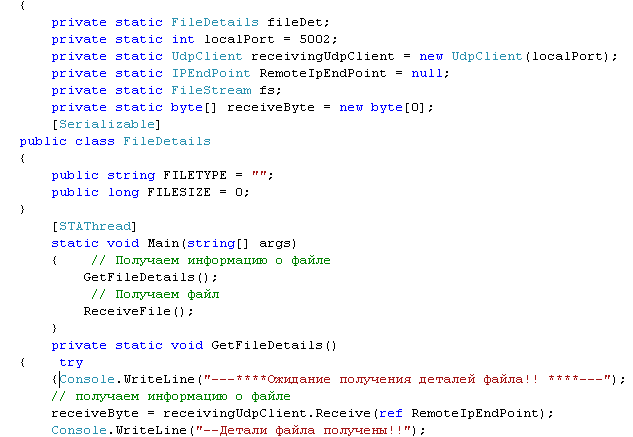
using System.Diagnostics;

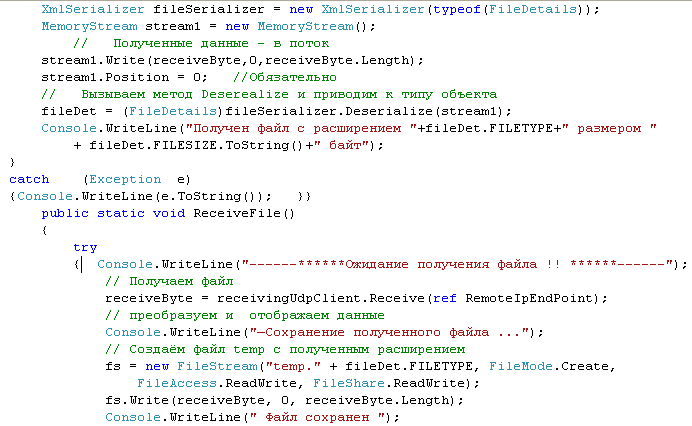
using System.IO;

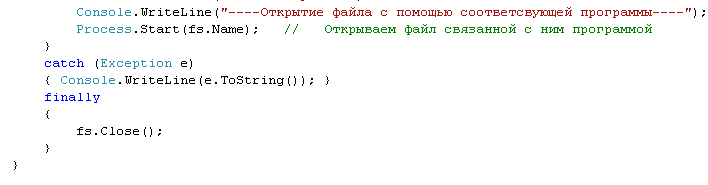
namespace FileRecs

{

class Program







3.2.**2. Типовые задания для оценки освоения МДК 01.02.**

Тема: «Знакомство со средой программирования Visual Basic»

1. Запустите среду проектирования Visual Basic стандартным способом.

2. Войдите в меню команды «File» – нажмите «Alt+F». Ознакомьтесь с командами находящимися в этой опции. Выйдите из меню при помощи кнопки «Esc».

3. В Окне проводника проекта (Project Explorer) щелкните мышью строку с именем файла проекта. Откроется «Окно свойств проекта» (Properties). Значение его свойства «Name» замените – слово «Project» на имя «Пример». Аналогично значение свойства «Name» экранной формы – заменить слово «Form1» на «Пример».

4. Если окно «Toolbox» отсутствует, то выполните следующие операции: «\View\ Toolbox» и рассмотрите инструменты окна.

5. В поле свойства «Caption» наберите «Вычисление функции», а в поле «Name» - «Моя форма». Для установки размеров экранной формы можно или «ухватить» мышью сторону формы и растянуть ее по горизонтали и вертикали, или установить в окне «Свойства» нужные значения свойств «Width» (ширина) и «Height» (высота). Эти величины измеряются в твипах – это 1/1440 логического дюйма. Логический дюйм – это такое расстояние на экранной форме, которое при ее печати на принтере будет равно 1 дюйму (приблизительно 2,54 см). Далее расположите экранную форму в нужном месте монитора. Это можно сделать на «глазок» с помощью мыши или с помощью установки значений свойств «Left» (левый край) и «Top» (верхний край). Необходимые сведения о положении и размерах экранной формы можно найти на индикаторе положения и размеров объекта, расположенном в правой части линейки инструментов «Главной панели проекта». То, что форма покрыта сетью точек, очень помогает при размещении на ней всевозможных объектов управления.

6. Установите в окне «Свойства» значение «Width» – 1440, «Height» – 1440. Переместите экранную форму по экрану, ухватив ее мышью за заголовок.

7. Создайте приложение, в котором после запуска на форме печатается некоторый текст, например, «Первый проект на языке Visual Basic».

