МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Пермский государственный национальный исследовательский университет»

Колледж профессионального образования

Методические рекомендации

для лабораторных и практических работ по изучению дисциплины

Системное программирование

для студентов Колледжа профессионального образования

специальности

09.02.03 Программирование в компьютерных системах

Утверждено на заседании ПЦК
Информационных технологий
Протокол № 9 от 23.05.2018
председатель _____Н.А. Серебрякова

Пермь 2018

Составитель:

Бочкарев Алексей Михайлович, преподаватель первой квалификационной категории, преподаватель ПГНИУ

СИСТЕМНОЕ ПРОГРАММИРОВАНИЕ: методические указания по практической работе для студентов Колледжа профессионального образования по специальностям 09.02.03 Программирование в компьютерных системах/ сост. А.М. Бочкарев; Колледж проф. образ. ПГНИУ. – Пермь, 2018. – 16 с.

Методические указания «СИСТЕМНОЕ ПРОГРАММИРОВАНИЕ» разработаны на основе требований Федерального государственного образовательного стандарта среднего профессионального образования по специальностям 09.02.03 Программирование в компьютерных системах и 09.02.04 Информационные системы (по отраслям) для оказания помощи студентам специальностей 09.02.03 Программирование в компьютерных системах по дисциплине «СИСТЕМНОЕ ПРОГРАММИРОВАНИЕ». Содержат типичные практические задания по всем разделам дисциплины.

Предназначены для студентов Колледжа профессионального образования ПГНИУ специальностей 09.02.03 Программирование в компьютерных системах (СПО) всех форм обучения.

Печатается по решению педагогического совета Колледжа профессионального образования Пермского государственного национального исследовательского университета

СОДЕРЖАНИЕ

Язык программирования Assembler

Задание.
Начиная с адреса 000Fh, в ОЗУ размещается массив из 10 чисел. Определить их сумму.

ПОРЯДОК ВЫПОЛНЕНИЯ ЗАДАНИЯ:

1) Для доступа к числам, находящимся в последовательных ячейках памяти, будем использовать косвенную адресацию через регистровые пары BC, DE или HL (т.к. адрес ячейки памяти 16-разрядный).

Блок-схема алгоритма решения данной задачи представлена на рисунке 1:

