Содержание

Введение

Глава 1. Исследовательская часть

1.1 Целесообразность выбора языка программирования

1.2. Основные структуры языка программирования C#

1.3. Кодирование по методу четности / нечетности

1.4. Кодирование по методу Хэмминга

1.5. Машина Поста

Глава 2. Инструкция программиста

Глава 3. Инструкция пользователя

3.1 Использование программы StudyProgram для усвоения учебного материала по кодированию информации методом четности и методом Хэмминга

3.2 Использование обучающей программы StudyProgram для овладения навыками составления программ для машины Поста

Заключение

Список литературы

Приложение

# Введение

В курсовой работе была поставлена задача создания обучающей программы по информатике, с помощью которой студенты смогут проверить свои знания в таких разделах данной дисциплины как кодирование по методу Хэмминга, кодирование по методу четности-нечетности, программирование машины Поста.

Цель работы: улучшить и расширить навыки работы на языке программирования C#.

Основные функции обучающей программы: изучение данных методов кодирования двоичных чисел и программирование машины Поста. Курсовой проект состоит из обучающей программы с разбиением по разделам.

Для создания курсового проекта использовался C# – новый язык программирования компании Microsoft, входящий в Visual Studio .Net (v.7).

# Глава 1. Исследовательская часть

##

## 1.1 Целесообразность выбора языка программирования

В курсовом проекте была поставлена задача, реализация которой может быть произведена с помощью различных языков программирования. Но мы выбрали язык С#, входящий в Visual Studio .Net (v.7), так как он имеет ряд преимуществ, которые упрощают процесс создания приложений. Немаловажную роль в выборе языка сыграло то, что C# - лучший .NET-язык, так как он был специально спроектирован для .NET Framework.

C# создавался Microsoft, как основной язык для .NET Framework. Microsoft спроектировала C# таким образом, что бы С, С++ и Java-программисты смогли легко перейти на него. C# имеет корни в языках С, С++ и Java, поэтому такой переход не должен вызвать затруднений.

Синтаксис у C# не такой сложный как у С++, поэтому его изучение намного легче. Большинство операций, которые вы можете делать на С++, можно сделать и на C#, за исключением операций доступа к низкоуровневым функциям (это все-таки можно сделать с помощью неуправляемого кода).

С# - первый язык, поддерживаемый версиями .NET Framework для других платформ.

Язык, который использует библиотеку классов .NET Framework (FCL) и управляется общеязыковой средой исполнения (CLR) является .NET-совместимым языком (язык, который поддерживает платформа .NET). Среди таких языков: Microsoft Visual Basic .NET (VB .NET), Microsoft Visual C++ .NET , а также COBOL, Eiffel, Jscript, RPG и другие. Как мы говорили ранее, C# был спроектирован специально для .NET Framework и содержит некоторую функциональность, которую вы не найдете в других языках.

Программист, привыкший к платформе Майкрософт, может представить себе C# как промежуточный вариант между C++ и Visual Basic, если рассматривать сложность языковых конструкций и возможности языка.

C# имеет C стиль синтаксиса (для управляющих конструкций, блоков кода, описания сигнатуры методов и др.), много общего с Java (отсутствие множественного наследования и шаблонов, наличие сборщика мусора) и Дельфи (ориентированность на создание компонент), в то же время имеет и свой колорит.

При создании языка в основу дизайна легла легкость использования, доминирующая над мощностью языка и скоростью выполнения. [7] Отсюда и сборщик мусора с управляемыми объектными ссылками, который автоматически освобождает за Вас память, отбирая при этом процессорное время. Вы также получаете безопасность работы с типами, а это, по мнению многих, является вторым важнейшим фактором избежания ошибок.
C# объектно-ориентированный язык, как и вся платформа .NET[7].

При создании языка рассматривалась не только простота написания приложений, но и их поддержки - в связи, с чем в язык включили поддержку XML комментариев и контроля версий. Настоящий подарок для программистов.

В C# представлена концепция пространств имен, аналогичная пакетам в Java. Это позволяет иерархически структурировать Вашу систему типов, делая код намного более понятным и позволяя избежать проблем с именованием. Вы можете рассматривать пространства имен как директории, а языковые типы как файлы в этих директориях.

Реализация структур как типов, работа с которыми идет по значению, вместе с возможностью использовать не только вложенные массивы (как в Java), но и многомерные позволяет оптимизировать производительность приложений.

Ввиду очень удобного объектно-ориентированного дизайна, C# является хорошим выбором для быстрого конструирования различных компонентов - от высокоуровневой бизнес логики до системных приложений, использующих низкоуровневый код[6]. Также следует отметить, что C# является и Web ориентированным - используя простые встроенные конструкции языка ваши компоненты могут быть легко превращены в Web сервисы, к которым можно будет обращаться из Internet посредством любого языка на любой операционной системе. Дополнительные возможности и преимущества перед другими языками приносит в C# использование передовых Web технологий, таких как: XML (Extensible Markup Language) и SOAP (Simple Object Access Protocol). Среда разработки Web сервисов позволяет программисту смотреть на существующие сегодня Web приложения, как на родные C# объекты, что дает возможность разработчикам соотнести имеющиеся Web сервисы с их познаниями в объектно-ориентированном программировании[6].

Как было показано выше, выбранный язык программирования наиболее удобен для выполнения поставленной задачи.

## 1.2. Основные структуры языка программирования C#

**Классы**

Классы — сердце каждого объектно-ориентированного языка. Как вы помните, класс представляет собой инкапсуляцию данных и методов их обработки. Это справедливо для любого объектно-ориентированного языка и отличаются они в этом плане лишь типами тех данных, которые можно хранить в виде членов, а также возможностями классов. В том, что касается классов и многих функций языка, С# кое-что заимствует из C++ и Java, и привносит немного изобретательности, помогающей найти элегантные решения старых проблем.

Синтаксис определения классов на С#, прост, особенно если вы программируете на C++ или Java. Поместив перед именем вашего класса ключевое слово class, вы вставляете члены класса, заключенные в фигурные скобки, например:

class Employee

{

private long employeeld;

}

Как видите, этот простейший класс с именем Employee содержит единственный член — employeeld. Заметьте: имени члена предшествует ключевое слово private — это модификатор доступа (access modifier).

**Члены класса**

Типы, определенные в CIS (Common Type System) поддерживаются как члены классов С# и бывают следующих видов.

* Поле. Так называется член-переменная, содержащий некоторое значение. В ООП поля иногда называют данными объекта. К полю можно применять несколько модификаторов в зависимости от того, как вы собираетесь это поле использовать. В число модификаторов входят static, readonly и const.
* Метод. Это реальный код, воздействующий на данные объекта (или поля).
* Свойства. Их иногда называют "разумными" полями (smart fields), так как они на самом деле являются методами, которые клиенты класса воспринимают как поля. Это обеспечивает клиентам большую степень абстрагирования за счет того, что им не нужно знать, обращаются ли они к полю напрямую или через вызов метода-аксессора.
* Константы. Как можно предположить, исходя из имени, константа — это поле, значение которого изменить нельзя.
* Индексаторы. Если свойства — это "разумные" поля, то индексаторы - это "разумные" массивы, так как они позволяют индексировать объекты методами-аксессорами get и set. С помощью индексатора легко проиндексировать объект для установки или получения значений.
* События. Событие вызывает исполнение некоторого фрагмента кода. События — неотъемлемая часть программирования для Microsoft Windows. Например, события возникают при движении мыши, щелчке или изменении размеров окна. События С# используют ту же стандартную модель публикации/подписки (publish/subscribe), что и в MSMQ (Microsoft Message Queuing) и асинхронной модели событий СОМ+, которая дает приложению средства асинхронной обработки событий. Но в С# это базовая концепция, встроенная в язык.
* Операторы. Используя перегрузку операторов С#, можно добавлять к классу стандартные математические операторы, которые позволяют писать более интуитивно понятный код.

**Массивы**

Массивы в C# очень просты и похожи на массивы в C++. В C# имеется три основных типа массивов: одномерные, многомерные и неровные(jagged):
С одномерными массивами все как обычно:

Инициализация.

Инициализируются они также как в C++:

int[] array = new int[10];

int[] array = {1, 2, 3, 4, 5};

Доступ

Доступ к элементам производится с помощью оператора []:

int element = array[0];

Элементы нумеруются индексами от 0 до N - 1, где N - размер массива.
Многомерные массивы представляют собой многомерные кубы значений. Элементы таких массивов идентифицируются набором индексов - "координат" в многомерном пространстве. Каждое измерение имеет свою размерность, не зависящую от других. Отметим, что многомерные массивы являются важным отличием от других подобных языков (Java), ибо по сравнению с неровными массивами, обеспечивают гораздо большую производительность.

**Инициализация**

При инициализации размерности измерений указываются через запятую:

int[,] array = new int[10, 20];

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

Доступ

Доступ к элементам производится с помощью оператора [], в котором индексы также указываются через запятую:

int element = array[0, 2];

Элементы нумеруются наборами индексов, каждый из которых может принимать значение от 0 до N - 1, где N - размер соответствующего измерения массива.

**Управляющие конструкции**

Управляющие конструкции - это то средство, которое позволяет вам создавать алгоритмы. Стандартными управляющими конструкциями являются if, for, while goto. К тому же в C# есть много дополнительных конструкций: switch, do и foreach.

if..else

Блок if..else, как следует из названия, позволяет выполнять код при выполнении определенного условия.

if (a > b)

{

 Console.WriteLine("a > b");

}

else

{

 Console.WriteLine("a <= b");

}

Логическое выражение в скобках должно иметь тип bool (в отличие от C/C++, где выражение было целым). Если оно истинно, выполнится первый блок команд, если ложно - второй. При этом если какой-либо блок команд состоит лишь из одной команды, обрамляющие фигурные скобки можно опустить[8]:

if (serverObject != null)

 serverObject.Initialize();

Покажем несколько примеров:

// Идет проверка, чтобы избежать деления на 0

if (x != 0)

{

 d /= x;

}

else

 return;

// Проверяем, вернула ли процедура значение и, если нет, возвращаем null

if (command.Parameters["RETURN\_VALUE"].Value != Convert.DBNull)

 return command.Parameters["RETURN\_VALUE"].Value;

else

 return null;

Цикл for

Цикл for представляет цикл с инициализирующей командой, условием окончания и командой перехода.

ArrayList items = new ArrayList(); // Инициализируем список некоторым образом

CreateArray(items); // Выводим все его элементы по очереди

for (int i = 0; i < items.Count; i++)

{

Console.WriteLine("i[{0}] = {1}", i, items[i]);

}

Логика выполнения цикла следующая: сначала выполняется инициализирующая команда, далее идет шаг цикла: проверяется условие окончания, если оно ложно, то цикл прекращается и управление переходит на следующую за циклом команду, иначе выполняется тело цикла, а после него - команда перехода.

В простейшем случае цикл for оформлен следующим образом:

// Складываем все элементы массива

for (int i = 0; i < 10; i++)

sum += array[i];

Такая форма цикла используется, если вам нужно проитерироваться с целым параметром, указывающим номер итерации или если просто нужно выполнить действие некоторое количество (в нашем примере 10) раз.

while

Цикл while является циклом с предусловием: сначала проверяется определенное условие перехода, а затем выполняется итерация цикла. Синтаксис цикла while в общем случае следующий:

// Задаем число

int num = 678;

// Число разрядов - 0

int len = 0;

// считаем сколько разрядов в числе

while (num > 0)

{

 len++;

 num /= 10;

}

Console.WriteLine(len);

Семантика очень проста: проверяем условие перехода. Если оно истинно, то переходим к телу цикла и затем снова к проверке условия, иначе - прекращаем выполнение цикла.

switch

Оператор switch является расширенным оператором ветвления, который позволяет в зависимости от значения выражения перейти к выполнению определенного кода. По сути, он эквивалентен набору блоков if, но гораздо более оптимален.

Наиболее распространенным синтаксисом switch является следующий:

int a = 1;

switch (a)

{

 case 0:

 //Операторы, выполняющиеся если a = 0

 Console.WriteLine("A = 0");

 break;

 case 1:

 //Операторы, выполняющиеся если a = 1

 Console.WriteLine("A = 1");

 break;

 default:

 //Операторы, выполняющиеся, если ни один из явных случаев не подошел

 Console.WriteLine("A <> 0 и A <> 1");

 break;

}

Блок case x выполняется, если выражение, указанное в скобках после switch равно x. Блок default - если ни одно из case-выражений не выполнилось. Удобно switch использовать для обработки выражений перечислимых типов.

Для переходов между блоками можно использовать команды goto case и goto default.

do

Оператор do реализует цикл с постусловием. Суть этого цикла аналогична while, но условие выхода проверяется не до итерации, а после. Это иногда бывает удобным.

string s = "A, B, C, D";

do

{

 s = s.Substring(s.IndexOf(",") + 1);

} while (s.Length > 0);

Обычно его используют, когда проверку нужно выполнить после того, как прошла хотя бы одна итерация.

Поля и методы

Объявление полей аналогично объявлению переменных и имеет вид:

<модификатор\_доступности> <тип> <идент\_поля>;

Существует несколько различных модификаторов доступа. Мы будем использовать только один – public. В качестве типа может использоваться любой из стандартных или пользовательских. Идентификатор принято начинать с буквы f с указанием, какое свойство реального объекта отражает это поле, например, fSpeed.