В первую очередь создайте заготовку проекта, свойству «Caption» формы «Form1» присвойте значение «Первый проект». Вывод на форму текстовых сообщений можно производить различными способами:

* с помощью элемента управления «Label» (Метка);
* с помощью элемента управления «TextBox» (Текстовое поле);
* используя «метод Print».

Выберите на «Панели инструментов» класс управляющих элементов «Label» и разместите экземпляр метки «Label l» на форме «Form1». Активизируйте метку «Label l». В окне «Свойств объекта» выберите свойство «Caption» и присвойте ему значение «Первый проект на языке Visual Basic».

Выберите на «Панели инструментов» класс управляющих элементов «TextBox» и разместите экземпляр текстового поля «Text l» на форме «Form1». Активизируйте текстовое поле «Text l». В окне «Свойств объекта» выберите свойство «Caption» и присвойте ему значение "Первый проект на языке Visual Basic".

Для вывода на форму текстового сообщения с помощью метода «Print» необходимо воспользоваться событийной процедурой. Для создания событийной процедуры выберите на «Панели инструментов» класс управляющих элементов «CommandButton» и разместите экземпляр кнопки «Commandl» на форме «Form1». Активизируйте кнопку «Commandl». В окне «Свойств объекта» выберите свойство «Caption» и присвойте ему значение "Метод Print". Двойным щелчком по кнопке «Commandl» вызовите окно «Программного кода» с пустой процедурой «Commandl\_Click()». Введите в процедуру следующий код: Form1.Print "Первый проект на языке Visual Basic". Введите команду «Run-Start». Появится окно приложения «Первый проект». Для выполнения событийной процедуры щелкните по кнопке «Метод Print».

8. Сделайте внешний вид проекта более привлекательным для чего измените свойства объектов, определяющие их внешний вид (цвет фона формы, цвет, размер и выравнивание шрифта на метке, в текстовом поле и в «методе Print»).

Активизируйте форму «Forml». В окне «Свойств объекта» выберите свойство «BackColor» (цвет фона) и двойным щелчком откройте диалоговое окно с цветовой палитрой. Выберите цвет «желтый».

Активизируйте метку «Label l». В окне «Свойств объекта» для свойств установите значения: «BackColor» - «зеленый», «ForeColor» (цвет надписи) - «синий», «Font» - «размер шрифта 18», «Alignment» (выравнивание) - «center».

Активизируйте текстовое поле «Text l». В окне «Свойств объекта» для свойств установите значения: «ForeColor» (цвет надписи) - «красный», «Font» - размер шрифта «14» и начертание «курсив». «Alignment» (выравнивание) - «center».

Для установки параметров шрифта, которые используются в «методе Print», необходимо для шрифта (объект «Font») установить требуемые значения его свойств (размер шрифта, начертание и др.). Например, установите размер шрифта 12:

Font.Size = 12.

Для того чтобы новые установки свойств шрифта вступили в действие, они должны быть вставлены в событийную процедуру перед «методом Print». Двойным щелчком по кнопке «Commandl» вызовите окно «Программного кода» с процедурой «Commandl\_Click ()». Введите в процедуру строку «Font. Size =12». После запуска проекта и щелчка по кнопке «Meтод Print» вы получите новый вид графического интерфейса.

9. Сохраните файл формы командой «Save Form1.frm As …» как «frm1.frm» и файл проекта командой «Save Project As …» как «prj1.vbp».

Сохраненный проект может выполняться только в системе программирования Visual Basic. Для того чтобы проект мог запускаться в среде операционной системы необходимо сохранить его в исполняемом файле (типа EXE). Для трансляции проекта в исполняемый файл используется команда «File-Make…».

Тема: «Разработка линейных программ»

Построить простую прикладную программу в среде программирования Visual Basic – решение квадратного уравнения. Введем ограничение: дискриминант число положительное (коэффициенты *а* и *с* разных знаков).

1. Необходимо составить блок-схему алгоритма и зарисовать ее в рабочей тетради.
2. Необходимо создать экранную форму приложения. Пользователь после запуска приложения увидит на экране окно, типичное для Windows-приложений. Окно содержит стандартные элементы: кнопки минимизации, распахивания и закрытия окна, строку заголовка. Кроме этих элементов проектировщику приложения нужно позаботиться о нестандартных элементах. Это название приложения, которое будет содержаться в строке заголовка, например, «Решение квадратного уравнения». В окне приложения должны быть специальные окошки для ввода исходных данных: коэффициентов квадратного уравнения.