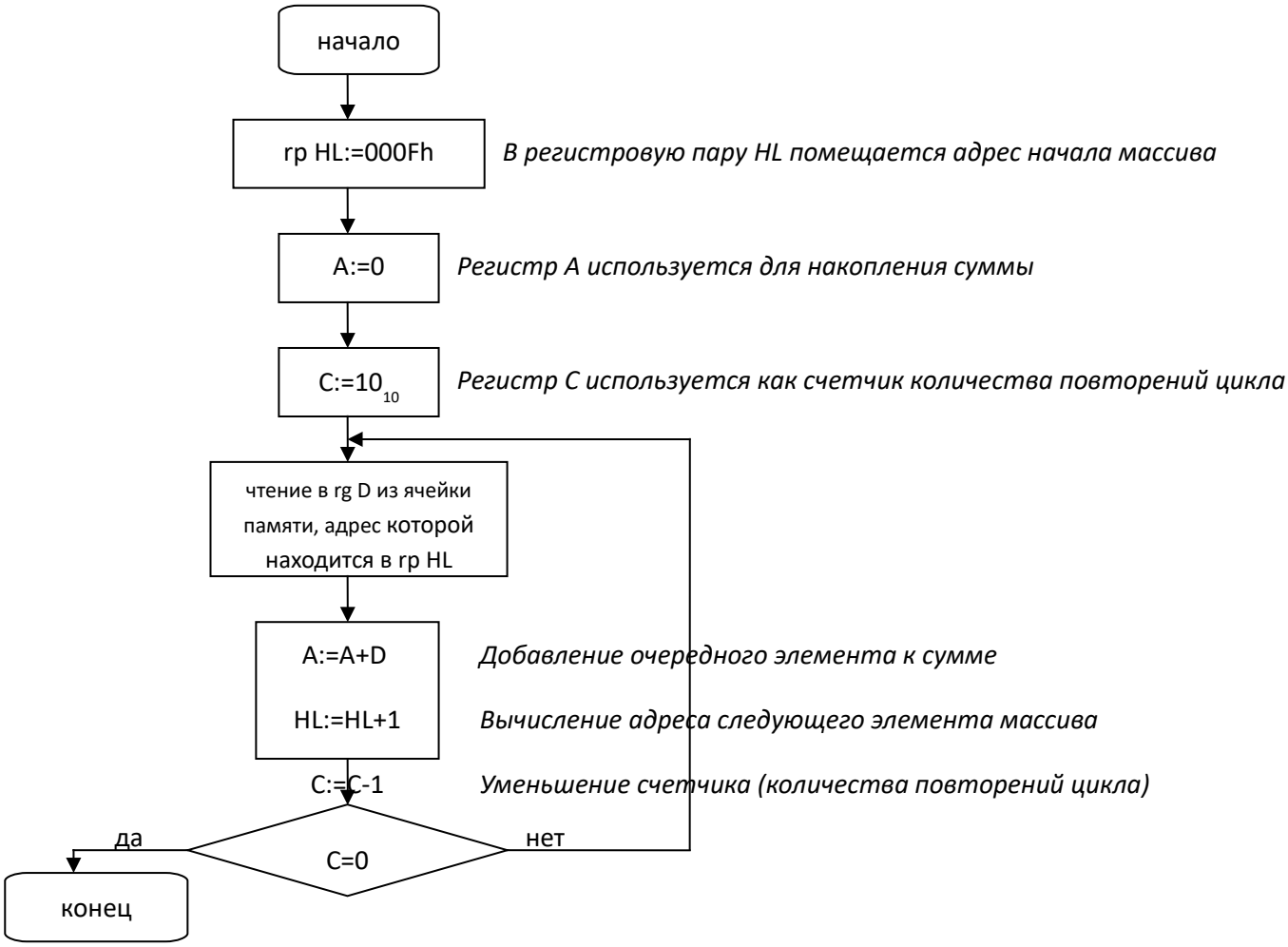


Рисунок 1 – Блок-схема алгоритма

2) Написать программу на языке ассемблера для процессора К580ВМ80 и занести машинные коды в ОЗУ.

В программе будем использовать команды:

LXI H, 000Fh – для помещения в регистровую пару адреса начала массива;

MOV D,M – переслать в регистр D содержимое ячейки памяти, адрес которой находится в HL;

INX H – для увеличения адреса, находящегося в регистровой паре на 1 (для получения адреса следующего элемента массива).

3) Начиная с адреса 000Fh в ОЗУ, разместить массив из 10 произвольных чисел, используя область «Ячейка ОЗУ и ее значение».

4) Запустить программу в режиме «Выполнить команду» и фиксировать значения программно-доступных компонентов в таблице 1 для первого и последнего прохождения цикла.

Таблица 1

Счетчик команд	Аккумулятор	Регистр H	Регистр L	Регистр D	Регистр C

5) Добавить последней командой программы команду HLT (76h). Запустить программу в режиме «Выполнить программу» и самостоятельно определить назначение этой команды.

6) Запишите в отчет значение аккумулятора (сумму чисел).

Работа с файлами в языке Assembler

Работа со всеми внешними устройствами эмулятора заключается в отправке или приёме на (с) соответствующий(его) устройству порт(а) МП-системы значения из (в) регистра-аккумулятора. Это осуществляется путём выполнения на эмуляторе команд ввода-вывода, таких как IN (принять из порта) и OUT (вывести в порт).

Всего к МП-системе подключено 5 устройств, соответственно портам ввода/вывода (00h… 04h):

- Порт 00h - «Монитор КР580» - Представляет собой виртуальный монитор, обеспечивающий вывод графической или текстовой информации. Графический режим соответствует разрешению 256x256 пикселей и глубине цвета - 128 бит на пиксель, а текстовый - 39x20 символов и глубине цвета 128 бит на символ. Одновременно монитор поддерживает два этих режима, т.е. может содержать и текст и графику;
- Порт 01h - «Накопитель на гибких магнитных дисках КР580» - Представляет собой виртуальный буфер дисковода, обеспечивающий вывод данных в файл на накопитель на гибких дисках реальной машины в реальном времени при наличии дискеты в дисководе A;
- Порт 02h - «Накопитель на жёстких магнитных дисках КР580» - Представляет собой виртуальный буфер жёсткого диска, обеспечивающий вывод данных в файл в реальном времени на накопитель на жёстких дисках реальной машины;
- Порт 03h - «Сетевой адаптер КР580» - Представляет собой виртуальный полудуплексный буфер данных, обеспечивающий передачу данных в реальном времени по сети реальных вычислительных машин по протоколу TCP/IP. Адрес и порт указывается в «настройках «сетевого адаптера»»;
- Порт 04h - «Принтер КР580» - Представляет собой виртуальный буфер данных, обеспечивающий вывод данных на принтер реальной машины по согласию пользователя.

РАБОТА С МОНИТОРОМ

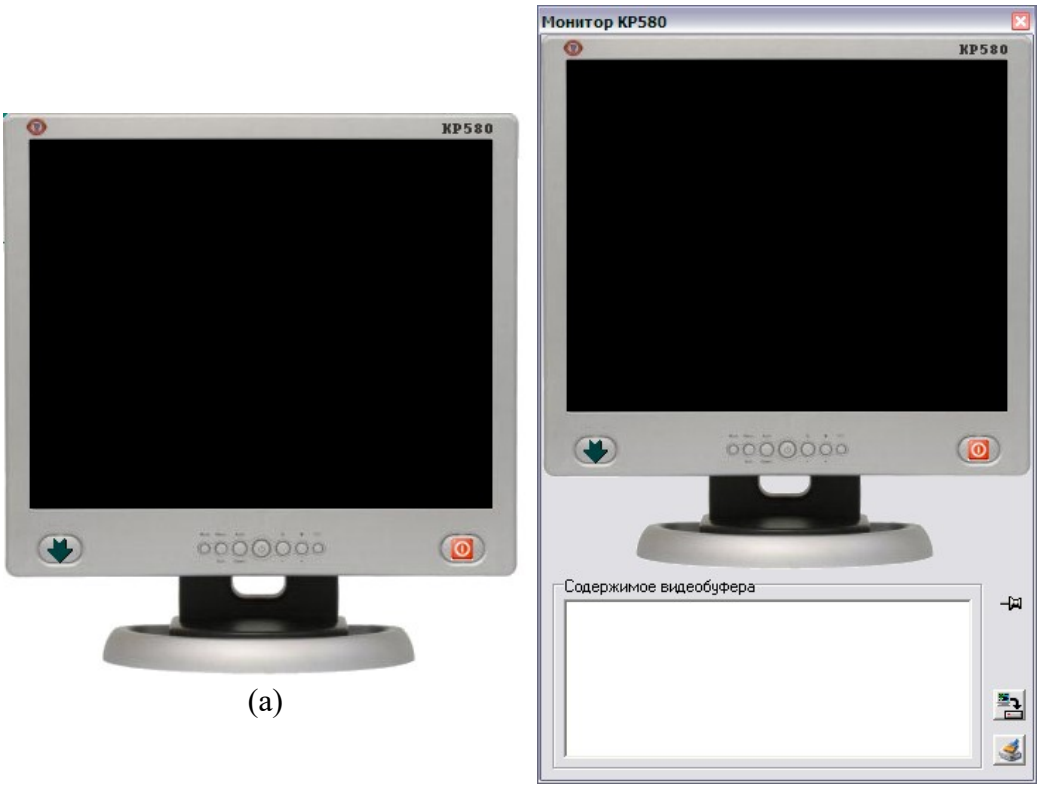
Команды засылаются в порт 00h побайтно. Различаются 3-х байтные и 2-х байтные команды:
2-х байтная.
1-ый байт: 1-ый бит - 0-текст, 1-графика; остальные 7 бит на цвет.
2-ой байт: номер символа в кодовой таблице (например, латинская буква «а» имеет код 61h).
3-х байтная.
1-ый байт: 1-ый бит - 0-текст, 1-графика; остальные 7 бит на цвет.
2 байт: координата по X (может принимать значения из диапазона от 00h до FFh).
3 байт: координата по Y (может принимать значения из диапазона от 00h до FFh).
За начало координат принимается левый верхний угол.