Если имя поля состоит из нескольких слов, то используется верблюжья нотация fMaxFloor. Например, объявление полей классов может иметь вид:

public int fLevel;

public int fPassenger;

public float fSpeed;

При объявлении методов используется конструкция вида:

<модиф\_дост><тип\_возвр\_знач><идент\_метода>(<формальн\_парам>opt)

{

 <тело\_метода>

}

Метод в процессе своей работы может не только изменять поля объекта, но и вычислять некоторые дополнительные значения. Тип возвращаемого методом значения и указывается вместо <тип\_возвр\_знач>.

Если возвращать значения не нужно, то в качестве типа возвращаемого значения указывается void.

Параметры методов бывают формальные и реальные. Формальные параметры используются при объявлении метода в круглых скобках, идущих за идентификатором метода. Реальные параметры передаются в метод при обращении к нему в круглых скобках. В случае если у метода есть параметры, его объявление будет иметь вид:

<модиф\_дост><тип\_возвр\_знач> <идент\_перем> (<список\_форм\_пар>)

{

 <тело\_цикла>

}

<список\_формальных\_параметров>::=<тип><идент>,<тип><идент>opt ..

Например, public void Move (byte level)

В качестве реального параметра в метод может передаваться константа, переменная, результат выполнения операции, значение возвращаемой функции и т.д. Единственное ограничение – тип реального параметра должен совпадать с типом формального параметра[9].

В случае использования простых типов для объявления формальных параметров значения передаются в метод, и даже если их в методе изменять, реальный параметр не изменится.

В случае, когда методу требуется вернуть более 1 значения, для 2, 3 и т.д. возвращаемого значения перед типом формального параметра, используемого для возвращения значения, ставится ключевое слово ref. В случае вызова такого метода перед реальным параметром также указывается ключевое слово ref, и реальный параметр в этом случае может быть только переменной.

## 1.3. Кодирование по методу четности / нечетности

Для контроля правильности передачи информации, а также как средство шифрования информации используются различные коды. Коды, использующие для передачи информации два различных элементарных сигнала, называются двоичными. Эти сигналы удобно обозначать символами 0 и 1. Тогда кодовое слово будет состоять из последовательности нулей и единиц.

При выполнении арифметических операций в цифровом автомате правильный результат будет получен только в случае, если машина работает без нарушений. При возникновении какой-либо ошибки в двоичном коде результат операции будет неверным, однако пользователь об этом не узнает, если не будут предусмотрены меры, сигнализирующие о появлении ошибки. То есть должна быть разработана определенная система контроля работы цифрового автомата. Для этой цели и служат различные методы кодирования двоичной информации.

Систематический код – это код, содержащий в себе информационные и контрольные разряды. В контрольные разряды записывается некоторая информация об исходном числе, поэтому систематический код обладает избыточностью.

При этом абсолютная избыточность будет выражаться количеством контрольных разрядов – k, а относительная избыточность – , где m – количество информационных разрядов.

Понятие корректирующей способности кода связывают с возможностью обнаружения и исправления ошибки. Количественно корректирующая способность кода определяется вероятностью обнаружения или исправления ошибки[4].

Одной из простейших форм проверки ошибок является контроль на четность. В чем же заключается смысл кодирования по методу четности / нечетности? Его суть заключается в том, что каждой кодовой комбинации добавляется один разряд, в который записывается единица, если число единиц в кодовой комбинации нечетное, или ноль, если четное. При декодировании подсчитывается количество единиц в кодовой комбинации. Если оно оказывается четным, то поступившая информация считается правильной, если нет, то ошибочной.

Кроме проверки по горизонтали контроль на четность и нечетность может проводиться и по вертикали.

Преимущества контроля на четность заключается в минимальном значении коэффициента избыточности (для пятиэлементного кода К =0,17) и в простоте его технической реализации, а недостаток - в том, что обнаруживаются ошибки, имеющие только нечетную кратность.

Однако такая методика проверки не может обнаружить ошибки в случае двойного переброса (например, две единицы перебросились в ноль), что может привести к высокому уровню ошибок в некоторых передачах. Многоуровневая модуляция (когда проверка сигнала осуществляется по двум или трем битам) требует более сложной техники.

Проверка на четность/нечетность по одному биту также является неприемлемой и для многих аналоговых линий речевого диапазона из-за группирования ошибок, которое обычно происходит в линиях связи такого типа.

Двойная проверка на четность/нечетность является усовершенствованием одинарной проверки. В этой методике вместо бита четности в каждом символе определяется четность или нечетность целого блока символов. Проверка блока позволяет обнаруживать ошибки как внутри символа, так и между символами. Эта проверка называется также двумерным кодом проверки на четность. Она имеет значительное преимущество по сравнению с одинарной. С помощью такой перекрестной проверки может быть существенно улучшена надежность работы обычной телефонной линии, вероятность появления ошибки в которой составляет 10 . Однако, как ординарная, так и двойная проверка на четность означают увеличение накладных расходов и относительное уменьшение выхода информации для пользователя.

 Рассмотрим подробно способ нахождения одной ошибки. Если в математическом коде выделен один контрольный разряд, то к каждому двоичному числу добавляется один избыточный разряд. В этот разряд записывается 1 или 0 с таким условием, чтобы сумма цифр по модулю 2 была равна 0 для случая четности или 1 для случая нечетности. Появление ошибки в кодировании обнаруживается по нарушению четности / нечетности. При таком кодировании допускается, что может возникнуть только одна ошибка.

Пример реализации метода четности:

|  |  |  |
| --- | --- | --- |
| Число | Контрольный разряд | Проверка |
| 10101011 | 1 | 0 |
| 11001010 | 0 | 0 |
| 10010001 | 1 | 0 |
| 11001011 | 0 | 1 – ошибка |

Можно представить и несколько видоизмененный способ контроля по методу четности / нечетности. Длинное слово разбивается на группы, каждая из которых содержит n разрядов. Контрольные разряды – k, выделяются всем группам по строкам и столбцам согласно следующей схеме:



Увеличение избыточности приводит к тому, что появляется возможность не только обнаружить ошибку, но и исправить ее.

Например: число 1000111011010101110010101 представим по указанной выше схеме, получим:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 |  |

Теперь, если при передаче было получено число:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 |  |

Тогда проверка показывает, что ошибка возникла в информации третьей строки и четвертого столбца. Следовательно, разряд, содержащий ошибочную информацию, находится на пересечении третьей строки и четвертого столбца. Ошибку можно устранить, изменив 0 на 1.

## 1.4. Кодирование по методу Хэмминга

Код Хэмминга – систематический код, то есть состоящий из информационных и корректирующих символов, расположенных по строго определенной системе, имеющих одинаковую длину и всегда занимающих строго определенные места в кодовых комбинациях.

Предложенные Хэммингом регулярные методы построения кодов, корректирующих ошибки, имеют фундаментальное значение. Они демонстрируют инженерам практическую возможность достижения пределов, на которую указывали законы теории информации. Эти коды нашли практическое применение при создании компьютерных систем. Работа Хэмминга привела к решению проблемы более плотной упаковки для конечных полей. Он ввел в научный обиход важнейшие понятия теории кодирования – расстояние Хэмминга между кодовыми комбинациями в векторном пространстве, определяемом для двоичных кодов как количество позиций этих комбинаций с различными символами, и границы Хэмминга для исправляющей способности блочных корректирующих кодов. Граница Хэмминга для двоичных кодов рассчитывается по следующей формуле:

В этом выражении число ошибок e может быть исправлено корректирующим блочным кодом длиной N, имеющим М кодовых комбинаций (CjN – биномиальный коэффициент)[3].

Работа Хэмминга сыграла ключевую роль в последующем развитии теории кодирования и стимулировала обширные исследования, выполненные в последующие годы.

Коды Хэмминга позволяют не только обнаружить наличие ошибки, но и место ее нахождения и, следовательно, дают возможность ее исправить. Однако коды Хэмминга обладают меньшей избыточностью (по сравнению с кодированием по методу четности-нечетности), т.е. количеством дополнительных контрольных разрядов.

При передаче кода может быть искажен или не искажен любой символ. Если длина кода – n символов, то – полное количество комбинаций кода. По методике Хэмминга можно определить число информационных символов кода, обнаруживающего и корректирующего одиночную ошибку следующим образом:



, где

 – число информационных символов в коде;

 – число контрольных символов;

 – длина кода Хемминга.

Соотношение n, и  для кода Хэмминга можно представить в виде таблицы:

Таблица 2.2.a

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  n | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|  | 0 | 0 | 1 | 1 | 2 | 3 | 4 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 11 |
|  | 1 | 2 | 2 | 3 | 3 | 3 | 3 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 5 |

Предположим, что имеется код, содержащий m информационных и k контрольных разрядов. Все разряды, включая контрольные, разбиваются на k групп по определенным правилам, о которых будет сказано ниже. Каждая группа, содержащая один контрольный разряд, проверяется на четность. Пусть были проведены все k проверок. Если результат данной проверки свидетельствует об отсутствии ошибки, то записывается 0, если есть ошибка, то записывается 1. В результате получается последовательность, состоящая из k нулей и единиц. При отсутствии ошибки в коде получается последовательность нулей. Полученное k-разрядное двоичное число может содержать 2k различных комбинаций нулей и единиц. С помощью этой информации нужно определить ошибочный разряд в коде, содержащем m+k разрядов. Для того чтобы это было возможно должно выполняться неравенство:

2k  (m+k+1)

Определить максимальное значение m для данного k можно из следующей таблицы.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| n | 1,2,3,4… | 8,…,15 | 16,…31 | … |
| m | 0,0,1,1… | 4,…11 | 11,…26 | … |
| k | 1,2,2,3 | 4…4 | 5…5 | … |

Из таблицы видно, для 16-ти разрядного числа требуется 5 контрольных разрядов. В качестве сравнения, в случае модифицированного метода четности потребовалось бы 8 контрольных разрядов. Позиции контрольных разрядов в методе Хэмминга определены заранее, это разряды 1,2,4,8,… Разряды, входящие в каждую группу проверки представлены в следующей таблице (1-й разряд в каждой группе является контрольным).

|  |  |
| --- | --- |
| номер группы проверки | проверяемые разряды |
| 1 | 1,3,5,7,9,11,13,15,… |
| 2 | 2,3,6,7,10,11,14,15,18,19,22,23,… |
| 3 | 4,5,6,7,12,13,14,15,20,21,22,23,… |
| 4 | 8,9,10,11,12,13,14,15,24,… |

Из таблицы видно, что если, например код Хэмминга содержит 9 разрядов, включая контрольные, то 1-я группа проверки содержит 1,3,5,7,9 разряды. 2-я группа проверки содержит 2,3,6,7 разряды. 3-группа проверки содержит 4,5,6,7 разряды и 4-я группа – 8,9 разряды. Каждой группе проверки приписывается 1, если проверка на четность обнаруживает ошибку и 0, если ошибки нет. Полученное двоичное число дает номер ошибочного разряда.

Код Хэмминга имеет существенный недостаток: при обнаружении любого числа ошибок он исправляет лишь одиночные ошибки. Например, избыточность семиэлементного кода Хэмминга равна 0,43. При увеличении значности кодовых комбинаций увеличивается число проверок, но уменьшается избыточность кода. К тому же код Хэмминга не позволяет обнаружить групповые ошибки. Длина пакета ошибок представляет собой увеличенную на единицу разность между именами старшего и младшего ошибочных элементов.

Рассмотрим в качестве примера 5-ти разрядное двоичное число 10011. В этом случае, как следует из вышеприведенной таблицы, 1-я группа проверки состоит из 1,3, и 5-го разрядов. 2-я группа проверки состоит из 2 и 3-го разряда. 3-я группа проверки состоит из 4 и 5-го разрядов. Результат проверки на четность 1-й группы дает 0 (101), проверка 2-й группы дает 0 (00), проверка 3-й группы дает 0 (11).

k1 = 1 + 0 + 1 = 0 – нет ошибки;

k2 = 0 + 0 =0 – нет ошибки;

k3 = 1 + 1 = 0 – нет ошибки.

Таким образом, данное число не содержит ошибки. Искусственно введем ошибку, заменив, например, 4-й разряд на 0. В этом случае 1, 2 и 3-я проверки дадут соответственно 0, 0, 1.

k1 = 1 + 0 + 1 = 0 – нет ошибки;

k2 = 0 + 0 =0 – нет ошибки;

k3 = 0 + 1 = 1 – ошибка.

Полученное двоичное число 100 дает номер ошибочного разряда, т.е. 4.

**1.5. Машина Поста**

**«Внешний вид» машины Поста**

Машина Поста не есть реально существующее, сделанное кем-то устройство; поэтому слова «внешний вид» и взяты в кавычки. Машина Поста, как и ее близкий родствен­ник - машина Тьюринга, - представляет собой мысленную конструкцию, существующую лишь в нашем воображении(хотя ее в принципе и можно было бы изготовить «в металле»). Именно это имеют в виду, когда говорят о машинах Поста и Тьюринга, что они суть «абстрактные» вычислительные машины. И подобно тому, как можно выучиться считать на счетах или на логарифмической линейке, не имея перед собой этих приборов, а, пользуясь лишь их описаниями и представляя их себе мысленно, так же и мы научимся вычислять на машине Поста.

Машина Поста состоит из ленты и каретки (назы­ваемой также считывающей и записывающей головкой).

Лента бесконечна и разделена на секции одинакового размера: для наглядности ленту будем считать расположенной горизонтально (рис. 13).