После ввода исходных данных пользователь с помощью мыши «щелкнет» командную кнопку, которая также должна находиться на экранной форме. Сразу после этого в специальных окнах должен появиться результат. Это сведения о значении дискриминанта и числовые значения корней.

Чтобы пользователь знал, для каких именно данных предназначено то или иное окно, рядом с каждым из них должна стоять пояснительная надпись – метка. Также с помощью метки на экранную форму будут выведены формулы для нахождения корней квадратного уравнения.

В рабочей тетради нарисуйте эскиз будущей экранной формы. Один из возможных вариантов эскиза экранной формы представлен на рисунке 2.1.

3. Войдите в систему проектирования Visual Basic 6.0. Создайте окно Windows-приложения, на котором будут размещены 8 надписей (меток), 3 поля для ввода исходных данных и 3 поля для вывода сведений о дискриминанте и значения корней квадратного уравнения, а также такой важный элемент, как командная кнопка. Дополнительным заданием является использование дополнительной кнопки «Exit» – завершение работы.



Рисунок 2.1 – Эскиз экранной формы (вариант)

4. В задании есть только одно событие: щелчок мышью командной кнопки. Поэтому составляется программный код только одной процедуры, которая относится к данному событию. Для этого необходимо открыть «Окно программного кода». Сделать это можно, выбрав команду «Code» меню «View». В верхней части этого окна есть две строки: левая – список уже созданных объектов и правая – список еще не существующих процедур. **Процедура** – это фрагмент программного кода, с помощью которого решается какая-то локальная задача.

Щелкнув мышью по списку объектов, можно увидеть идентификаторы всех созданных объектов. Выберите идентификатор «Командная кнопка» и щелкните по нему мышью. Раскроется список процедур, а в окне программного кода появятся первая и последняя строки процедуры для самого распространенного события «Click», относящегося к объекту «Командная кнопка».

Первая строка программы начинается словами «Private Sub», а заканчивается словами «End Sub». С помощью клавиатуры поместите между этими двумя строками строки программного кода:

A=val (Коэффициент1.Text) «Функция val преобразует текст в число»

B=val (Коэффициент2.Text)

C=val (Коэффициент3.Text)

D=B^2-4\*A\*C

X1=(-B+SQR(D))/(2\*A)

X2=(-B-SQR (D))/(2\*A)

Дискриминант.Text=str(D) «Функция str преобразует число в текст»

Корень1.Text=str(xl)

Kopeнь2.Text=str(x2)

Если система Visual Basic не выдала никаких предупреждений о синтаксических ошибках, то программу можно запустить на выполнение:

• с помощью кнопки «Run» и команды «Start» главной панели проекта;

• с помощью кнопки «►» (Start) линейки инструментов той же панели;

• с помощью клавиши F5 клавиатуры.

После ввода исходных данных и щелчка мышью по командной кнопке, сработает процедура и в заданных полях появятся результаты.

Отчетностью за занятие являются – блок-схема алгоритма программы, эскиз экранной формы разрабатываемого проекта и действующая программа, запускаемая с помощью выполняемого файла.

Тема: Программирование ветвлений

Разработать проект, который позволит контролировать знания обучаемых по изучаемой дисциплине. Алгоритм контроля должен последовательно реализовывать следующие операции:

• регистрация;

• задание (напечатать) вопроса;

• запрос ответа и запоминание введенного с клавиатуры значения;

• сравнение полученного ответа с эталонным и, в зависимости от выполнения или невыполнения этого условия, реализация различных действий.

Регистрация проверяемого реализуется с использованием функций «InputBox» и «MsgBox».

1. Разместить на форме кнопку «cmdl» и начать создание событийной процедуры «cmdl\_Click()».

2. С помощью функции «InputBox» запросить имя, фамилию и присвоить ее значение строковой переменной «strA», а с помощью функции «MsgBox» вывести результаты регистрации:

Private Sub cmdl\_Click()

strA = InputBox("Введите ваше имя и фамилию:",\_"Регистрация")

bytB *=* MsgBox ("Уважаемый " + strA + ", Вы готовы к проверке знаний?", 36, «Конец регистрации")

End Sub

3. Запустить проект и щелкнуть по кнопке «Начать проверку»*.* На появившемся диалоговом окне «Регистрация» ввести в текстовое поле имя и фамилию.

В функции «MsgBox» второму аргументу присвоено числовое значение, которое одновременно задает тип выводимого информационного окна и набор размещенных на нем кнопок.