При нажатии левой кнопки мыши на пиктограмме монитора  mk:@MSITStore:C:\Program Files\KP580\Help\KP580_Help.chm::/HTML/Prog_MainWnd.htm, открывается окно виртуального монитора КР580, как показано на рисунке 2а. Оно содержит основной экран для вывода графической и текстовой информации, а также 2 кнопки:

1. Копка «Сокращённого/Подробного» варианта окна  - позволяет привести подробный вариант окна, как показано на рисунке 2б, или сокращённый вариант (рисунок 2а);
2. Кнопка «Закрыть»  - позволяет закрыть окно виртуального монитора КР580.

Подробный вариант окна показан на рисунке 2б, и содержит следующие элементы:

1. Основной экран - для вывода графической и текстовой информации;
2. Кнопка-переключатель режима «Всегда сверху»  - позволяет удерживать окно «монитор КР580» поверх всех остальных окон;
3. Содержимое видеобуфера - в этом элементе представлены данные в шестнадцатеричной форме в виде байтов, выведенные в виртуальный буфер монитора (порт 00h), и сгруппированы в строки по 16 значений в каждой. Начальный байт каждой строки также нумерован в виде шестнадцатеричного кода и располагается в левой части рассматриваемого элемента;
4. Кнопка «Сохранить изображение»  - позволяет сохранить изображение основного экрана окна «Монитор КР580» в файл в формате Windows Bitmap (bmp);
5. Кнопка «Очистить видеобуфер»  - позволяет очистить содержимое виртуального видеобуфера (00h), а также изображение на основном экране окна «Монитор КР580».



(а)

(б)

Рисунок 2 - Виды окна «Монитор КР580»:
а) сокращённый вариант; б) подробный вариант

Работа с памятью

Задание

Вывести на монитор буквы латинского алфавита по порядку.

Словесный алгоритм выглядит следующим образом:

1) B:=61h – код латинской буквы a

2) C:=1Ah – 26_{10} количество символов латинского алфавита

3) аккумулятору присвоить первый байт, содержащий в старшем разряде 0 – текстовый режим и код цвета; например, 01010000b=50h

4) с помощью команды OUT заслать байт из аккумулятора в порт вывода монитора

5) аккумулятору присвоить второй байт, содержащий код символа, т.е. содержимое регистра B