Бесконечность ленты находится в противоречии со сделанным выше утверждением, что машину Поста можно было бы в принципе построить. Дело в том, что мы объявили ленту бесконечной лишь для простоты изложения. С тем же успехом можно было бы предположить, что лента не бесконечная, а лишь неограниченно растущая в обе стороны: например, можно было бы считать, что лента наращивается на одну секцию, как только каретка доходит до конца ленты и должна двигаться дальше, или считать, что за каждую единицу времени слева и справа нарастает по одной секции. Однако будет удобнее считать, что все секции слева и справа уже наросли, и тем самым, хотя и в ущерб реальности, полагать ленту бесконечной в обе стороны.

Порядок, в котором расположены секции ленты, подобен порядку, в котором расположены все целые числа. Поэтому естественно ввести на ленте «целочисленную систему координат», занумеровав секции целыми числами ..., -3,-2,-1,0,1, 2,3, ... (рис. 14).

Мы будем считать, что система координат жестко сопоставлена с лентой, и получим, таким образом, воз­можность указывать какую-либо секцию ленты, называя ее порядковый номер, или координату. (Иногда, впрочем, бывает удобно наряду с основной, «постоянной» системой координат, ввести еще вспомогательную, «временную», сдвинутую по отношению к первоначальной)

В каждой секции ленты может быть либо ничего не записано (такая секция называется пустой), либо записана метка V (тогда секция называется отмеченной) (рис. 15).

Информация о том, какие секции пусты, а какие отмечены, образует состояние ленты. Иными словами, со­стояние ленты — это распределение меток по ее сек­циям. Как мы далее увидим, состояние ленты ме­няется в процессе работы машины.

Каретка может передвигаться вдоль ленты влево и вправо. Когда она неподвижна, она стоит против ровно одной секции ленты на этом и следующих чертежах каретка изображена в виде зачерненного квадрата); говорят, что каретка обозревает эту секцию, или держит ее в поле зрения.

Информация о том, какие секции пусты, а какие отмечены и где стоит каретка, образует состояние машины Поста.

Таким образом, состояние машины слагается из состояния ленты и указания номера той секции, которую обозревает каретка. За единицу времени (которую мы будем называть шагом) каретка может сдвинуться на одну секцию влево или вправо. Кроме того, каретка может поставить (напечатать) или уничтожить (стереть) метку на той секции, против которой она стоит, а также распознать, стоит или нет метка в обозреваемой ею секцией[2].

**Программа машины Поста**

Работа машины Поста состоит в том, что каретка передвигается вдоль ленты и печатает или стирает метки. Эта работа происходит по инструкции определенного вида, называемой программой. Для машины Поста возможно составление различных программ. Посмотрим, как устроена программа.

Каждая программа машины Поста состоит из команд. Командой машины Поста будем называть выражение, имеющее один из следующих шести видов (буквы i, j, j1, j2 означают всюду натуральные числа 1, 2, 3, 4, 5, …):

Первый вид. Команды движения вправо.

Второй вид. Команды движения влево.

Третий вид. Команды печатания метки.

Четвертый вид. Команды стирания метки.

Пятый вид. Команды передачи управления.

Шестой вид. Команды остановки.

Например,

является командой движения вправо,

- командой передачи управления, а

-командой остановки.

Число i, стоящее в начале команды, называется номером команды. Так, у приведенных только что команд номера соответственно 137, 25 и 6386. Число j, стоящее в конце команды (а у команд передачи управления - каждое из чисел j1, и j2), будем называть отсылкой (при этом в команде передачи управления j1—верхней, а j2 — нижней отсылкой). У команд остановки нет отсылки. Так, у приведенных только что команд отсылками служат числа 1, 32, 25, причем 32 —верхняя отсылка, а 25 — нижняя отсылка.

Программой машины Поста будем называть конечный непустой (т. е. содержащий хотя бы одну команду) список команд машины Поста, обладающий следующими двумя свойствами:

1) На первом месте в этом списке стоит команда с номером 1, на втором месте (если оно есть) – команда с номером 2 и т. д.; вообще на k-м месте стоит команда с номером k.

2) Отсылка любой из команд списка совпадает с номером некоторой (другой или той же самой) команды списка (более точно: для каждой отсылки каждой команды списка найдется в.списке такая команда, номер которой равен рассматриваемой отсылке).

Например, следующий список будет программой машины Поста:

А эти два списка не будут программами машины Поста,хотя и составлены из команд машины Поста:

Для наглядности программы машины Поста мы будем записывать столбиком. Число команд программы называется длиной программы[1].

**Работа машины Поста**

Чтобы машина Поста начала работать, надо задать, во-первых, некоторую программу, а во-вторых, некоторое ее (машины) состояние, т. е. как-то расставить метки по секциям ленты (в частности, можно все секции оставить пустыми) и поставить каретку против одной из секций. Как правило, мы будем предполагать, что в начальном (т.е. в задаваемом вначале) состоянии машины каретка ставится всегда против секции с номером (координатой) нуль. При таком соглашении начальное состояние машины полностью определено состоянием ленты.

Как уже говорилось, программа является той инструкцией, на основании которой работает машина. Работа машины на основании заданной программы (и при заданном начальном состоянии) происходит следующим образом. Машина приводится в начальное состояние и приступает к выполнению первой команды программы. Эта команда выполняется за один шаг, после чего машина приступает к выполнению той команды, номер которой (назовем его а) равен отсылке (одной из отсылок, если их две) первой команды. Эта команда также выполняется за один шаг, после чего начинается выполнение команды, номер которой равен отсылке команды с номером а. Вообще каждая команда выполняется за один шаг, а переход от выполнения одной команды к выполнению другой происходит по следую­щему правилу: пусть на k-м шаге выполнялась команда с номером i, тогда, если эта команда имеет единственную отсылку j, то на k+1-м шаге выполняется команда с номером j; если эта команда имеет две отсылки j1 и j2, то на k+1-м шаге выполняется одна из двух команд – с номером j1 или с номером; если, наконец, выполняющаяся на k-м шаге команда вовсе не имеет отсылки, то на k + 1-м шаге и на всех последующих шагах не выполняется никакая команда: машина останавливается. Осталось объяснить, что значит выполнить команду и какая из отсылок - при наличии двух - выбирается в качестве номера следующей команды.

Выполнение команды движения вправо состоит а том, что каретка сдвигается на одну секцию вправо. Выполнение команды движения влево состоит в том, что каретка сдвигается на одну секцию влево. Выполнение команды печатания метки состоит в том, что каретка ставит метку на обозреваемой секции; выполнение этой команды возможно лишь в том случае, если обозреваемая перед началом выполнения команды секция пуста; если же на обозреваемой секции уже стоит метка, команда считается невыполнимой. Выполнение команды стирания метки состоит в том, что каретка уничтожает метку в обозреваемой секции; выполнение этой команды воз­можно лишь в том случае, если обозреваемая секция отмечена; если же на обозреваемой секции и так нет метки, команда считается невыполнимой. Выполнение команды передачи управления с верхней отсылкой j1 и нижней отсылкой j2 никак не изменяет состояния машины: ни одна из меток не уничтожается и не ставится, и каретка также остается неподвижной (машина делает, так сказать, «шаг на месте»); однако если секция, обозреваемая перед началом выполнения этой команды, была пуста, то следующей должна выполняться команда с номером j1, если же эта секция была отмечена, следующей должна выполняться команда с номером j2 (роль команды передачи управления сводится, следовательно, к тому, что каретка во время выполнения этой команды как бы «распознает», стоит ли перед ней метка). Выполнение команды остановки тоже никак не меняет состояния машины и состоит в том, что машина останавливается.

Если теперь, задавшись программой и каким-либо начальным состоянием, пустить машину в ход, то осуществится один из следующих трех вариантов:

1) В ходе выполнения программы машина дойдет до выполнения невыполнимой команды (печатания метки в непустой секции или стирания метки в пустой секции); выполнение программы тогда прекращается, машина останавливается; происходит так называемая безрезультатная остановка.

2) В ходе выполнения программы машина дойдет до выполнения команды остановки; программа в этом случае считается выполненной, машина останавливается; происходит так называемая результативная остановка.

3) В ходе выполнения программы машина не дойдет до выполнения ни одной из команд, указанных в первых двух вариантах; выполнение программы при этом никогда не прекращается, машина никогда не останавливается; процесс работы машины происходит бесконечно[5].

**Глава 2. Инструкция программиста**

Данная учебная программа должна запускаться на IBM и совместимых компьютерах. Минимальные системные требования: процессор Pentium и выше, объем оперативной памяти 16MB, видеокарта с расширением 800x600, 256 цветов, операционная система семейства Windows 95 и выше, на компьютере также должен быть установлен .NET Framework.

Основную ценность представляют файлы расширением \*.cs (в них содержится весь код программы).

Опишем основные файлы, содержащиеся в этой папке:

* Even\_ueven.cs – файл, в котором содержится код, отвечающий за работу алгоритма «Кодирование по методу четности\нечетности»;
* Even\_ueven.resx – файл, в котором содержится информация о графическом представлении формы «Кодирование по методу четности\нечетности»;
* Hemming.cs - файл, в котором содержится код, отвечающий за работу алгоритма «Кодирование по методу Хэмминга»;
* Hemming.resx – файл, в котором содержится информация о графическом представлении формы «Кодирование по методу Хэмминга»;
* Post.cs - файл, в котором содержится код, отвечающий за работу алгоритма «Программирование машины Поста»;
* Post.resx – файл, в котором содержится информация о графическом представлении формы «Программирование машины Поста»;
* Error.cs - файл, в котором содержится код, отвечающий за работу формы Error;
* Error.resx – файл, в котором содержится информация о графическом представлении формы Error;
* Help.cs - файл, в котором содержится код, отвечающий за работу формы Справка;
* Help.resx – файл, в котором содержится информация о графическом представлении формы Справка;
* Start.cs - файл, в котором содержится код, отвечающий за работу формы Start;
* Start.resx – файл, в котором содержится информация о графическом представлении формы Start;
* AssemblyInfo.cs - файл, в котором содержится информация о версии файла, внутреннем и исходном имени файла, производителе;
* StudyProgram.exe - закомпилированный файл программы, находящийся в папке debug, которая находится в папке bin.

Для изменения программы необходимо изменить ее исходный код. Это можно сделать, используя любой текстовый редактор. Пример кода представлен на Рис.2.

Изменение кода при помощи текстового редактора не очень удобно, для этого лучше использовать Visual Studio .Net, специально для этого разработанный.

На Рис.3 можно увидеть внешний облик Visual Studio .Net. Для его запуска необходимо кликнуть кнопку Пуск, выбрать в пункте Все программы \Microsoft Visual Studio .NET 2003\Microsoft Visual Studio .NET 2003. Далее в запустившемся приложении кликнуть кнопку Open Project и используя проводник, выбрать файл Study\_Programs.sln. Программа откроет приложение. Для изменения внешнего вида программы необходимо воспользоваться ярлыком Design. Можно добавить новые элементы формы, перетащив их из выезжающего окна Toolbox. Исходный код программы см. в Приложении.

В самом начале программы задаются поля, так называемые член- переменные, содержащие некоторое значение: int Prav; int ch; int ChVoprosov=1 (в ООП поля иногда называют данными объекта).Где Prav-это количество правильных ответов, ch – это сумма всех разрядов генерируемого числа и контрольного разряда, а ChVoprosov-число вопросов заданных пользователю (нумерация начинается с 1) Перед идентификаторами полей указываются их типы.

При запуске формы Even\_ueven во время работы программы сразу срабатывает метод Generation(); .Суть данного метода состоит в том, чтобы сгенерировать произвольное семизначное двоичное число, которое появится в окне textBox1 и контрольный разряд(который также является двоичным числом), который появится в окне textBox2. Для этого создается генератор случайных чисел, который записывает в переменную d либо 0, либо 1( d=r.Next(0,2); ), данная операция повторяется 7 раз с помощью цикла с параметром for( for (int i=0; i<7; i++) ). В результате, последовательно добавив каждую из сгенерированных цифр, мы получаем требуемое семизначное число. Контрольный разряд создается тоже с помощью генератора случайных чисел ( g=r.Next(0,2); ).Также в этом методе сразу подсчитывается сумма всех разрядов генерируемого числа и контрольного разряда ch: сначала складываются разряды числа все в том же цикле for (ch+=d; - 7 раз), а затем прибавляется контрольный разряд g (ch+=g;).

При нажатии кнопки button1 (Код числа правильный) вначале запускается метод ChangeChVoposov(); который увеличивает номер вопроса на одну единицу в окне textBox3. Затем в поле Prav(количество правильных ответов) считывается значение из окна textBox4( при первом запуске оно равняется 0), Далее с помощью управляющей конструкции if (ch==0 || ch==2 || ch==4 || ch==6) определяется правильно ли пользователь ответил : ch - сумма всех разрядов генерируемого числа и контрольного разряда должна быть четной или равной 0. В случае верного выбора количество правильных ответов увеличивается на 1 ( Prav++; ). Затем запускается метод ChangePrav();, который устанавливает число правильных ответов в окне textBox4. После чего ch обнуляется ( ch=0; ) , и запускается метод Generation();.

При нажатии кнопки button2 (Код числа неправильный) алгоритм программы является практически аналогичным за исключением того, что в управляющей конструкции if (ch==1 || ch==3 || ch==5 || ch==7) правильность ответа пользователя заключается в том, чтобы сумма ch являлась нечетным числом.

Также следует обратить внимание на то, что при работе с дизайном формы нужно заблокировать окна от пользователя, чтобы он не смог перепечатать значение генерируемых семиразрядного числа и контрольного разряда, а также количество заданных вопросов и данных правильных ответов. Исходный код программы представлен в Приложении.