4. Число 36 обеспечивает вывод информационного окна типа «Вопрос»*,* которое имеет две кнопки «Да» и «Нет»*.* Щелчок по одной из кнопок приводит к возвращению функцией определенного числового значения («Да» *—* 6, «Нет» *—* 7), которое присваивается числовой переменной «bytB».

5. С помощью условного оператора можно реализовать либо выход из программы (щелчок по кнопке «Нет»*),* либо продолжение работы и переход к проверке знаний (щелчок по кнопке «Да»):

If bytB = 7 Then End

Проверка знаний.

Вопрос задается с помощью функции «InputBox», проверка правильности ответа производится с помощью оператора условного перехода: «If-Then-Else», а вывод информации о правильности или неправильности ответа с помощью функции «MsgBox» в форме оператора (бесскобочная запись) и числовым значением аргумента 0, что обеспечивает вывод информационного окна одной кнопкой «ОК»*.*

1. Ввести в событийную процедуру программный код, реализующий проверку знаний в виде последовательности вопросов. В переменной «bytN» накапливать количество неправильных ответов:

strC = InputBox("Чему равен 1 байт?:", "Первый вопрос")

If strC = "8 бит" Then MsgBox "Правильно!", 0, "Первый вопрос"

Else MsgBox "Неправильно!", 0, "Первый вопрос":

bytN = bytN + 1

strC = InputBox("Переведите десятичное число 5

в двоичную систему счисления:", "Второй вопрос")

If strC = "101" Then MsgBox "Правильно!", 0, "Второй вопрос" \_

Else MsgBox "Неправильно!", 0, "Первый вопрос":

bytN = bytN + 1

MsgBox "Уважаемый " + strA + ", Вы сделали " +

Str(bytN) + " ошибок!", 0, "Конец опроса"

2. Запустить проект, пройти регистрацию и ответить на вопросы. Результат будет выведен с помощью информационного окна функции «MsgBox». Вывод результатов контроля знаний можно реализовать также с использованием текстовых полей.

3. Разместить на форме два текстовых поля «txtName» и «txtError» и две поясняющие метки «LblName» и «LblError».

4. Присвоить свойству «Text» текстовых полей значения переменных «strA» (содержит регистрационные данные) и «bytN» (содержит количество ошибок):

txtName.Text = strA

txtError.Text = Str(bytN)

5. Запустить проект, пройти регистрацию и проверку знаний. На форме должны появиться результаты, представленные на рисунке 3.1.

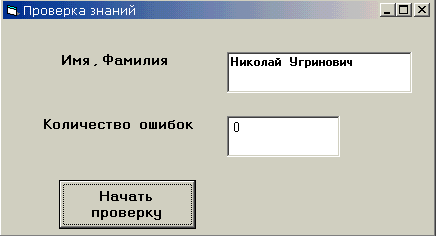


Рисунок 3.1 – Экранная форма тестирующей программы

Тема: «Исследование циклических алгоритмов в Visual Basic»

Вычислить значение функции F(x) по вариантам, представленным в таблице 4.1, используя для вычисления приближенные выражения (см. табл. 4.2) с точностью . Суммирование членов ряда прекратить, если очередной член ряда, прибавляемый к сумме, будет меньше . Пример экранной формы разрабатываемого проекта представлен на рисунке 4.1.



Рисунок 4.1– Образец экранной формы разрабатываемого проекта

Таблица 4.1 – Варианты заданий

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № варианта | F(x) | x |  | № варианта | F(x) | x |
| 1 |  | 4 |  | 14 |  | 4 |
| 2 |  | 4 |  | 15 |  |  |
| 3 |  | 4 |  | 16 |  |  |
| 4 |  | 4 |  | 17 |  |  |
| 5 |  | 4 |  | 18 |  |  |
| 6 |  | 4 |  | 19 |  |  |
| 7 |  | 4 |  | 20 |  |  |
| 8 |  | 4 |  | 21 |  |  |
| 9 |  | 4 |  | 22 |  |  |
| 10 |  | 4 |  | 23 |  |  |
| 11 |  | 4 |  | 24 |  |  |
| 12 |  | 4 |  | 25 |  |  |
| 13 |  | 4 |  |  |  |  |

Таблица 4.2 – Формулы приближенных вычислений

|  |  |  |
| --- | --- | --- |
|  |  |  |

Тема: «Обработка массивов»

Написать программу, позволяющую для матрицы размером 5×5 выполнить операции из таблицы 5.1 согласно списочного номера. Пример экранной формы разрабатываемого проекта представлен на рисунке 5.1.