6) с помощью команды OUT заслать байт в порт вывода монитора

7) увеличить B на 1, т.е. получить код следующего символа латинского алфавита

8) уменьшить счетчик числа повторений цикла C на 1

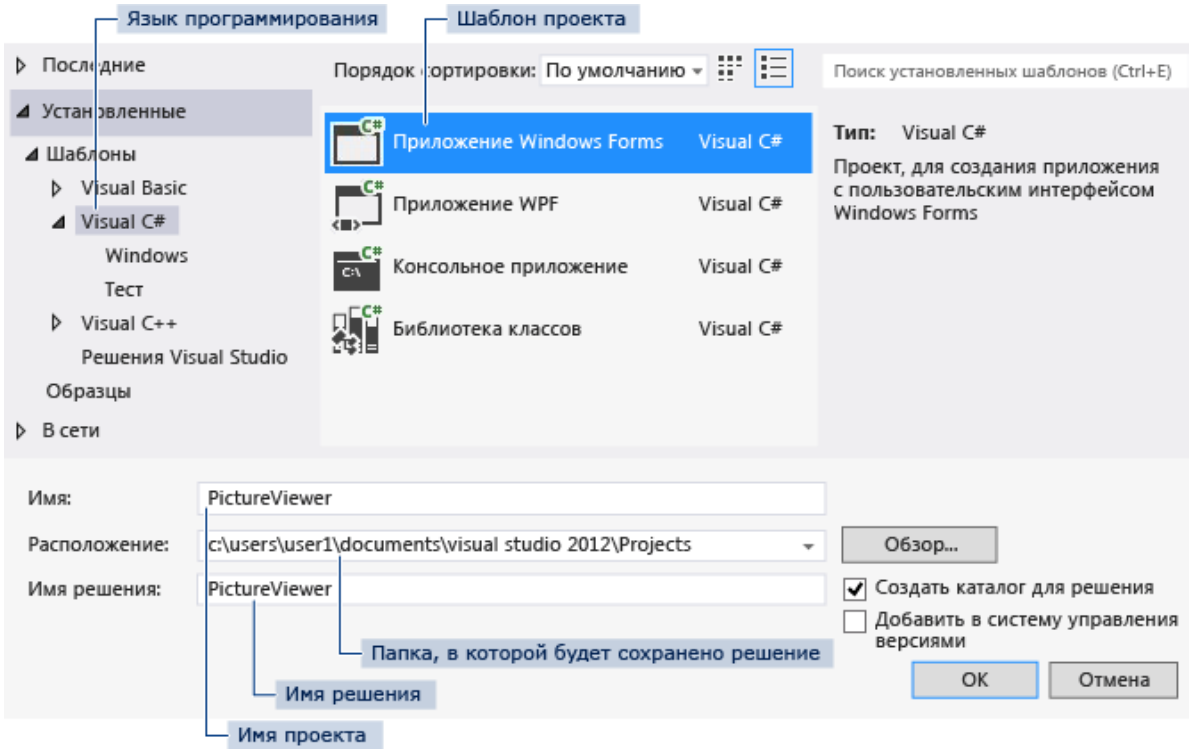
9) если C не равен нулю, вернуться к пункту 3

Сохраните экранную картинку в файл *.bmp. Откройте его с помощью программы MS Paint.

Характеристика языка Си

➢ КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ.

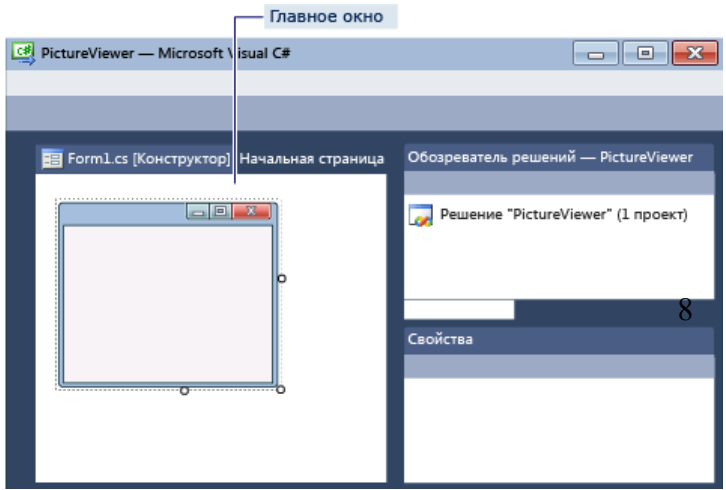
В строке меню выберите **Файл**, **Создать**, **Проект**. Диалоговое окно должно выглядеть следующим образом.



Диалоговое окно "Новый проект"

В списке **Установленные шаблоны** выберите **Visual C#** или **Visual Basic**.

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

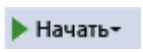
Visual Studio автоматически заполняет имя папки и имя проекта, а затем сохраняет проект в папке проектов.

Для запуска программы используйте один из следующих методов.

Нажмите клавишу **F5**.

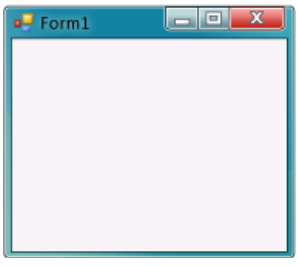
В строке меню выберите **Отладка**, **Начать отладку**.

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



Кнопка панели инструментов "Начать отладку"

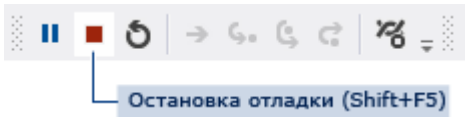
Visual Studio запускает программу, и открывается окно **Form1**. На следующей диаграмме показана только что созданная программа. Программа выполняется и скоро она будет дополнена.



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 дей-

ствия, 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**.

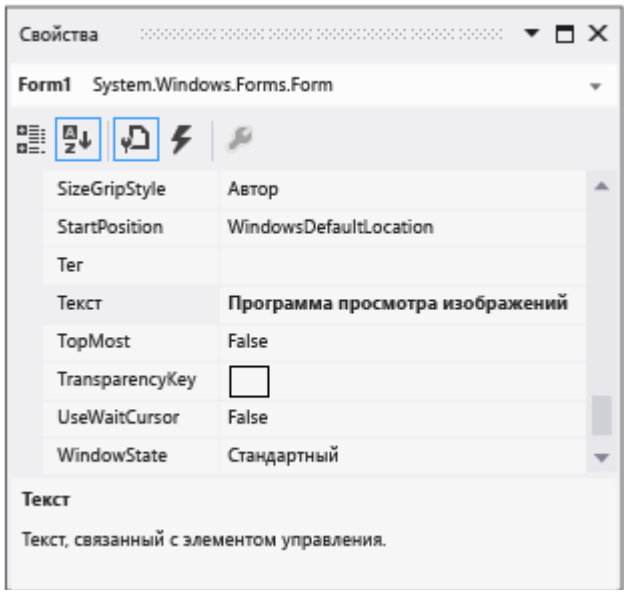
В строке меню выберите **Отладка**, **Остановить отладку**.