**Кодирование по методу Хэмминга**

В самом начале программы задаются поля: int ChVoprosov=1,

int X; int Nomer; int [] x=new int[10] .Где X – это переменная значение которой не позволяет в определенных случаях нажиманием на одну и ту же кнопку увеличивать число правильных ответов, Nomer – это получаемый номер разряда, который подлежит замене на противоположный, int [] x=new int[10] – это массив, состоящий из значений, которые являются разрядами генерируемого двоичного числа, а ChVoprosov-число вопросов заданных пользователю (нумерация начинается с 1). Перед идентификаторами полей указываются их типы.

При запуске формы Hemming во время работы программы сразу срабатывает метод Generation(); и метод Number(); .Суть первого метода состоит в том, чтобы сгенерировать произвольное девятизначное двоичное число, которое появится в окне textBox1. Суть второго метода заключается в том, чтобы подсчитать номер разряда, в котором есть ошибка (если конечно такова вообще имеется). Следует учесть, что номер подсчитывается в двоичном виде.

Для формирования произвольного девятизначного двоичного числа создается генератор случайных чисел, который записывает в переменную d либо 0, либо 1( d=r.Next(0,2); ), данная операция повторяется 9 раз с помощью цикла с параметром for( for (int i=1; i<10; i++) ). В результате, последовательно добавив каждую из сгенерированных цифр, мы получаем требуемое девятизначное число. Также на каждом шаге цикла с параметром считываются значения в массив int [] x ( x[i]=d; ), т.е. получается , что каждый элемент массива равен соответствующему по номеру разряду сгенерированного числа. В данном методе переменная X приравнивается к 1, что означает, что число только что сгенерированно и пользователь еще не пробовал дать свой ответ.

В методе Number() подсчитываются суммы контрольных разрядов сгенерируемого по числа, соответствующие определенным группам проверки. В нашем случае количество групп проверки для девятизначного числа равно 4, следовательно, мы получаем четыре суммы x1, x2, x3, x4.

Затем подсчитывается номер ошибочного разряда (Nomer=x4\*1000+x3\*100+x2\*10+x1;)

При нажатии кнопки button1 (Ошибки нет) в начале исчезает все то, что пользователь успел ввести в окно «Исправленный вариант» (textBox2.Text="";), т.к. нажав эту кнопку пользователь показывает то, что он уже не находит ошибок в числе. Потом с помощью управляющей конструкции if проверяется не записан ли уже ответ в окне «Правильный ответ»( if(textBox1.Text.Length!=textBox3.Text.Length) ели не записан, то тогда X приравнивается к единице: X=1;) Затем с помощью управляющей конструкции if проверяется, равен ли X единице (if (X==1) – если это выполнено, то значит, что пользователь еще не пробовал дать свой ответ). Дальнейшее выполнение действий возможно только если X=1. Затем, используя снова if, Nomer сравнивается с нулем (if (Nomer==0)). В случае выполнения данного условия (т.е. номер ошибочного разряда равняется нулю, что означает отсутствие ошибки) число правильных ответов увеличивается на одну единицу ( Prav++; ), и запускается метод Otvet(); . Данный метод записывает в окно textBox3 (Правильный ответ) ответ, получаемый путем записи всех элементов массива int [] x в порядке возрастания:

textBox3.Text=x[1]+""+x[2]+""+x[3]+""+x[4]+""+x[5]+""+[6]+""+x[7]+""+x[8]+""+x[9];

В данном случае ошибка отсутствует и все элементы запишутся без изменения, т.е. в окне textBox3 появится число, аналогичное тому, которое находится в окне textBox1 (Число, возможно содержащее ошибку).

Если Nomer не равняется нулю, то тогда запускается метод Sravnenie();. Этот метод вначале проверяет не дали ли вы уже свой окончательный ответ (не равно ли уже X нулю) и только в случае равенства X единице (if (X==1)) выполняется вся остальная часть метода: переменную i приравнивают т к нулю (int i=0;). Затем происходит сравнение числа Nomer с двоичными цифрами, которые при переводе в десятичную систему счисления соответствуют числам от 0 до 9 (например: if (Nomer == 101) – число Nomer сравнивается с двоичным числом 101, которому в десятичной системе счисления соответствует 5). Если Nomer равняется какому-то из предложенных двоичных чисел, то тогда переменной i присваивается значение соответствующего десятичного числа (Например в случае равенства номера числу 101, переменной i присвоится значение 5 (i = 5;)). Если переменная i меняет свое значение с нуля на какое-то от 1 до 9, то тогда выполняется условие следующей за этим управляющей конструкции if (i!=0) (i отлично от нуля) и осуществляется следующие операции: x[i] сравнивается с 0, если это выполняется, то тогда x[i] меняет свое значение на противоположное, т.е. на 1, и затем запускается уже упомянутый выше метод Otvet(); (только теперь в окне textBox3 появится число, аналогичное тому, которое находится в окне textBox1 (Число, возможно содержащее ошибку), но уже с замененным на противоположный разрядом, номер которого равен i.). Если x[i] не равняется 0(следовательно равняется 1), то тогда выполняется ветка else с аналогичными действиями, только для x [i] противоположным значением будет являться 0.

Если в ходе выполнения метода Sravnenie(); не было произведено никаких замен (не выполнялся метод Otvet();), то это означает, что числу Nomer соответствует какое-то двоичное число, которое при переводе в десятичную систему счисления дает цифру большую 9. Это означает, что в сгенерированном девятиразрядном числе отсутствуют ошибки. Тогда с помощью сравнения длины строки в окне textBox3 с нулем (if(textBox3.Text.Length==0)) мы получаем в случае выполнения данного условия, что ошибки отсутствуют. Следовательно, мы можем увеличить число правильных ответов на 1 и запустить метод Otvet();,который запишет в окно textBox3 число, равное тому, которое находится в окне textBox1 (Число, возможно содержащее ошибку).

В итоге мы меняем число правильных ответов в окне textBox5, а также приравниваем X к нулю, это означает, что мы сделали свой окончательный выбор того, содержит ли сгенерированное число ошибки.

При нажатии кнопки button2 (Ошибка есть) в начале запускается метод Proverka();. Смысл этого метода заключается в том, чтобы проверить ввел ли пользователь достаточное количество разрядов в окно «Исправленный вариант» и являются ли они двоичными числами (0 и 1). Вначале с помощью управляющей конструкции if проверяется, записал ли пользователь строку в окно «Исправленный вариант» требуемой длины и нет ли уже появившегося ответа в окне textBox3 (Правильный ответ): if(textBox2.Text.Length! =textBox1.Text.Length && textBox1.Text.Length! = textBox3.Text.Length)

Если это условие выполняется, то это означает, что пользователь ввел в окно «Исправленный вариант» строку меньшую по своей длине 9 или то, что уже был получен ответ и, следовательно, выполняется действие, заключенное в фигурных скобках: запуск формы об ошибке, содержащее пояснение того, в чем заключается ошибка ввода (A5=new Error("Исправленный вариант должен содержать 9-ти разрядное число"); A5.ShowDialog(); )

Если это условие не выполняется, то работа алгоритма программы продолжается по ветке else. Вначале снова проверяется отсутствие ответа if(textBox1.Text.Length!=textBox3.Text.Length) и только когда это условие выполнено запускается цикл с параметром (for(int i=0; i<textBox2.Text.Length; i++)) , который сравнивает все разряды введенного пользователем числа с 0 или 1.Для этого используется заложенный в языке C# метод работы со строками Substring(i,1) (string d=textBox2.Text.Substring(i,1); - в строку d записывается один символ введенного пользователем числа , начиная с i-ой позиции (а i меняется от 0 до длины введенной строки, т.е. до 9 )). В ходе работы цикла d 9 раз сравнивается с 0 или 1 (if(d!=b && d!=a)) ,и при первом же случая, когда d не является 1 или 0 появляется форма ошибки, содержащее пояснение того, в чем заключается ошибка ввода (A5=new Error("Исправленный вариант должен содержать 9-ти разрядное двоичное число"); A5.ShowDialog(); ), а цикл прекращается, т.к. переменной i присваивается значение равное textBox2.Text.Length. Также в случае появления ошибки переменная X приравнивается к 0. В случае отсутствии ошибки ввода переменная X приравнивается к 1.На этом метод Proverka(); заканчивается. Если X=1, то возможны дальнейшие операции заключенные в фигурных скобках условия if (X==1). Используя условную конструкцию if, Nomer сравнивается с нулем (if (Nomer==0)). Если данное условие выполняется (т.е. номер ошибочного разряда равняется нулю, что означает отсутствие ошибки), то пользователь ответил неправильно, неверно предположив, что содержится ошибка. Затем запускается метод Otvet(), который выведет число в окне textBox3, равное сгенерированному числу.

После этого запускается метод Sravnenie(); (его запуск требуется, если не выполняется условие Nomer==0). Если Nomer не равен нулю и метод Sravnenie(); не дает правильного ответа в окне textBox3,, то, следовательно, Nomer при переводе в десятичную систему счисления дает число, большее, чем 9. Это означает, что сгенерированное число не содержит неправильных разрядов и в окно textBox3 нужно записать заданное число без изменений. Для этого используется условная конструкция, которая проверяет, записан ли ответ в textBox3: if (textBox3.Text.Length==0) Otvet();. Затем в переменную Moe считывается число, которое пользователь ввел в окно textBox2, а в переменную Pravilnoe считывается ответ из textBox3. После чего эти две переменные сравниваются между собой и в случае равенства количество правильных ответов увеличивается на 1: if (Moe==Pravilnoe) Prav++; . Затем переменная X приравнивается к нулю, что означает, что пользователь уже сделал свой окончательный выбор.

Кнопка button3 служит для того чтобы сгенерировать следующее число, когда работа с предыдущим закончена. При нажатии кнопки button3 (Следующее число) число вопросов увеличивается на единицу, а также запускаются методы: Generation() и Number();

**Машина Поста**

Для того чтобы лучше понять, как изменить программу, подробнее разберем ее код (см. Приложение).

private void button2\_Click(object sender, System.EventArgs e) – метод, вызывающийся при нажатии кнопки «Вернуться к выбору темы».

private void help\_Click(object sender, System.EventArgs e) - метод, вызывающийся при нажатии кнопки «Справка».

private void button3\_Click(object sender, System.EventArgs e) - метод, вызывающийся при нажатии кнопки «GO».

public static byte first() – метод, который запускается при выполнении команды типа «Движение головки на одну клетку вправо и переход к команде m1».

public static byte second() – метод, который запускается при выполнении команды типа «Движение головки на одну клетку влево и переход к команде m1».

public static byte third() – метод, который запускается при выполнении команды типа «Нанесение метки в ячейку и переход к команде m1.».

public static byte fourth() – метод, который запускается при выполнении команды типа «Стирание метки в ячейке и переход к команде m1.».

public void fifth() – метод, который запускается при выполнении команды типа «Проверка наличия метки в клетке, над которой находится головка. Если метка отсутствует, то управление передается с номером m1, если метка есть, то с номером m2.».

bool check\_string(string str) метод, вызываемый для проверки командной строки.

Разберем подробно работу программы.

Задаются два массива типа char input[] и command[][]. При нажатии кнопки «GO» формируется массив из входной строки и записывается в input[]. Далее этот массив проверяется, если в нем содержатся символы, отличные от 0 и 1, то выдается ошибка. Затем проверяется список команд в методе check\_string(string str), если он не соответствует заданному шаблону, то выдается ошибка, в противном случае он записывается в массив command[][]. После этого запускается цикл с постусловием, в котором проверяется какая команда должна выполняться в данный момент и запускается соответствующий метод. Во время выполнения команды останова – цикл завершается и выводится результат работы программы.

**Глава 3. Инструкция пользователя**

Программа StudyProgram должна запускаться на IBM и совместимых компьютерах. Минимальные системные требования: процессор Pentium и выше, объем оперативной памяти 16MB, видеокарта с расширением 800x600, 256 цветов, операционная система семейства Windows 95 и выше, на компьютере также должен быть установлен .NET Framework.

## 3.1 Использование программы StudyProgram для усвоения учебного материала по кодированию информации методом четности и методом Хэмминга

Программа StudyProgram предоставляет возможность выполнить упражнения по кодированию информации методом четности и методом Хэмминга. Для запуска программы сделайте двойной щелчок по ярлыку программы. После запуска программы на экране появляется окно «StudyProgram», представленное на Рис.4.:

Пункт меню StudyProgram содержит следующие разделы:

«Программирование машины Поста»

«Кодирование по методу четности\нечетности»

«Кодирование по методу Хэмминга»

При выборе пункта меню «Кодирование по методу четности\нечетности» появляется диалоговое окно.

В данном диалоговом окне предлагаются вопросы по теме «Кодирование по методу четности\нечетности». Число вопросов неограниченно. После каждого ответа показывается номер задаваемого вопроса и количество правильных ответов.