Рисунок 5.1 – Образец экранной формы разрабатываемого проекта

Таблица 5.1 – Варианты заданий

|  |  |
| --- | --- |
| № варианта | Задание |
| 1 | Найти определитель матрицы |
| 2 | Транспонировать матрицу |
| 3 | Найти наибольшую сумму элементов столбца |
| 4 | Найти наименьшее произведение элементов строк |
| 5 | Найти наименьшую сумму элементов столбца |
| 6 | Найти наибольшую сумму элементов строки |
| 7 | Найти наименьшую сумму элементов строки |
| 8 | Поменять местами 1 строку и последний столбец |
| 9 | Поменять местами 2 строку и 2 столбец |
| 10 | Найти наибольшее произведение элементов строк |
| 11 | Найти наибольшее произведение элементов |
| 12 | Вычесть из первого столбца последнюю строку |
| 13 | Поменять местами первый столбец и последнюю строку |
| 14 | Сложить каждый столбец с каждой строкой |
| 15 | Найти сумму максимальных элементов строк |
| 16 | Найти произведение минимальных элементов столбцов |

Окончание таблицы 5.1

|  |  |
| --- | --- |
| № варианта | Задание |
| 17 | Найти произведение максимальных элементов строк |
| 18 | Найти сумму минимальных элементов столбцов |
| 19 | Умножить матрицу на ее определитель |
| 20 | Найти определитель матрицы |
| 21 | Транспонировать матрицу |
| 22 | Найти наибольшую сумму элементов столбца |
| 23 | Найти наименьшее произведение элементов строк |
| 24 | Найти наименьшую сумму элементов столбца |
| 25 | Найти наибольшую сумму элементов строки |

**4. Оценка по учебной и (или) производственной практике**

**4.1. Общие положения**

Целью оценки по учебной и (или) производственной практике является оценка: 1) практического опыта и умений; 2) профессиональных и общих компетенций.

Оценка по учебной и (или) производственной практике (по профилю специальности) выставляется на основании данных аттестационного листа (характеристики учебной и профессиональной деятельности обучающегося/студента на практике) с указанием видов работ, выполненных обучающимся во время практики, их объема, качества выполнения в соответствии с технологией и (или) требованиями организации, в которой проходила практика, либо образовательного учреждения (для учебной практики).

**4.2. Виды работ практики и проверяемые результаты обучения по профессиональному модулю**

**4.2.1. Учебная практика**

Таблица 4

|  |  |
| --- | --- |
| **Содержание учебного материала, виды работ** | **Проверяемые результаты** |
|  |  |
| Технологический процесс подготовки и выполнения программ на языке ассемблер.  Регистры, память и логическая адресация.  Реальный режим. Защищённый режим.  Режимы MASM и Ideal при использовании Turbo Assembler.  Стандартные и упрощённые директивы сегментов.  Способы адресации операндов машинных команд.  Режимы адресации, приводящие к образованию 32-битовых адресов  Использование общих данных.  Передача параметров.  Обработка прерываний. Структуры и особенности работы обработчиков прерываний.  Системные средства распределения памяти. Организация дочерних процессов.  Программы, резидентные в памяти.  Программирование операций над файлами, каталогами и дисками.  Защита программ от копирования и несанкционированного доступа.  Структура макроопределения. Макрокоманды. Макрорасширения.  Псевдооператоры макроассемблера.  Преобразование ASCII-формата в двоично-десятичный формат и обратно.  Арифметические операции над данными в ASCII - и BCD –форматах  Форматы представления чисел.  Команды пересылки данных, арифметических операций, сравнения, специальных операций, управления FPU.  Turbo Assembler и Borland С++.  Turbo Assembler и Borland Pascal. | ОК 1 Понимать сущность и социальную значимость своей будущей профессии, проявлять к ней устойчивый интерес.  ОК 2 Организовывать собственную деятельность, выбирать типовые методы и способы выполнения профессиональных задач, оценивать их эффективность и качество.  ОК 3 Принимать решения в стандартных и нестандартных ситуациях и нести за них ответственность.  ОК 4 Осуществлять поиск и использование информации, необходимой для эффективного выполнения профессиональных задач, профессионального и личностного развития.  ОК 5 Использовать информационно-коммуникационные технологии в профессиональной деятельности.  ОК 6 Работать в коллективе и в команде, эффективно общаться с коллегами, руководством, потребителями.  ОК 7 Брать на себя ответственность за работу членов команды (подчиненных), за результат выполнения заданий.  ОК 8 Самостоятельно определять задачи профессионального и личностного развития, заниматься самообразованием, осознанно планировать повышение квалификации.  ОК 9 Ориентироваться в условиях частой смены технологий в профессиональной деятельности.  ПК 1.1 Выполнять разработку спецификаций отдельных компонент.  ПК 1.2 Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля.  ПК 1.3 Выполнять отладку программных модулей с использованием специализированных программных средств.  ПК 1.4 Выполнять тестирование программных модулей.  ПК 1.5 Осуществлять оптимизацию программного кода модуля.  ПК 1.6 Разрабатывать компоненты проектной и технической документации с использованием графических языков спецификаций. |

**Вопросы для защиты производственной практики (по профилю специальности)**

Ознакомление с перечнем и конфигурацией средств вычислительной техники, архитектурой сети.