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).

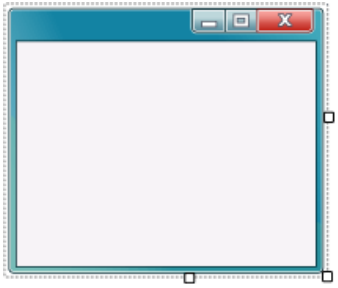
Чтобы выделить форму **Form1**, щелкните в любом ее месте. Посмотрите на окно **Свойства**. Теперь в нем должны отображаться свойства формы. У формы есть различные свойства. Например, можно установить цвет переднего плана и фона, текст заголовка, который отображается в верхней части формы, размер формы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

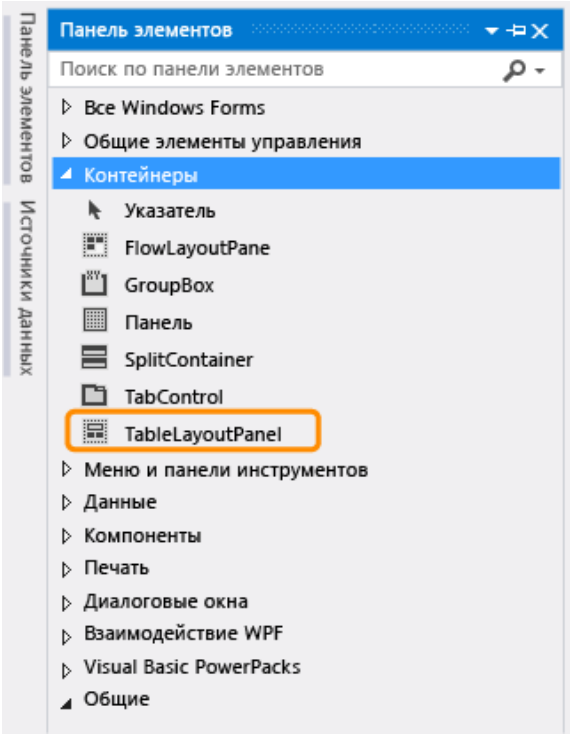
На панели инструментов нажмите кнопку **Остановить отладку**.

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**.)

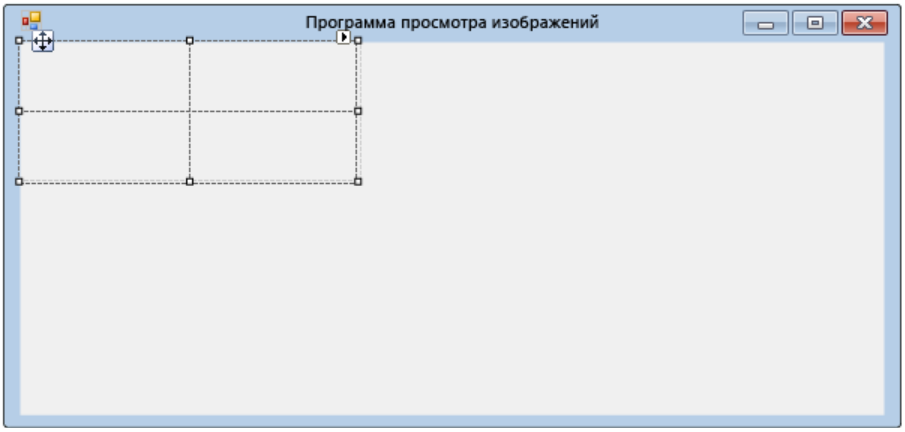
Выберите маленький треугольник рядом с группой **Контейнеры**, чтобы открыть ее, как показано на рисунке ниже.



Группа "Контейнеры"

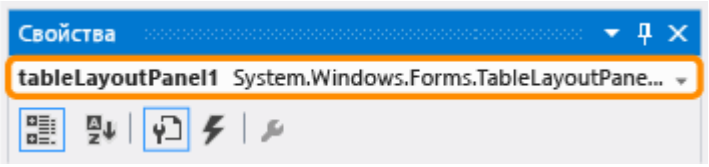
В форму можно добавить такие элементы управления, как кнопки, флажки и метки. Дважды щелкните элемент управления TableLayoutPanel в панели элементов. (Можно также перетащить элемент управления с панели элементов в форму.) В результате этого действия интегрированная среда разработки добавляет в форму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