Диалоговое окно содержит три кнопки: «Код числа правильный», «Код числа неправильный», «Справка» и «Вернуться к выбору темы». Вверху расположено текстовое поле «Цифровые разряды», в котором задано случайное семизначное двоичное число. Чуть ниже расположено текстовое поле «Контрольного разряд», в котором задан контрольный разряд, с помощью которого происходит кодирование случайного семизначного двоичного числа по методу четности. Нужно определить присутствует или не присутствует ошибка в записи числа. Если число записано правильно, нужно щелкнуть по кнопке «Код числа правильный». Если число содержит ошибку, нужно щелкнуть кнопку «Код числа неправильный». Если ответ был правильным, счетчик правильных ответов увеличивается на единицу. Переход к следующему заданию происходит автоматически при нажатии на любую из двух кнопок: «Код числа правильный», «Код числа неправильный», при этом счетчик заданных вопросов увеличивается на единицу. Число заданий неограниченно. Чтобы прервать работу с диалоговым окном, нужно щелкнуть по кнопке «Вернуться к выбору темы» при этом счетчик правильных ответов обнуляется и при следующем запуске раздела «Кодирование по методу четности\нечетности» количество верных ответов снова равно нулю. Если пользователю требуется помощь по теории, то тогда можно щелкнуть по кнопке «Справка», и тогда появится диалоговое окно Help(см. Рис.6)

Диалоговое окно, которое появляется при выборе пункта меню «Кодирование по методу Хэмминга» имеет вид.

В данном диалоговом окне предлагается найти и исправить одну ошибку (если она имеется) в 9-ти разрядном двоичном числе, закодированном по методу Хэмминга. В диалоговом окне «Кодирование по методу Хэмминга» имеется 5 кнопок: «Ошибки нет», «Ошибка есть», «Следующее число», «Справка», «Вернуться к выбору темы». Вверху расположены текстовые поля: «Число, возможно содержащее ошибку», «Исправленный вариант», «Правильный ответ». Если пользователь считает, что число появившееся в текстовом поле «Число, возможно содержащее ошибку» не содержит ошибочных разрядов, то следует щелкнуть по кнопке «Ошибки нет». После чего в текстовом поле «Правильный ответ» записывается верный ответ. Если пользователь отвечает правильно, то число правильных ответов увеличивается на 1. Если пользователь считает, что число появившееся в текстовом поле «Число, возможно содержащее ошибку» содержит ошибку в одном из разрядов числа, то тогда результат корректировки нужно записать в текстовое поле «Исправленный вариант», после чего нажать кнопку «Ошибка есть». Пользователь должен учесть, что вводимое им число в текстовое поле «Исправленный вариант» должно быть двоичным и содержать 9 разрядов. В случае неправильного ввода на экран выведется сообщение об ошибке, см. Рис.8:

Если ответ правильный, счетчик правильных ответов увеличивается на единицу, при этом на экране появляется правильный вариант ответа. После того, как пользователь дал свой ответ нужно нажать кнопку «Следующее число», и тогда программа сгенерирует новое число, возможно содержащее ошибку. Работа с диалоговым окном завершается нажатием кнопки «Вернуться к выбору темы» при этом счетчик правильных ответов обнуляется. При работе с данным разделом пользователь может просмотреть информацию по кодированию по методу Хэмминга, нажав кнопку «Справка».

## 3.2 Использование обучающей программы StudyProgram для овладения навыками составления программ для машины Поста

Программа StudyProgram предназначена для того, чтобы помочь в усвоении приемов составления программ для машины Поста. Работа с программой осуществляется следующим образом. После запуска программы на экране появляется окно «StudyProgram», представленное на Рис.4. В нем имеется 3 раздела. Выберите 1-й раздел «Программирование машины Поста».

При выборе данного пункта меню появляется диалоговое окно.

Диалоговое окно содержит три кнопки: «GO», «Справка» и «Вернуться к выбору темы». Вверху расположено текстовое поле «Входная строка машины Поста». Сюда вводится входная строка машины Поста. Каждый символ строки соответствует одной ячейке ленты, 0 - отсутствие метки, 1 – наличие метки. Предполагается, что головка ленты находится над 1-й слева ячейкой. В окно, расположенное под данным окном вводится набор команд, выполняющий поставленную задачу. Каждая команда должна состоять из 4-х целых положительных чисел. 1-е число – это порядковый номер команды, 2-е число – тип команды, 3-е число – номер следующей команды при безусловном переходе и 4-е число – номер следующей команды при условном переходе. Если команда не является командой условного перехода, то 4-е число может быть любым. Числа в команде должны отделяться друг от друга запятой, а каждая команда заканчиваться точкой с запятой.

Порядок записи и выполнения программы следующий:

1) В окно «Входная строка машины Поста» введите произвольную строку, состоящую из нулей и единиц, для которой поставленная задача имеет смысл. Например, если задача состоит в стирании 1-й метки, то, чтобы проверить правильность работы программы, во входной строке должна присутствовать хотя бы одна метка, т.е. единица.

2) В нижележащее окно введите в соответствии с выше перечисленными правилами набор команд, решающих поставленную задачу.

3) Нажмите кнопку «GO», после чего записанные команды будут выполнены и в поле «Выходная строка» появится результат работы машины Поста в виде выходной строки символов, состоящей из нулей и единиц.

В общем виде отдельная команда машины Поста имеет следующий вид: n,K,m1,m2; где n – порядковый номер команды, К – тип команды, m1, m2 – порядковые номера следующих команд. Имеется шесть типов команд машины Поста, которым для удобства можно дать номера от 1 до 6. Ниже дано описание команд машины Поста.

|  |  |
| --- | --- |
| Тип команды | Действие команды |
| 1 | Движение головки на одну клетку вправо и переход к команде m1 |
| 2 | Движение головки на одну клетку влево и переход к команде m1 |
| 3 | Нанесение метки в ячейку и переход к команде m1 |
| 4 | Стирание метки в ячейке и переход к команде m1 |
| 5 | Проверка наличия метки в клетке, над которой находится головка. Если метка отсутствует, то управление передается команде с номером m1, если метка есть, то с номером m2 |
| 6 | Остановка машины |

Пример команды машины Поста: 1,1,2,0;. Данная команда имеет порядковый номер 1, тип команды – 1 (движение головки на одну клетку вправо). После выполнения команды управление передается команде, имеющей порядковый номер 2.

Рассмотрим пример простой программы, которая стирает первую метку, записанную на ленте. Предположим, что в начале работы головка находится над крайней левой ячейкой ленты. Поставленная задача может быть выполнена с помощью следующего набора команд.

* 1, 5, 2, 4; 1-я команда
* 2, 1, 3, 0; 2-я команда
* 3, 5, 2, 4; 3-я команда
* 4, 4, 5, 0; 4-я команда
* 5, 6, 0, 0; 5-я команда

Разберем работу программы. 1-я команда – это команда типа 5, т.е. условный переход. Если ячейка пустая, то управление передается команде 2, в противном случае – 4-й команде. 2-я команда – это сдвиг головки на одну ячейку вправо и передача управления 3-й команде. 3-я команда – это опять условный переход. Если метка отсутствует, то возврат ко второй команде и переход на 4-ю в противном случае. 4-я команда стирает метку (очевидно, она там есть, т.к. мы попадаем на 4-ю команду только при наличии метки) и передает управление 5-й команде, которая является командой остановки машины.

При нажатии кнопки «Справка» выводится окно со справочным материалом по данной теме.

При нажатии кнопки «Вернуться к выбору темы» Вы возвращаетесь на стартовое окно программы.

Если входная строка введена неправильно, то выведется сообщение об ошибке Рис.11.

Если список команд не соответствует заданному шаблону, то выведется сообщение об ошибке.

#

# Заключение

В ходе курсовой работы была разработана обучающая программа по информатике, с помощью которой студенты смогут проверить свои знания в таких разделах данной дисциплины как кодирование по методу Хэмминга, кодирование по методу четности-нечетности и программирование машины Поста.

Разработанная курсовая работа содержит следующие разделы:

* Исследовательская часть, содержащая разделы, посвященные целесообразности выбора языка, основным структурам языка программирования C#, кодированию по методу четности / нечетности, кодированию по методу Хэмминга, а также программированию машины Поста.
* Инструкция программиста, которая содержит общую информацию по созданию и изменению обучающей программы, а также подробно расписанный алгоритм работы программы.
* Инструкция пользователя, включающая в себя подразделы: использование программы StudyProgram для усвоения учебного материала по кодированию информации методом четности и методом Хэмминга; использование обучающей программы StudyProgram для овладения навыками составления программ для машины Поста.

 Задание курсовой работы выполнено полностью в соответствии с заданием.

# Список литературы

1. Акулов О.А., Медведев Н.В. «Информатика базовый курс». Москва, Омега-Л, 2005
2. Успенский В.А. «Машина Поста». Москва, Наука, 1988
3. http://computer-museum.ru
4. http://pascalstudy.narod.ru
5. http://de.uspu.ru
6. http://c-sharp.ru
7. http://dotsite.spb.ru
8. http://i-u.ru
9. http://mf.grsu.by
10. Троелсен Э. «С# и платформа .NET. Библиотека программиста»

# Приложение

Start

using System;

using System.Drawing;

using System.Collections;

using System.ComponentModel;

using System.Windows.Forms;

using System.Data;

namespace Study\_Programs

{

 public class Start : System.Windows.Forms.Form

 {

 private System.Windows.Forms.Label label1;

 private System.Windows.Forms.GroupBox groupBox1;

 private System.ComponentModel.Container components = null;

 Post A1;

 Even\_uneven A2;

 Hemming A3;

 private System.Windows.Forms.Button even\_uneven\_button;

 private System.Windows.Forms.Button hemming\_button;

 private System.Windows.Forms.Button exit\_button;

 private System.Windows.Forms.Button post\_button;

 public Start(){InitializeComponent();}

 protected override void Dispose( bool disposing )

 {

 if(disposing) if (components != null) components.Dispose();

 base.Dispose( disposing );

 }

 #region Windows Form Designer generated code

 private void InitializeComponent()

 {

 this.label1 = new System.Windows.Forms.Label();

 this.groupBox1 = new System.Windows.Forms.GroupBox();

 this.hemming\_button = new System.Windows.Forms.Button();

 this.post\_button = new System.Windows.Forms.Button();

 this.even\_uneven\_button = new System.Windows.Forms.Button();

 this.exit\_button = new System.Windows.Forms.Button();

 this.groupBox1.SuspendLayout();

 this.SuspendLayout();

 //

 // label1

 //

 this.label1.Dock = System.Windows.Forms.DockStyle.Top;

 this.label1.Font = new System.Drawing.Font("MS Reference Sans Serif", 8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((System.Byte)(204)));

 this.label1.Location = new System.Drawing.Point(0, 0);

 this.label1.Name = "label1";

 this.label1.Size = new System.Drawing.Size(306, 48);

 this.label1.TabIndex = 0;

 this.label1.Text = "Приветствуем Вас в обучающей программе по информатике.";

 this.label1.TextAlign = System.Drawing.ContentAlignment.MiddleCenter;

 //

 // groupBox1

 //

 this.groupBox1.Controls.Add(this.hemming\_button);

 this.groupBox1.Controls.Add(this.post\_button);

 this.groupBox1.Controls.Add(this.even\_uneven\_button);

 this.groupBox1.Font = new System.Drawing.Font("MS Reference Sans Serif", 8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((System.Byte)(204)));

 this.groupBox1.Location = new System.Drawing.Point(8, 48);

 this.groupBox1.Name = "groupBox1";

 this.groupBox1.Size = new System.Drawing.Size(296, 120);

 this.groupBox1.TabIndex = 1;

 this.groupBox1.TabStop = false;

 this.groupBox1.Text = "Выберите раздел";

 //

 // hemming\_button

 //

 this.hemming\_button.Anchor = ((System.Windows.Forms.AnchorStyles)((((System.Windows.Forms.AnchorStyles.Top | System.Windows.Forms.AnchorStyles.Bottom)

 | System.Windows.Forms.AnchorStyles.Left)

 | System.Windows.Forms.AnchorStyles.Right)));

 this.hemming\_button.Cursor = System.Windows.Forms.Cursors.Hand;

 this.hemming\_button.Location = new System.Drawing.Point(8, 89);

 this.hemming\_button.Name = "hemming\_button";

 this.hemming\_button.Size = new System.Drawing.Size(280, 22);

 this.hemming\_button.TabIndex = 2;

 this.hemming\_button.Text = "Кодирование по методу Хэмминга";

 this.hemming\_button.Click += new System.EventHandler(this.hemming\_button\_Click);

 //

 // post\_button

 //

 this.post\_button.Anchor = ((System.Windows.Forms.AnchorStyles)((((System.Windows.Forms.AnchorStyles.Top | System.Windows.Forms.AnchorStyles.Bottom)

 | System.Windows.Forms.AnchorStyles.Left)

 | System.Windows.Forms.AnchorStyles.Right)));

 this.post\_button.Cursor = System.Windows.Forms.Cursors.Hand;

 this.post\_button.Location = new System.Drawing.Point(8, 25);

 this.post\_button.Name = "post\_button";

 this.post\_button.Size = new System.Drawing.Size(280, 22);

 this.post\_button.TabIndex = 0;

 this.post\_button.Text = "Программирование машины Поста";

 this.post\_button.Click += new System.EventHandler(this.post\_button\_Click);

 //

 // even\_uneven\_button

 //

 this.even\_uneven\_button.Anchor = ((System.Windows.Forms.AnchorStyles)((((System.Windows.Forms.AnchorStyles.Top | System.Windows.Forms.AnchorStyles.Bottom)

 | System.Windows.Forms.AnchorStyles.Left)

 | System.Windows.Forms.AnchorStyles.Right)));

 this.even\_uneven\_button.Cursor = System.Windows.Forms.Cursors.Hand;

 this.even\_uneven\_button.Location = new System.Drawing.Point(8, 56);

 this.even\_uneven\_button.Name = "even\_uneven\_button";

 this.even\_uneven\_button.Size = new System.Drawing.Size(280, 22);

 this.even\_uneven\_button.TabIndex = 1;

 this.even\_uneven\_button.Text = "Кодирование по методу четности\\нечетности";

 this.even\_uneven\_button.Click += new System.EventHandler(this.even\_uneven\_button\_Click);