Ознакомление перечня и назначения программных средств, установленных на ПК предприятия.

Разработка требований к программному продукту.

Форматы данных микропроцессора. Числа. Символы. Указатели.цепочки.

Адресация памяти. Внутренние регистры процессора. Регистры общего назначения. Сегментные регистры.

Регистры смещения. Регистр флагов. Режимы адресации. Регистровая адресация.

Непосредственная адресация. Прямая адресация. Косвенная регистровая адресация. Базовая адресация.

Прямая адресация с индексированием. Базовая адресация с индексированием.

Система команд микропроцессора. Команды пересылки данных.

Арифметические команды. Логические команды. Команды передачи управления. Цепочечные (строковые) команды.

Директивы и операторы ассемблера. Структура программы. Организация программы.

Модели памяти. Процедуры.

Директивы задания набора допустимых команд. Примеры использования директив в программах типа .exe и .com.

Архитектура и система команд арифметического сопроцессора. Форматы чисел сопроцессора. Целые числа. Вещественные числа

Диапазоны вещественных чисел в х87. Особые случаи вещественной арифметики. Формирование специальных значений в особых случаях.

Случай неточного результата. Численное антипереполнение. Денормализованный операнд. Деление на ноль. Численное переполнение. Недействительная операция. Регистры математического сопроцессора.

Численные регистры (регистровый стек). Регистр управления (cw). Регистр состояния.

Разработка документации

Оценка результатов производственной практики

**Критерии оценки отчета о прохождении практики:**

|  |  |
| --- | --- |
| - отражение обязательных вопросов в отчете: характеристика предприятия, анализ аппаратного обеспечения, анализ программного обеспечения; ТЗ; описаны входные и выходные данные; спроектированы и документированы диаграммы потоков данных, концептуальная, инфологическая и даталогическая модели данных, алгоритмы обработки данных; реализованы хранилища данных;  описано тестирование разработанных модулей. | 1 балл |
| - наличие правильно оформленного дневника | 1 балл |
| - наличие в отчете результатов решения профессиональной задачи (скриншоты программы, тексты программных модулей, работающее приложение на CD). | 1 балл |
| - глубина и полнота изложения и наглядность отражения результатов работы; | 0,5 балла |
| - логичность, структурированность, связанность изложения материала; | 0,5 балла |
| - оформление (наличие плана, списка литературы, соблюдение правил оформления и т.д.); | 0,5 балла |
| - стилистическая грамотность в изложении материала; | 0,5 балла |
| - соблюдение научного стиля изложения; | 0,5 балла |
| - наличие выводов по результатам анализа. | 0,5 балла |
| **Результаты оценивания** | |
| 5 баллов | Отлично |
| 4 балла | Хорошо |
| 3 балла | Удовлетворительно |
| Менее 3 баллов | Неудовлетворительно |

**4.3**. Положением о квалификационном экзамене (утвержденном приказом №29а от 20.09.2018 г и одобренном решением педагогического совета СмК протокол №6 от 19.09.2018 г.) определена форма аттестационного листа практики производственной (по модулю).

**5. Контрольно-оценочные материалы для экзамена (квалификационного)**

**5.1. Общие положения**

Экзамен (квалификационный) предназначен для контроля и оценки результатов освоения профессионального модуля ПМ01 «Разработка программных модулей программного обеспечения для компьютерных систем»

специальности СПО: 09.02.03 Программирование в компьютерных системах

Экзамен включает: проверку теоретических и практических знаний и умений

Итогом экзамена является однозначное решение: «вид профессиональной деятельности освоен (с оценкой) / не освоен».

Положением о квалификационном экзамене (утвержденном приказом №29а от 20.09.2018 г и одобренном решением педагогического совета СмК протокол №6 от 19.09.2018 г.) определены формы:

* оценочный лист экзамена квалификационного на каждого обучающегося;
* для экзаменующегося текст задания, с инструкцией для выполнения;
* протокол заседания аттестационной комиссии.