Значок канцелярской кнопки

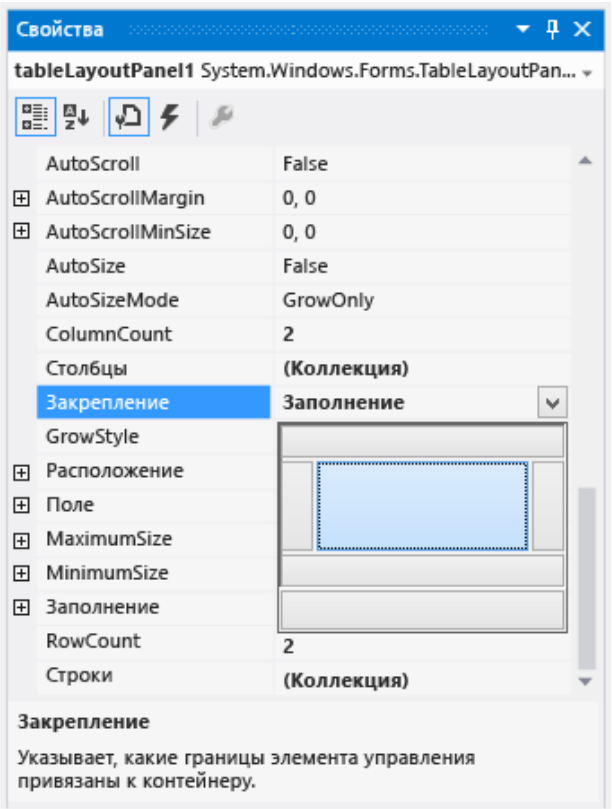
Убедитесь, что выделен элемент управления **TableLayoutPanel**, для этого щелкните по нему. Чтобы проверить, что элемент управления выделен, необходимо посмотреть в раскрывающийся список в верхней части окна **Свойства**, как показано на рисунке ниже.



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 пример 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 **Обозреватель 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.

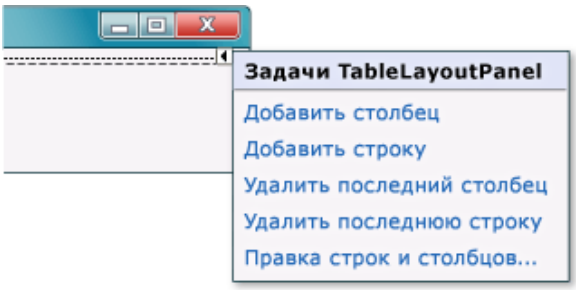
В данный момент элемент управления TableLayoutPanel содержит две одинаковые по размеру строки и два одинаковых по размеру столбца. Нужно изменить их размер, чтобы верхняя строка и правый столбец были намного

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

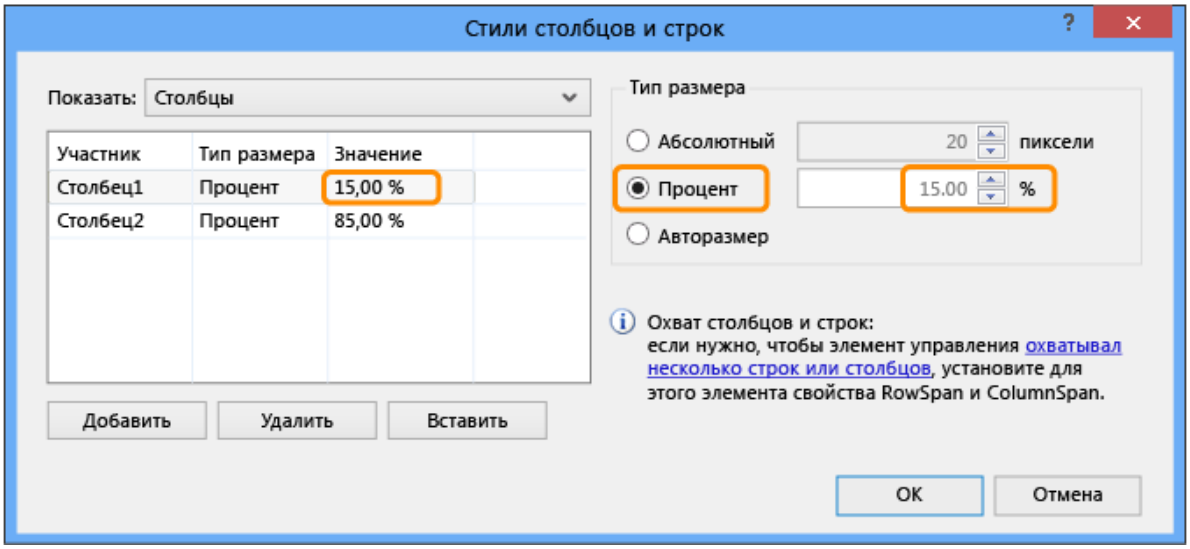
Это кнопка указывает, что элемент управления содержит задачи, которые помогут автоматически задать его свойства.

Чтобы отобразить список задач элемента управления, как показано на рисунке ниже, нажмите треугольник.



Задачи элемента управления TableLayoutPanel

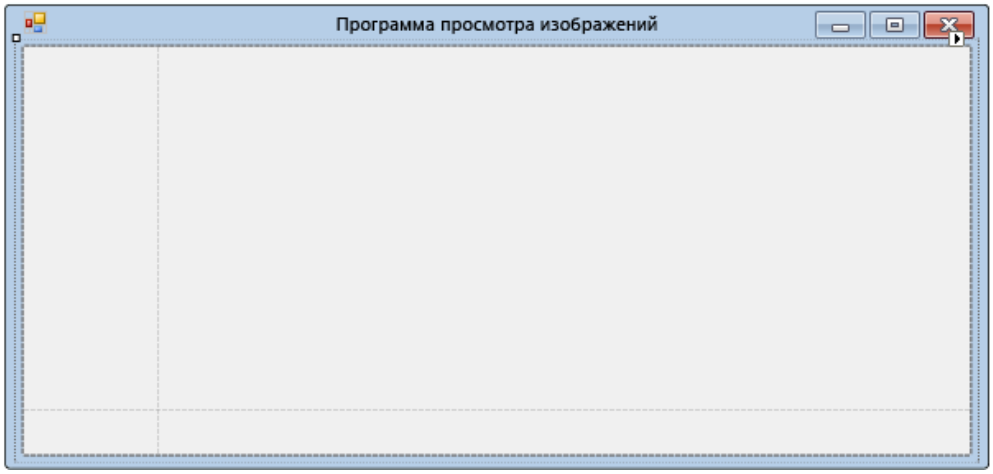
Выберите задачу **Изменить строки и столбцы**, чтобы открыть окно **Стили столбцов и строк**. Выберите **Столбец1**, убедитесь, что нажата кнопка **Проценты**, установите его размер равным 15 процентам — введите 15 в поле **Проценты**. (это элемент управления NumericUpDown, который далее будет использоваться в этом руководстве). Выберите **Столбец2** и задайте значение 85 процентов. Пока не нажимайте кнопку **ОК**, так как окно будет закрыто. (но если это было сделано, его можно открыть повторно с помощью списка задач).