 //

 // exit\_button

 //

 this.exit\_button.Cursor = System.Windows.Forms.Cursors.Hand;

 this.exit\_button.Font = new System.Drawing.Font("MS Reference Sans Serif", 8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((System.Byte)(204)));

 this.exit\_button.Location = new System.Drawing.Point(112, 176);

 this.exit\_button.Name = "exit\_button";

 this.exit\_button.Size = new System.Drawing.Size(72, 24);

 this.exit\_button.TabIndex = 2;

 this.exit\_button.Text = "Выход";

 this.exit\_button.Click += new System.EventHandler(this.exit\_button\_Click);

 //

 // Start

 //

 this.AutoScale = false;

 this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);

 this.BackColor = System.Drawing.SystemColors.InactiveBorder;

 this.ClientSize = new System.Drawing.Size(306, 206);

 this.ControlBox = false;

 this.Controls.Add(this.exit\_button);

 this.Controls.Add(this.groupBox1);

 this.Controls.Add(this.label1);

 this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedSingle;

 this.MaximizeBox = false;

 this.MinimizeBox = false;

 this.Name = "Start";

 this.ShowInTaskbar = false;

 this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;

 this.Text = "Study Programs";

 this.TopMost = true;

 this.groupBox1.ResumeLayout(false);

 this.ResumeLayout(false);

 }

 #endregion

 [STAThread]

 static void Main() {Application.Run(new Start());}

 private void exit\_button\_Click(object sender, System.EventArgs e){this.Close();}

 private void post\_button\_Click(object sender, System.EventArgs e)

 {

 A1 = new Post();

 A1.ShowDialog();

 }

 private void even\_uneven\_button\_Click(object sender, System.EventArgs e)

 {

 A2 = new Even\_uneven();

 A2.ShowDialog();

 }

 private void hemming\_button\_Click(object sender, System.EventArgs e)

 {

 A3 = new Hemming();

 A3.ShowDialog();

 }

 }

}

Post

using System;

using System.Drawing;

using System.Collections;

using System.ComponentModel;

using System.Windows.Forms;

namespace Study\_Programs

{

 public class Post : System.Windows.Forms.Form

 {

 private System.Windows.Forms.Label label1;

 private System.Windows.Forms.TextBox textBox1;

 private System.Windows.Forms.Label label2;

 private System.Windows.Forms.TextBox textBox2;

 private System.Windows.Forms.Label label3;

 private System.Windows.Forms.TextBox textBox3;

 private System.Windows.Forms.Button button2;

 private System.Windows.Forms.Button button3;

 private System.ComponentModel.Container components = null;

 private System.Windows.Forms.Button help;

 Help A4;Error A5;

 static char []input;

 static char [][] command;

 static byte l,finish,k;

 static sbyte position;

 static bool chk=true;

 public Post(){InitializeComponent();}

 protected override void Dispose(bool disposing)

 {

 if( disposing )if(components != null) components.Dispose();

 base.Dispose(disposing);

 }

 #region Windows Form Designer generated code

 private void InitializeComponent()

 {

 this.label1 = new System.Windows.Forms.Label();

 this.textBox1 = new System.Windows.Forms.TextBox();

 this.label2 = new System.Windows.Forms.Label();

 this.textBox2 = new System.Windows.Forms.TextBox();

 this.label3 = new System.Windows.Forms.Label();

 this.textBox3 = new System.Windows.Forms.TextBox();

 this.help = new System.Windows.Forms.Button();

 this.button2 = new System.Windows.Forms.Button();

 this.button3 = new System.Windows.Forms.Button();

 this.SuspendLayout();

 //

 // label1

 //

 this.label1.Font = new System.Drawing.Font("MS Reference Sans Serif", 8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((System.Byte)(204)));

 this.label1.Location = new System.Drawing.Point(8, 8);

 this.label1.Name = "label1";

 this.label1.Size = new System.Drawing.Size(192, 16);

 this.label1.TabIndex = 0;

 this.label1.Text = "Входная строка машины Поста";

 //

 // textBox1

 //

 this.textBox1.Font = new System.Drawing.Font("MS Reference Sans Serif", 8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((System.Byte)(204)));

 this.textBox1.Location = new System.Drawing.Point(8, 24);

 this.textBox1.Name = "textBox1";

 this.textBox1.Size = new System.Drawing.Size(224, 21);

 this.textBox1.TabIndex = 1;

 this.textBox1.Text = "Введите строку, состоящую из 0 и 1";

 //

 // label2

 //

 this.label2.Font = new System.Drawing.Font("MS Reference Sans Serif", 8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((System.Byte)(204)));

 this.label2.Location = new System.Drawing.Point(8, 56);

 this.label2.Name = "label2";

 this.label2.Size = new System.Drawing.Size(216, 16);

 this.label2.TabIndex = 2;

 this.label2.Text = "Список команд";

 //

 // textBox2

 //

 this.textBox2.Font = new System.Drawing.Font("MS Reference Sans Serif", 8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((System.Byte)(204)));

 this.textBox2.Location = new System.Drawing.Point(8, 72);

 this.textBox2.Name = "textBox2";

 this.textBox2.Size = new System.Drawing.Size(224, 21);

 this.textBox2.TabIndex = 3;

 this.textBox2.Text = "Введите список команд";

 //

 // label3

 //

 this.label3.Font = new System.Drawing.Font("MS Reference Sans Serif", 8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((System.Byte)(204)));

 this.label3.Location = new System.Drawing.Point(8, 104);

 this.label3.Name = "label3";

 this.label3.Size = new System.Drawing.Size(112, 16);

 this.label3.TabIndex = 4;

 this.label3.Text = "Выходная строка";

 //

 // textBox3

 //

 this.textBox3.BorderStyle = System.Windows.Forms.BorderStyle.None;

 this.textBox3.Font = new System.Drawing.Font("MS Reference Sans Serif", 8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((System.Byte)(204)));

 this.textBox3.Location = new System.Drawing.Point(8, 120);

 this.textBox3.Name = "textBox3";

 this.textBox3.ReadOnly = true;

 this.textBox3.Size = new System.Drawing.Size(224, 14);

 this.textBox3.TabIndex = 5;

 this.textBox3.Text = "";

 //

 // help

 //

 this.help.Font = new System.Drawing.Font("MS Reference Sans Serif", 8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((System.Byte)(204)));

 this.help.Location = new System.Drawing.Point(136, 144);

 this.help.Name = "help";

 this.help.Size = new System.Drawing.Size(72, 24);

 this.help.TabIndex = 8;

 this.help.Text = "Справка";

 this.help.Click += new System.EventHandler(this.help\_Click);

 //

 // button2

 //

 this.button2.Font = new System.Drawing.Font("MS Reference Sans Serif", 8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((System.Byte)(204)));

 this.button2.Location = new System.Drawing.Point(40, 176);

 this.button2.Name = "button2";

 this.button2.Size = new System.Drawing.Size(168, 24);

 this.button2.TabIndex = 9;

 this.button2.Text = "Вернуться к выбору темы";

 this.button2.Click += new System.EventHandler(this.button2\_Click);

 //

 // button3

 //

 this.button3.Font = new System.Drawing.Font("MS Reference Sans Serif", 8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((System.Byte)(204)));

 this.button3.Location = new System.Drawing.Point(40, 144);

 this.button3.Name = "button3";

 this.button3.Size = new System.Drawing.Size(72, 24);

 this.button3.TabIndex = 10;

 this.button3.Text = "GO";

 this.button3.Click += new System.EventHandler(this.button3\_Click);

 //

 // Post

 //

 this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);

 this.BackColor = System.Drawing.SystemColors.InactiveBorder;

 this.ClientSize = new System.Drawing.Size(242, 208);

 this.ControlBox = false;

 this.Controls.Add(this.button3);

 this.Controls.Add(this.button2);

 this.Controls.Add(this.help);

 this.Controls.Add(this.textBox3);

 this.Controls.Add(this.label3);

 this.Controls.Add(this.textBox2);

 this.Controls.Add(this.label2);

 this.Controls.Add(this.textBox1);

 this.Controls.Add(this.label1);

 this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedSingle;

 this.MaximizeBox = false;

 this.Name = "Post";

 this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;

 this.Text = "Post";

 this.TopMost = true;

 this.ResumeLayout(false);

 }

 #endregion

 public static byte first()

 {

 position++;

 return(k=Convert.ToByte(char.GetNumericValue(command[k][2])-1));

 }

 public static byte second()

 {

 position--;

 return(k=Convert.ToByte(char.GetNumericValue(command[k][2])-1));

 }

 public static byte third()

 {

 input[position]='1';

 return(k=Convert.ToByte(char.GetNumericValue(command[k][2])-1));

 }

 public static byte fourth()

 {

 input[position]='0';

 return(k=Convert.ToByte(char.GetNumericValue(command[k][2])-1));

 }

 public void fifth()

 {

 if(position<input.Length)

 {

 if (input[position]=='0') k=Convert.ToByte(char.GetNumericValue(command[k][2])-1);

 else k=Convert.ToByte(char.GetNumericValue(command[k][3])-1);

 }

 else

 {

 A5=new Error("Выход за границы входной строки");

 A5.ShowDialog();

 finish=1;chk=false;

 }

 }

 bool check\_string(string str)

 {

 int i,s=0,cnt=0;

 for (i=0;i<str.Length;i++) if(str[i]==';')cnt++;

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

 {

 if (!char.IsDigit(str[s])||str[s]=='0') return(false);

 s++;

 if (str[s]!=',') return(false);

 s++;

 if (char.GetNumericValue(str[s])>6||char.GetNumericValue(str[s])<1) return(false);

 s++;

 if (str[s]!=',') return(false);

 s++;

 if (!char.IsDigit(str[s])) return (false);

 s++;

 if (str[s]!=',') return(false);

 s++;

 if (!char.IsDigit(str[s])) return(false);

 s++;

 if (str[s]!=';') return(false);

 s++;

 }

 if(s!=str.Length) return(false);

 return(true);

 }

 private void button2\_Click(object sender, System.EventArgs e){this.Close();}

 private void help\_Click(object sender, System.EventArgs e)

 {

 A4 = new Help(0);

 A4.ShowDialog();

 }

 private void button3\_Click(object sender, System.EventArgs e)

 {

 l=1;k=0;position=0;finish=0;textBox3.Text="";chk=true;

 int i,j;

 //массив из входной строки

 input = new char [textBox1.Text.Length];

 command = new char [textBox2.Text.Length/8][];

 for (i=0;i<textBox1.Text.Length;i++)

 {

 if (textBox1.Text[i]!='0' && textBox1.Text[i]!='1')

 {

 A5 = new Error("Неправильный формат входной строки");

 A5.ShowDialog();

 i=textBox1.Text.Length;

 chk=false;

 }

 else input[i]=textBox1.Text[i];

 }

 //массив из списка комманд

 if (check\_string(textBox2.Text))

 {

 for (i=0;i<textBox2.Text.Length/8;i++)

 {

 command [i]=new char[4];

 for (j=0;j<4;j++) command[i][j]=textBox2.Text[i\*8+j\*2];

 }

 }

 else

 {

 A5 = new Error("Некорректный формат команды");

 A5.ShowDialog();

 chk=false;

 }

 if (chk)

 {

 do

 {

 switch(command[k][l])

 {

 case('1'):

 {

 first();

 break;

 }

 case('2'):

 {

 second();

 break;

 }

 case('3'):

 {

 third();

 break;

 }

 case('4'):

 {

 fourth();

 break;

 }

 case('5'):

 {

 fifth();

 break;

 }

 case('6'):

 {

 finish=1;

 break;

 }

 }

 }while (finish!=1);

 }

 if(chk) for(i=0;i<input.Length;i++) textBox3.Text+=input[i];

 }

 }

}

Hemming

using System;

using System.Drawing;

using System.Collections;

using System.ComponentModel;

using System.Windows.Forms;

namespace Study\_Programs

{

 public class Hemming : System.Windows.Forms.Form

 {

 int ChVoprosov=1;

 int X=0;

 int Nomer=0;

 int [] x=new int[10];

 Help A4; Error A5;

 private System.Windows.Forms.TextBox textBox1;

 private System.Windows.Forms.TextBox textBox2;

 private System.Windows.Forms.TextBox textBox3;

 private System.Windows.Forms.TextBox textBox4;

 private System.Windows.Forms.TextBox textBox5;

 private System.Windows.Forms.Button button1;

 private System.Windows.Forms.Button button2;

 private System.Windows.Forms.Button button3;

 private System.Windows.Forms.Button button4;

 private System.Windows.Forms.Button button5;

 private System.Windows.Forms.Label label3;

 private System.Windows.Forms.Label label4;

 private System.Windows.Forms.Label label1;

 private System.Windows.Forms.Label label2;

 private System.Windows.Forms.Label label5;

 private System.Windows.Forms.GroupBox groupBox1;

 private System.Windows.Forms.ToolTip toolTip1;

 private System.ComponentModel.IContainer components;

 public Hemming(){InitializeComponent();}

 protected override void Dispose( bool disposing )

 {

 if(disposing) if(components != null) components.Dispose();

 base.Dispose(disposing);

 }

 public void Generation()

 {

 X=1;

 textBox1.Text="";

 Random r=new System.Random();

 int d;

 for (int i=1; i<10; i++)

 {

 d=r.Next(0,2);

 textBox1.Text+=d;

 x[i]=d;

 }

 }

 public void Number()