**5.2. Выполнение заданий в ходе экзамена квалификационного**

**Вариант 1.**

1. Основные понятия и определения
2. Построить простую прикладную программу в среде программирования Visual Basic – решение квадратного уравнения
3. Написать калькулятор на языке с простейшими вычислительными функциями (+, -, \*, /).

**Вариант 2.**

1. Команды обработки строк
2. Разработать проект, который позволит контролировать знания обучаемых по изучаемой дисциплине. Алгоритм контроля должен последовательно реализовывать следующие операции:

• регистрация;

• задание (напечатать) вопроса;

• запрос ответа и запоминание введенного с клавиатуры значения;

• сравнение полученного ответа с эталонным и, в зависимости от выполнения или невыполнения этого условия, реализация различных действий.

1. Создать программу для перевода температуры из различных единиц измерения (C, F, K).

**Вариант 3.**

1. Свойство операторов работы с экраном
2. Вычислить значение функции F(x)

|  |  |
| --- | --- |
| F(x) | x |
|  | 4 |

используя для вычисления приближенные выражения

|  |  |  |
| --- | --- | --- |
|  |  |  |

с точностью . Суммирование членов ряда прекратить, если очередной член ряда, прибавляемый к сумме, будет меньше .

1. Написать калькулятор для возведения в степень.

**Вариант 4.**

1. Ввод и выполнение программ
2. Вычислить значение функции F(x)

|  |  |
| --- | --- |
| F(x) | x |
|  | 4 |

используя для вычисления приближенные выражения

|  |  |  |
| --- | --- | --- |
|  |  |  |

с точностью . Суммирование членов ряда прекратить, если очередной член ряда, прибавляемый к сумме, будет меньше .

1. Создать программу для вывода сообщения «Hello Word!» на экран.

**Вариант 5.**

1. Алгоритмы работы Ассемблеров
2. Вычислить значение функции F(x)

|  |  |
| --- | --- |
| F(x) | x |
|  |  |

используя для вычисления приближенные выражения

|  |  |  |
| --- | --- | --- |
|  |  |  |

с точностью . Суммирование членов ряда прекратить, если очередной член ряда, прибавляемый к сумме, будет меньше .

1. Создать программу для вывода случайных целых чисел от 1 до 100 (10шт).

**Вариант 6.**

1. Логика и организация программы
2. Создать приложение «Курс валют» с обновлением данных из интернета. (Даллар США, Китайский юань)
3. Создать программу сортировки целых чисел от меньшего к большему.

**Вариант 7.**

1. Компоновка программы, выполнение программ
2. Создать приложение «Курс валют» с обновлением данных из интернета. (Австралийский доллар, Японская йена)
3. Создать программу сортировки целых чисел от большему к меньшему.

**Вариант 8.**

1. Макросредства, макропроцессоры
2. Создать приложение «Курс валют» с обновлением данных из интернета. (ЕВРО, Английский фунт стерлингов)
3. Написать калькулятор для вычисления квадратного корня.

**Вариант 9.**

* 1. Кросс-системы

1. Создание программы, использующей графические компоненты, для автоматической замены фона рабочего стола (по таймеру).
2. Создать программу для вычисления минимального значения ряда чисел.

**Вариант 10.**

1. Макроассемблер
2. Разработать программу, реализующую игру «Тетрис».
3. Создать программу для вычисления максимального значения ряда чисел.

**Вариант 11.**

1. Команда прерывания INT
2. Разработать программу, позволяющею создавать слайд-шоу.
3. Создать программу для вычисления среднего значения ряда чисел.

**Вариант 12.**

1. Директивы ассемблера
2. Создать DLL для проверки имени компьютера и номера жесткого диска.
3. Написать программу, которая выводит текущую дату в MessageBox.

**Вариант 13.**

1. Основные понятия макропроцессора
2. Создать приложение, которое будет осуществлять создание и перемещение случайных фигур в пределах окна приложения.
3. Написать программу для выключения ПК через заданное количество времени (1 минута).

**Вариант 14.**

1. Опции MASM
2. Создать многомерный массив (5х5) случайных чисел и его сортировка по возрастанию.
3. Написать программу, которая выводит на экран имя компьютера и текущего пользователя.