Стили столбцов и строк TableLayoutPanel

В раскрывающемся списке **Показать** в верхней части окна выберите **Строки**. Задайте для **Row1** значение 90 процентов, а для **Row2** 10 процентов.

Нажмите кнопку **ОК**. Элемент управления TableLayoutPanel теперь должен содержать большую верхнюю строку, маленькую нижнюю строку, маленький левый столбец и большой правый столбец. Можно изменить размер строк и столбцов в элементе управления TableLayoutPanel, выбрав tableLayoutPanel1 в форме, а затем перетащив границы его строк и столбцов.



Form1 с измененным размером TableLayoutPanel

- **ПОСЛЕДОВАТЕЛЬНОСТЬ ВЫПОЛНЕНИЯ РАБОТЫ**

1. Изучите методические указания и конспект лекций.
2. Проанализируйте задание по своему варианту.
3. Создайте приложение, отвечающее запросу задания.

- **МЕТОДИКА АНАЛИЗА РЕЗУЛЬТАТОВ, ОБРАЗЕЦ ОТЧЕТА.**

1. Отчет должен содержать цель работы.
2. Содержание индивидуального задания.
   Перечень элементов, выбранных для создания приложения.

Решение задач вычисления математических выражений различной сложности

Задание

Научиться использовать делегаты, перечисления и интерфейсы при программировании на языке C#

- **ОБЕСПЕЧЕННОСТЬ ЗАНЯТИЯ**

Оборудование лаборатории и рабочих мест лаборатории:

компьютеры, принтер, сканер, проектор, программное обеспечение общего и профессионального назначения, комплект учебно-методической документации.

Реализация профессионального модуля предполагает обязательную учебную практику.

Оборудование и технологическое оснащение рабочих мест:

1.Комплект ТС компьютера IBM-PC

2.Методические указания для выполнения практических работ

4. Microsoft Visio.

5. Microsoft Visual Studio.

6.Microsoft Office.

➢ КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ.

Современные объектно-ориентированные языки программирования в качестве кусочка предлагают нам объекты — собственно отсюда и название. Объектом как правило является экземпляр класса, который делает что-то нужно и полезное. Самое простое средство декомпозиции в C++ — это сделать классы глобальными и непосредственно вызывать их методы. Подход не лишен недостатков — понять кто кого вызывает по прошествии времени становится все труднее, и через пару лет такой подход приведет к тем же последствиям, что и использование единственного класса. Одобренный святой инквизицией вариант — это передача классам указателей на те классы, с которыми им надобно общаться. Причем желательно, чтобы указатели были не просты, а на интерфейсы — тогда менять и развивать программу по прошествии времени станет намного проще.
Если объекту нужно всего несколько взаимодействий, например кнопке уведомить о том, что на нее кликнули — то реализация для этих целей отдельного интерфейса займет ощутимое количество строк кода. Также интерфейсы не решают задачу когда одному объекту нужно уведомить о чем-то несколько других — создание и поддержание листа подписки на основании интерфейсов тоже не самое малое число строк кода.
В динамических языках программирования конкуренцию интерфейсам составляют делегаты. Как правило, делегат очень похож на «указатель на функцию» в C++, с той основной разницей что делегат может указывать на метод произвольного объекта.

Это должно быть нечто, что можно ассоциировать с методом произвольного класса а затем вызвать как функцию. В C++ это можно сделать только в виде класса с перегруженным оператором вызова (такие классы обычно называют функторами). В C# для ассоциации делегата и метода используется оператор "+=", но в C++ это к сожалению невозможно — оператор "+=" принимает только один параметр, в то время как указатель на функцию — член класс в C++ определяется двумя параметрами.

Попробуем реализовать это поведение. Чтобы делегат мог получить указатель на **любой** метод **любого** класса его собственный метод Connect() явно должен быть шаблонным. Почему не сделать шаблонным сам делегат? Потому что тогда придется указывать конкретный класс при создании делегата, а это противоречит возможности ассоциировать делегат с любым классом. Также у делегата должен быть перегруженный оператор вызова, тоже шаблонный — чтобы можно было вызвать с теми же типами аргументов, что и у ассоциированного с ним метода.
Метод Connect() можно вызывать с указателем на метод любого класса, а оператор вызова позволяет вызывать сам делегат с любым аргументом. Теперь все что нужно сделать — это каким-то образом сохранить указатель на класс i_class и на метод i_method, чтобы их можно было использовать в операторе вызова. Тут случается затык номер раз — сам делегат о T и M ничего не знает и знать не должен, сохранить их как поля не получится (это аргументы шаблонного метода, который для одного и того же делегата можно вызвать много раз с разными классами и методами). Что делать? Придется обратиться к небольшой шаблонной магии (поверьте, это заклинания первого уровня по сравнению с теми, что применяются в boost:function).