 {

 int x1=0;int x2=0;int x3=0;int x4=0;

 x1=x[1]+x[3]+x[5]+x[7]+x[9];

 if (x1==0 || x1==2 || x1==4) x1=0; else x1=1;

 x2=x[2]+x[3]+x[6]+x[7];

 if (x2==0 || x2==2 || x2==4) x2=0; else x2=1;

 x3=x[4]+x[5]+x[6]+x[7];

 if (x3==0 || x3==2 || x3==4) x3=0; else x3=1;

 x4=x[8]+x[9];

 if (x4==0 || x4==2) x4=0;else x4=1;

 Nomer=x4\*1000+x3\*100+x2\*10+x1;

 }

 public void Otvet(){textBox3.Text = x[1] + "" + x[2] + "" + x[3] + "" + x[4] + "" + x[5] + "" + x[6] + "" + x[7] + "" + x[8] + "" + x[9];}

 public void Sravnenie()

 {

 if (X==1)

 {

 int i=0;

 if (Nomer == 1) i = 1;

 if (Nomer == 10) i = 2;

 if (Nomer == 11) i = 3;

 if (Nomer == 100) i = 4;

 if (Nomer == 101) i = 5;

 if (Nomer == 110) i = 6;

 if (Nomer == 111) i = 7;

 if (Nomer == 1000) i = 8;

 if (Nomer == 1001) i = 9;

 if (i!=0)

 {

 if (x[i] == 0)

 {

 x[i] = 1; Otvet();

 }

 else

 {

 x[i] = 0; Otvet();

 }

 }

 }

 }

 public void Proverka()

 {

 if(textBox2.Text.Length!=textBox1.Text.Length && textBox1.Text.Length!=textBox3.Text.Length)

 {

 A5=new Error("Исправленный вариант должен содержать 9-ти разрядное двоичное число");

 A5.ShowDialog();

 X=0;

 }

 else

 {

 if(textBox1.Text.Length!=textBox3.Text.Length)

 {

 for(int i=0; i<textBox2.Text.Length; i++)

 {

 string a="1";

 string b="0";

 string d=textBox2.Text.Substring(i,1);

 if(d!=b && d!=a)

 {

 A5=new Error("Исправленный вариант должен содержать 9-ти разрядное двоичное число");

 A5.ShowDialog();

 X=0;

 i=textBox2.Text.Length;

 }

 else X=1;

 }

 }

 }

 }

 #region Windows Form Designer generated code

 private void InitializeComponent()

 {

 this.components = new System.ComponentModel.Container();

 System.ComponentModel.ComponentResourceManager resources = new System.ComponentModel.ComponentResourceManager(typeof(Hemming));

 this.textBox1 = new System.Windows.Forms.TextBox();

 this.textBox2 = new System.Windows.Forms.TextBox();

 this.textBox3 = new System.Windows.Forms.TextBox();

 this.textBox4 = new System.Windows.Forms.TextBox();

 this.textBox5 = new System.Windows.Forms.TextBox();

 this.button1 = new System.Windows.Forms.Button();

 this.button2 = new System.Windows.Forms.Button();

 this.button3 = new System.Windows.Forms.Button();

 this.button4 = new System.Windows.Forms.Button();

 this.button5 = new System.Windows.Forms.Button();

 this.label3 = new System.Windows.Forms.Label();

 this.label4 = new System.Windows.Forms.Label();

 this.label1 = new System.Windows.Forms.Label();

 this.label2 = new System.Windows.Forms.Label();

 this.label5 = new System.Windows.Forms.Label();

 this.groupBox1 = new System.Windows.Forms.GroupBox();

 this.toolTip1 = new System.Windows.Forms.ToolTip(this.components);

 this.groupBox1.SuspendLayout();

 this.SuspendLayout();

 //

 // textBox1

 //

 this.textBox1.Location = new System.Drawing.Point(216, 16);

 this.textBox1.Name = "textBox1";

 this.textBox1.ReadOnly = true;

 this.textBox1.Size = new System.Drawing.Size(88, 20);

 this.textBox1.TabIndex = 0;

 this.textBox1.TextAlign = System.Windows.Forms.HorizontalAlignment.Center;

 //

 // textBox2

 //

 this.textBox2.Location = new System.Drawing.Point(216, 40);

 this.textBox2.MaxLength = 9;

 this.textBox2.Name = "textBox2";

 this.textBox2.Size = new System.Drawing.Size(88, 20);

 this.textBox2.TabIndex = 1;

 this.textBox2.TextAlign = System.Windows.Forms.HorizontalAlignment.Center;

 //

 // textBox3

 //

 this.textBox3.Location = new System.Drawing.Point(216, 64);

 this.textBox3.Name = "textBox3";

 this.textBox3.ReadOnly = true;

 this.textBox3.Size = new System.Drawing.Size(88, 20);

 this.textBox3.TabIndex = 2;

 this.textBox3.TextAlign = System.Windows.Forms.HorizontalAlignment.Center;

 //

 // textBox4

 //

 this.textBox4.Location = new System.Drawing.Point(104, 136);

 this.textBox4.Name = "textBox4";

 this.textBox4.ReadOnly = true;

 this.textBox4.Size = new System.Drawing.Size(40, 20);

 this.textBox4.TabIndex = 3;

 this.textBox4.Text = "1";

 this.textBox4.TextAlign = System.Windows.Forms.HorizontalAlignment.Center;

 //

 // textBox5

 //

 this.textBox5.Location = new System.Drawing.Point(272, 136);

 this.textBox5.Name = "textBox5";

 this.textBox5.ReadOnly = true;

 this.textBox5.Size = new System.Drawing.Size(40, 20);

 this.textBox5.TabIndex = 4;

 this.textBox5.Text = "0";

 this.textBox5.TextAlign = System.Windows.Forms.HorizontalAlignment.Center;

 //

 // button1

 //

 this.button1.Location = new System.Drawing.Point(8, 102);

 this.button1.Name = "button1";

 this.button1.Size = new System.Drawing.Size(88, 24);

 this.button1.TabIndex = 5;

 this.button1.Text = "Ошибки нет";

 this.button1.Click += new System.EventHandler(this.button1\_Click);

 //

 // button2

 //

 this.button2.Location = new System.Drawing.Point(108, 102);

 this.button2.Name = "button2";

 this.button2.Size = new System.Drawing.Size(88, 24);

 this.button2.TabIndex = 6;

 this.button2.Text = "Ошибка есть";

 this.button2.Click += new System.EventHandler(this.button2\_Click);

 //

 // button3

 //

 this.button3.Location = new System.Drawing.Point(206, 101);

 this.button3.Name = "button3";

 this.button3.Size = new System.Drawing.Size(112, 24);

 this.button3.TabIndex = 7;

 this.button3.Text = "Следующее число";

 this.button3.Click += new System.EventHandler(this.button3\_Click);

 //

 // button4

 //

 this.button4.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

 this.button4.Location = new System.Drawing.Point(128, 160);

 this.button4.Name = "button4";

 this.button4.Size = new System.Drawing.Size(72, 24);

 this.button4.TabIndex = 11;

 this.button4.Text = "Справка";

 this.button4.Click += new System.EventHandler(this.button4\_Click);

 //

 // button5

 //

 this.button5.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

 this.button5.Location = new System.Drawing.Point(80, 192);

 this.button5.Name = "button5";

 this.button5.Size = new System.Drawing.Size(168, 24);

 this.button5.TabIndex = 12;

 this.button5.Text = "Вернуться к выбору темы";

 this.button5.Click += new System.EventHandler(this.button5\_Click);

 //

 // label3

 //

 this.label3.Location = new System.Drawing.Point(8, 136);

 this.label3.Name = "label3";

 this.label3.Size = new System.Drawing.Size(96, 24);

 this.label3.TabIndex = 13;

 this.label3.Text = "Номер вопроса:";

 //

 // label4

 //

 this.label4.Location = new System.Drawing.Point(152, 136);

 this.label4.Name = "label4";

 this.label4.Size = new System.Drawing.Size(120, 24);

 this.label4.TabIndex = 14;

 this.label4.Text = "Правильных ответов:";

 //

 // label1

 //

 this.label1.Location = new System.Drawing.Point(8, 16);

 this.label1.Name = "label1";

 this.label1.Size = new System.Drawing.Size(216, 16);

 this.label1.TabIndex = 15;

 this.label1.Text = "Число, возможно содержащее ошибку";

 //

 // label2

 //

 this.label2.Location = new System.Drawing.Point(8, 40);

 this.label2.Name = "label2";

 this.label2.Size = new System.Drawing.Size(128, 16);

 this.label2.TabIndex = 16;

 this.label2.Text = "Исправленный вариант";

 //

 // label5

 //

 this.label5.Location = new System.Drawing.Point(8, 64);

 this.label5.Name = "label5";

 this.label5.Size = new System.Drawing.Size(112, 16);

 this.label5.TabIndex = 17;

 this.label5.Text = "Правильный ответ";

 //

 // groupBox1

 //

 this.groupBox1.Controls.Add(this.label2);

 this.groupBox1.Controls.Add(this.textBox3);

 this.groupBox1.Controls.Add(this.label5);

 this.groupBox1.Controls.Add(this.textBox1);

 this.groupBox1.Controls.Add(this.textBox2);

 this.groupBox1.Controls.Add(this.label1);

 this.groupBox1.Location = new System.Drawing.Point(6, -1);

 this.groupBox1.Name = "groupBox1";

 this.groupBox1.Size = new System.Drawing.Size(312, 96);

 this.groupBox1.TabIndex = 19;

 this.groupBox1.TabStop = false;

 this.toolTip1.SetToolTip(this.groupBox1, @"9-ти разрядное двоичное число, закодированно по методу Хэмминга. В одном из разрядов"+"\n"+"возможно присутствует ошибка. Если она есть, необходимо её исправить и записать правильный"+"\n"+"вариант в соответствующее окно, после этого нажмите кнопку «Ошибка есть».Если ошибка"+"\n"+"отсутствует, то в окно Исправленный вариант ничего писать не нужно и следуетнажать"+"\n"+"кнопку «Ошибки нет». Далее нужно нажать кнопку «Cледующее число»");

 //

 // toolTip1

 //

 this.toolTip1.AutomaticDelay = 10;

 this.toolTip1.AutoPopDelay = 100000;

 this.toolTip1.InitialDelay = 10;

 this.toolTip1.ReshowDelay = 2;

 this.toolTip1.ShowAlways = true;

 //

 // Hemming

 //

 this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);

 this.BackColor = System.Drawing.SystemColors.InactiveBorder;

 this.ClientSize = new System.Drawing.Size(322, 224);

 this.ControlBox = false;

 this.Controls.Add(this.groupBox1);

 this.Controls.Add(this.label4);

 this.Controls.Add(this.label3);

 this.Controls.Add(this.button5);

 this.Controls.Add(this.button4);

 this.Controls.Add(this.button3);

 this.Controls.Add(this.button2);

 this.Controls.Add(this.button1);

 this.Controls.Add(this.textBox5);

 this.Controls.Add(this.textBox4);

 this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedSingle;

 this.MaximizeBox = false;

 this.Name = "Hemming";

 this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;

 this.Text = "Hemming";

 this.TopMost = true;

 this.Load += new System.EventHandler(this.Hemming\_Load);

 this.groupBox1.ResumeLayout(false);

 this.groupBox1.PerformLayout();

 this.ResumeLayout(false);

 this.PerformLayout();

 }

 #endregion

 private void Hemming\_Load(object sender, System.EventArgs e)

 {

 Generation();

 Number();

 }

 private void button1\_Click(object sender, System.EventArgs e)

 {

 textBox2.Text="";

 if(textBox1.Text.Length!=textBox3.Text.Length) X=1;

 if (X==1)textBox3.Text="";

 int Prav=Convert.ToInt32(textBox5.Text);

 if (X==1)

 {

 if (Nomer==0)

 {

 Prav++; Otvet();

 }

 Sravnenie();

 if (textBox3.Text.Length==0)

 {

 Prav++; Otvet();

 }

 textBox5.Text=""; textBox5.Text+=Prav;

 }

 X=0;

 }

 private void button2\_Click(object sender, System.EventArgs e)

 {

 Proverka();

 if(X==1)textBox3.Text="";

 int Prav=Convert.ToInt32(textBox5.Text);

 if (X==1)

 {

 if (Nomer==0) Otvet();

 Sravnenie();

 if (textBox3.Text.Length==0) Otvet();

 int Moe=Convert.ToInt32(textBox2.Text);

 int Pravilnoe=Convert.ToInt32(textBox3.Text);

 if (Moe==Pravilnoe) Prav++;

 textBox5.Text="";

 textBox5.Text+=Prav;

 }

 X=0;

 }

 private void button4\_Click(object sender, System.EventArgs e)

 {

 A4 = new Help(2);

 A4.ShowDialog();

 }

 private void button5\_Click(object sender, System.EventArgs e){this.Close();}

 private void button3\_Click(object sender, System.EventArgs e)

 {

 ChVoprosov++;

 textBox2.Text="";

 textBox3.Text="";

 textBox4.Text="";

 textBox4.Text+=ChVoprosov;

 Generation();

 Number();

 }

 }

}

Even\_ueven

using System;

using System.Drawing;

using System.Collections;

using System.ComponentModel;

using System.Windows.Forms;

namespace Study\_Programs

{

 public class Even\_uneven : System.Windows.Forms.Form

 {

 int Prav;

 int ch;

 int ChVoprosov=1;