**Вариант 15.**

1. Сегментация программы
2. Создать программу, которая автоматически, с заданным пользователем интервалом, копирует указанный файл в архив zip
3. Написать программу, которая выводит на экран количество времени, прошедшее с момента запуска операционной системы (формат: сутки, часы, минуты, секунды)

# **4.2. Информационное обеспечение обучения**

**Перечень рекомендуемых учебных изданий, Интернет-ресурсов, дополнительной литературы**

**Основная литература:**

1. Программирование на языке высокого уровня. Программирование на языке Object Pascal : учеб. пособие / Т.И. Немцова, С.Ю. Голова, И.В. Абрамова ; под ред. Л.Г. Гагариной. — М. : ИД «ФОРУМ» : ИНФРА-М, 2018. — 496 с. + Доп. материалы [Электронный ресурс; Режим доступа http://www.znanium.com]. — (Профессиональное образование). - Режим доступа: http://znanium.com/catalog/product/944326
2. Базовые средства программирования на Visual Basic в среде Visual Studio .NET : учеб. пособие / В.Н. Шакин. — М. : ФОРУМ : ИНФРА-М, 2019. — 303 с. — (Среднее профессиональное образование). - Режим доступа: http://znanium.com/catalog/product/961507

**Дополнительная**

1. Проектирование цифровых устройств : учебник / А.В. Кистрин, Б.В. Костров, М.Б. Никифоров, Д.И. Устюков. — М. : КУРС : ИНФРА-М, 2019. — 352 с. — (Среднее профессиональное образование). - Режим доступа: http://znanium.com/catalog/product/1002587
2. Объектно-ориентированное программирование на Visual Basic в среде Visual Studio .NET : учеб. пособие / В.Н. Шакин, А.В. Загвоздкина, Г.К. Сосновиков. — М. : ФОРУМ : ИНФРА-М, 2018. — 398 с. — (Среднее профессиональное образование). - Режим доступа: http://znanium.com/catalog/product/961516

**5.2.1. Коды проверяемых профессиональных и общих компетенций и личностных результатов:**

|  |  |
| --- | --- |
| ПК 1.1 | Выполнять разработку спецификаций отдельных компонент. |
| ПК 1.2 | Осуществлять разработку кода программного продукта на основе готовых спецификаций на уровне модуля. |
| ПК 1.3 | Выполнять отладку программных модулей с использованием специализированных программных средств. |
| ПК 1.4 | Выполнять тестирование программных модулей. |
| ПК 1.5 | Осуществлять оптимизацию программного кода модуля. |
| ПК 1.6 | Разрабатывать компоненты проектной и технической документации с использованием графических языков спецификаций. |
| ОК 1 | Понимать сущность и социальную значимость своей будущей профессии, проявлять к ней устойчивый интерес. |
| ОК 2 | Организовывать собственную деятельность, выбирать типовые методы и способы выполнения профессиональных задач, оценивать их эффективность и качество. |
| ОК 3 | Принимать решения в стандартных и нестандартных ситуациях и нести за них ответственность. |
| ОК 4 | Осуществлять поиск и использование информации, необходимой для эффективного выполнения профессиональных задач, профессионального и личностного развития. |
| ОК 5 | Использовать информационно-коммуникационные технологии в профессиональной деятельности. |
| ОК 6 | Работать в коллективе и в команде, эффективно общаться с коллегами, руководством, потребителями. |
| ОК 7 | Брать на себя ответственность за работу членов команды (подчиненных), за результат выполнения заданий. |
| ОК 8 | Самостоятельно определять задачи профессионального и личностного развития, заниматься самообразованием, осознанно планировать повышение квалификации. |
| ОК 9 | Ориентироваться в условиях частой смены технологий в профессиональной деятельности. |
|  |  |
| ЛР 2 | Проявляющий активную гражданскую позицию, демонстрирующий приверженность принципам честности, порядочности, открытости, экономически активный и участвующий в студенческом и территориальном самоуправлении, в том числе на условиях добровольчества, продуктивно взаимодействующий и участвующий в деятельности общественных организаций. |
| ЛР 4 | Проявляющий и демонстрирующий уважение к людям труда, осознающий ценность собственного труда. Стремящийся к формированию в сетевой среде личностно и профессионального конструктивного «цифрового следа». |
| ЛР 7 | Осознающий приоритетную ценность личности человека; уважающий собственную и чужую уникальность в различных ситуациях, во всех формах и видах деятельности. |
| ЛР 10 | Заботящийся о защите окружающей среды, собственной и чужой безопасности, в том числе цифровой. |
| ЛР 11 | Проявляющий уважение к эстетическим ценностям, обладающий основами эстетической культуры. |
| ЛР 13 | Демонстрирующий умение эффективно взаимодействовать в команде, вести диалог, в том числе с использованием средств коммуникации |
| ЛР 14 | Демонстрирующий навыки анализа и интерпретации информации из различных источников с учетом нормативно-правовых норм |