Единственный способ в C++ сохранить аргументы шаблона — это создать экземпляр шаблонного класса, который будет параметризирован этими аргументами, и, соответственно, будет их помнить. Следовательно, нам нужен шаблонный класс, который сможет запомнить T и M. А чтобы сохранить указатель на этот класс, он должен наследоваться от интерфейса, не имеющего шаблонных параметров:

В первом приближении этого хватит, чтобы можно было вызвать Connect() для любого метода любого класса и запомнить аргументы. Но запомнить — это половина дела. Вторая половина — это вызвать запомненный метод с переданным нам аргументом. С вызовом есть некая сложность — указатель на метод класса мы сохранили как интерфейс IContainer — как теперь вызвать метод класса параметром произвольного типа, который пользователь передал в operator()?

Самый простой способ — это запомнить переданный аргумент в контейнере так же как мы делали это для указателя на метод класса, передать контейнер с аргументом «внутрь» m_container, а затем воспользоваться dynamic_cast<>() чтобы «вынуть» аргумент из контейнера.
Последняя проблема на пути к работающему делегату — это извлечение аргумента из контейнера. Для того, чтобы это сделать нужно знать тип аргумента. Но ведь внутри контейнера, хранящего указатель на метод класса, мы не знаем тип аргумента? Тип аргумента не знаем — зато знаем сигнатуру метода, указатель на который мы храним. Следовательно все что нам нужно — это извлечь тип аргумента из сигнатуры.
Собственно, все. Получившийся делегат сохраняет указатель на любой метод любого класса и позволяет вызвать его с синтаксисом вызова функции.

ПОСЛЕДОВАТЕЛЬНОСТЬ ВЫПОЛНЕНИЯ РАБОТЫ

1. Изучите методические указания и конспект лекций.
2. Проанализируйте задание по своему варианту.
3. Создайте приложение, отвечающее запросу задания.

- **МЕТОДИКА АНАЛИЗА РЕЗУЛЬТАТОВ, ОБРАЗЕЦ ОТЧЕТА.**

1. Отчет должен содержать цель работы.
2. Содержание индивидуального задания.
3. Перечень элементов, выбранных для создания приложения.

Работа с файлами и строками

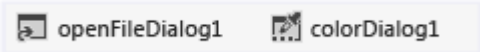
Задание.

1. Выберите конструктор Windows Forms (Form1.cs [Design] или Form1.vb [Design]) и откройте группу **Диалоговые окна** в панели элементов.

   Примечание

   Группа **Диалоговые окна** в панели элементов содержит компоненты, которые открывают множество полезных диалоговых окон. Эти диалоговые окна могут использоваться для открытия и сохранения файлов, просмотра папок, выбора шрифтов и цветов. В этом проекте используется два компонента диалоговых окон — **OpenFileDialog** и **ColorDialog**.
2. Чтобы добавить в форму компонент с именем **openFileDialog1** дважды щелкните компонент **OpenFileDialog**. Для добавления в форму компонента с именем **colorDialog1** дважды щелкните в

панели элементов компонент **ColorDialog**. (такой компонент используется в следующем шаге руководства). В нижней части конструктора Windows Forms должно быть поле (под формой программы просмотра изображений), которое содержит значок для каждого из двух добавленных компонентов диалоговых окон, как показано на рисунке ниже.



Компоненты диалоговых окон

3. Выберите значок **openFileDialog1** в области в нижней части конструктора Windows Forms. Установите значения двух свойств.
    - Задайте свойству **Фильтр** следующее значение (можно копировать и вставить):
    - JPEG Files (*.jpg)|*.jpg|PNG Files (*.png)|*.png|BMP Files (*.bmp)|*.bmp|All files (*.*)|*.*
    -
    - Установите для свойства **Title** следующее значение "Выбор файла изображения".

      Параметры свойства **Фильтр** определяют типы файлов, которые отображаются в диалоговом окне **Выбор файла изображения**.

➢ ПОСЛЕДОВАТЕЛЬНОСТЬ ВЫПОЛНЕНИЯ РАБОТЫ

1. Изучите методические указания и конспект лекций.
2. Проанализируйте задание по своему варианту.
3. Создайте приложение, отвечающее запросу задания.

➢ МЕТОДИКА АНАЛИЗА РЕЗУЛЬТАТОВ, ОБРАЗЕЦ ОТЧЕТА.

1. Отчет должен содержать цель работы.
2. Содержание индивидуального задания.
3. Перечень элементов, выбранных для создания приложения.

Методическое издание

«СИСТЕМНОЕ ПРОГРАММИРОВАНИЕ»:

методические указания по практической работе

для студентов Колледжа профессионального образования специальностям 09.02.03 Программирование в компьютерных системах

Составитель:

Бочкарев Алексей Михайлович

Редактор _____

Корректор _____

Подписано в печать _____

Формат 60x84/16. Усл.печ.л. _____. Уч.-изд.л. _____.

Тираж 100 экз. Заказ

Редакционно-издательский отдел

Пермского государственного университета

614990. Пермь, ул. Букирева, 15

Типография Пермского государственного университета

614990. Пермь, ул. Букирева, 15