 Help A4;

 private System.Windows.Forms.TextBox textBox1;

 private System.Windows.Forms.Button button1;

 private System.Windows.Forms.TextBox textBox2;

 private System.Windows.Forms.Button button2;

 private System.Windows.Forms.TextBox textBox3;

 private System.Windows.Forms.TextBox textBox4;

 private System.Windows.Forms.Label label1;

 private System.Windows.Forms.Label label2;

 private System.Windows.Forms.Label label3;

 private System.Windows.Forms.Label label4;

 private System.Windows.Forms.Button button3;

 private System.Windows.Forms.Button button4;

 private System.Windows.Forms.Label label6;

 private System.Windows.Forms.GroupBox groupBox1;

 private System.ComponentModel.Container components = null;

 public Even\_uneven(){InitializeComponent();}

 protected override void Dispose( bool disposing )

 {

 if( disposing ) if(components != null) components.Dispose();

 base.Dispose( disposing );

 }

 public void Generation()

 {

 textBox1.Text="";

 Random r=new System.Random();

 int d;

 for (int i=0; i<7; i++)

 {

 d=r.Next(0,2);

 textBox1.Text+=d;

 ch+=d;

 }

 textBox2.Text="";

 int g=r.Next(0,2);

 textBox2.Text+=g;

 ch+=g;

 }

 public void ChangeChVoprosov()

 {

 ChVoprosov++;

 textBox3.Text="";

 textBox3.Text+=ChVoprosov;

 }

 public void ChangePrav()

 {

 textBox4.Text="";

 textBox4.Text+=Prav;

 }

 #region Windows Form Designer generated code

 private void InitializeComponent()

 {

 this.textBox1 = new System.Windows.Forms.TextBox();

 this.button1 = new System.Windows.Forms.Button();

 this.textBox2 = new System.Windows.Forms.TextBox();

 this.button2 = new System.Windows.Forms.Button();

 this.textBox3 = new System.Windows.Forms.TextBox();

 this.textBox4 = new System.Windows.Forms.TextBox();

 this.label1 = new System.Windows.Forms.Label();

 this.label2 = new System.Windows.Forms.Label();

 this.label3 = new System.Windows.Forms.Label();

 this.label4 = new System.Windows.Forms.Label();

 this.button3 = new System.Windows.Forms.Button();

 this.button4 = new System.Windows.Forms.Button();

 this.label6 = new System.Windows.Forms.Label();

 this.groupBox1 = new System.Windows.Forms.GroupBox();

 this.groupBox1.SuspendLayout();

 this.SuspendLayout();

 //

 // textBox1

 //

 this.textBox1.Location = new System.Drawing.Point(192, 15);

 this.textBox1.Name = "textBox1";

 this.textBox1.ReadOnly = true;

 this.textBox1.Size = new System.Drawing.Size(88, 20);

 this.textBox1.TabIndex = 0;

 this.textBox1.Text = "";

 this.textBox1.TextAlign = System.Windows.Forms.HorizontalAlignment.Center;

 //

 // button1

 //

 this.button1.Location = new System.Drawing.Point(7, 156);

 this.button1.Name = "button1";

 this.button1.Size = new System.Drawing.Size(140, 22);

 this.button1.TabIndex = 1;

 this.button1.Text = "Код числа правильный";

 this.button1.Click += new System.EventHandler(this.button1\_Click);

 //

 // textBox2

 //

 this.textBox2.Location = new System.Drawing.Point(192, 45);

 this.textBox2.Name = "textBox2";

 this.textBox2.ReadOnly = true;

 this.textBox2.Size = new System.Drawing.Size(88, 20);

 this.textBox2.TabIndex = 2;

 this.textBox2.Text = "0";

 this.textBox2.TextAlign = System.Windows.Forms.HorizontalAlignment.Center;

 //

 // button2

 //

 this.button2.Location = new System.Drawing.Point(153, 156);

 this.button2.Name = "button2";

 this.button2.Size = new System.Drawing.Size(140, 21);

 this.button2.TabIndex = 3;

 this.button2.Text = "Код числа неправильный";

 this.button2.Click += new System.EventHandler(this.button2\_Click);

 //

 // textBox3

 //

 this.textBox3.Location = new System.Drawing.Point(103, 186);

 this.textBox3.Name = "textBox3";

 this.textBox3.ReadOnly = true;

 this.textBox3.Size = new System.Drawing.Size(34, 20);

 this.textBox3.TabIndex = 4;

 this.textBox3.Text = "1";

 this.textBox3.TextAlign = System.Windows.Forms.HorizontalAlignment.Center;

 //

 // textBox4

 //

 this.textBox4.Location = new System.Drawing.Point(260, 186);

 this.textBox4.Name = "textBox4";

 this.textBox4.ReadOnly = true;

 this.textBox4.Size = new System.Drawing.Size(33, 20);

 this.textBox4.TabIndex = 5;

 this.textBox4.Text = "0";

 this.textBox4.TextAlign = System.Windows.Forms.HorizontalAlignment.Center;

 //

 // label1

 //

 this.label1.Location = new System.Drawing.Point(7, 16);

 this.label1.Name = "label1";

 this.label1.Size = new System.Drawing.Size(113, 22);

 this.label1.TabIndex = 6;

 this.label1.Text = "Цифровые разряды:";

 //

 // label2

 //

 this.label2.Location = new System.Drawing.Point(7, 45);

 this.label2.Name = "label2";

 this.label2.Size = new System.Drawing.Size(120, 22);

 this.label2.TabIndex = 7;

 this.label2.Text = "Контрольный разряд:";

 //

 // label3

 //

 this.label3.Location = new System.Drawing.Point(12, 189);

 this.label3.Name = "label3";

 this.label3.Size = new System.Drawing.Size(86, 15);

 this.label3.TabIndex = 8;

 this.label3.Text = "Номер вопроса:";

 //

 // label4

 //

 this.label4.Location = new System.Drawing.Point(146, 186);

 this.label4.Name = "label4";

 this.label4.Size = new System.Drawing.Size(114, 14);

 this.label4.TabIndex = 9;

 this.label4.Text = "Правильных ответов:";

 //

 // button3

 //

 this.button3.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((System.Byte)(204)));

 this.button3.Location = new System.Drawing.Point(120, 208);

 this.button3.Name = "button3";

 this.button3.Size = new System.Drawing.Size(60, 22);

 this.button3.TabIndex = 10;

 this.button3.Text = "Справка";

 this.button3.Click += new System.EventHandler(this.button3\_Click);

 //

 // button4

 //

 this.button4.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((System.Byte)(204)));

 this.button4.Location = new System.Drawing.Point(80, 238);

 this.button4.Name = "button4";

 this.button4.Size = new System.Drawing.Size(140, 22);

 this.button4.TabIndex = 11;

 this.button4.Text = "Вернуться к выбору темы";

 this.button4.Click += new System.EventHandler(this.button4\_Click);

 //

 // label6

 //

 this.label6.Location = new System.Drawing.Point(8, 88);

 this.label6.Name = "label6";

 this.label6.Size = new System.Drawing.Size(288, 64);

 this.label6.TabIndex = 13;

 this.label6.Text = "Нужно определить присутствует или не присутствует ошибка в записи числа.Если числ" +

 "о записано правильно, нужно щелкнуть по кнопке «Код числа правильный». Если чис" +

 "ло содержит ошибку, нужно щелкнуть кнопку «Код числа неправильный». ";

 this.label6.TextAlign = System.Drawing.ContentAlignment.BottomLeft;

 //

 // groupBox1

 //

 this.groupBox1.Controls.Add(this.label1);

 this.groupBox1.Controls.Add(this.textBox1);

 this.groupBox1.Controls.Add(this.label2);

 this.groupBox1.Controls.Add(this.textBox2);

 this.groupBox1.Location = new System.Drawing.Point(7, 7);

 this.groupBox1.Name = "groupBox1";

 this.groupBox1.Size = new System.Drawing.Size(286, 75);

 this.groupBox1.TabIndex = 14;

 this.groupBox1.TabStop = false;

 this.groupBox1.Text = "Задано случайное число:";

 //

 // Even\_uneven

 //

 this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);

 this.BackColor = System.Drawing.SystemColors.InactiveBorder;

 this.ClientSize = new System.Drawing.Size(301, 272);

 this.ControlBox = false;

 this.Controls.Add(this.groupBox1);

 this.Controls.Add(this.label6);

 this.Controls.Add(this.button4);

 this.Controls.Add(this.button3);

 this.Controls.Add(this.label4);

 this.Controls.Add(this.label3);

 this.Controls.Add(this.textBox4);

 this.Controls.Add(this.textBox3);

 this.Controls.Add(this.button2);

 this.Controls.Add(this.button1);

 this.Font = new System.Drawing.Font("Microsoft Sans Serif", 8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((System.Byte)(204)));

 this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedSingle;

 this.Name = "Even\_uneven";

 this.ShowInTaskbar = false;

 this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;

 this.Text = "Even\_uneven";

 this.TopMost = true;

 this.Load += new System.EventHandler(this.Form1\_Load);

 this.groupBox1.ResumeLayout(false);

 this.ResumeLayout(false);

 }

 #endregion

 private void Form1\_Load(object sender, System.EventArgs e)

 {

 Generation();

 }

 private void button1\_Click(object sender, System.EventArgs e)

 {

 ChangeChVoprosov();

 Prav=Convert.ToInt32(textBox4.Text);

 if (ch==0 || ch==2 || ch==4 || ch==6) Prav++;

 ChangePrav();

 ch=0;

 Generation();

 }

 private void button2\_Click(object sender, System.EventArgs e)

 {

 ChangeChVoprosov();

 Prav=Convert.ToInt32(textBox4.Text);

 if (ch==1 || ch==3 || ch==5 || ch==7) Prav++;

 ChangePrav();

 ch=0;

 Generation();

 }

 private void button4\_Click(object sender, System.EventArgs e){this.Close();}

 private void button3\_Click(object sender, System.EventArgs e)

 {

 A4 = new Help(1); A4.ShowDialog();

 }

 }

}

Help

using System;

using System.Drawing;

using System.Collections;

using System.ComponentModel;

using System.Windows.Forms;

namespace Study\_Programs

{

 public class Help : System.Windows.Forms.Form

 {

 private System.Windows.Forms.Label label1;

 private System.Windows.Forms.ComboBox comboBox1;

 private System.Windows.Forms.RichTextBox richTextBox1;

 private System.Windows.Forms.Button button1;

 private System.ComponentModel.Container components = null;

 public Help(byte n)

 {

 InitializeComponent();

 comboBox1.SelectedIndex=n;

 }

 protected override void Dispose( bool disposing )

 {

 if(disposing) if(components != null)components.Dispose();

 base.Dispose( disposing );

 }

 #region Windows Form Designer generated code

 private void InitializeComponent()

 {

 this.label1 = new System.Windows.Forms.Label();

 this.comboBox1 = new System.Windows.Forms.ComboBox();

 this.richTextBox1 = new System.Windows.Forms.RichTextBox();

 this.button1 = new System.Windows.Forms.Button();

 this.SuspendLayout();

 //

 // label1

 //

 this.label1.Font = new System.Drawing.Font("MS Reference Sans Serif", 8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((System.Byte)(204)));

 this.label1.Location = new System.Drawing.Point(56, 0);

 this.label1.Name = "label1";

 this.label1.Size = new System.Drawing.Size(280, 24);

 this.label1.TabIndex = 0;

 this.label1.Text = "Выберите интересующий Вас раздел";

 this.label1.TextAlign = System.Drawing.ContentAlignment.MiddleCenter;

 //

 // comboBox1

 //

 this.comboBox1.DropDownStyle = System.Windows.Forms.ComboBoxStyle.DropDownList;

 this.comboBox1.Font = new System.Drawing.Font("MS Reference Sans Serif", 8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((System.Byte)(204)));

 this.comboBox1.Items.AddRange(new object[] {

 "Программирование машины Поста",

 "Кодирование по методу четности\\нечетности",

 "Кодирование по методу Хэмминга"});

 this.comboBox1.Location = new System.Drawing.Point(8, 24);

 this.comboBox1.Name = "comboBox1";

 this.comboBox1.Size = new System.Drawing.Size(376, 23);

 this.comboBox1.TabIndex = 1;

 this.comboBox1.SelectedIndexChanged += new System.EventHandler(this.comboBox1\_SelectedIndexChanged);

 //

 // richTextBox1

 //

 this.richTextBox1.Font = new System.Drawing.Font("MS Reference Sans Serif", 8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((System.Byte)(204)));

 this.richTextBox1.Location = new System.Drawing.Point(8, 56);

 this.richTextBox1.Name = "richTextBox1";

 this.richTextBox1.ReadOnly = true;

 this.richTextBox1.Size = new System.Drawing.Size(376, 184);

 this.richTextBox1.TabIndex = 2;

 this.richTextBox1.Text = "";

 //

 // button1

 //

 this.button1.Font = new System.Drawing.Font("MS Reference Sans Serif", 8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((System.Byte)(204)));

 this.button1.Location = new System.Drawing.Point(160, 248);

 this.button1.Name = "button1";

 this.button1.Size = new System.Drawing.Size(80, 24);

 this.button1.TabIndex = 3;

 this.button1.Text = "Выход";

 this.button1.Click += new System.EventHandler(this.button1\_Click);

 //

 // Help

 //

 this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);

 this.BackColor = System.Drawing.SystemColors.InactiveBorder;

 this.ClientSize = new System.Drawing.Size(394, 280);

 this.ControlBox = false;

 this.Controls.Add(this.button1);

 this.Controls.Add(this.richTextBox1);

 this.Controls.Add(this.comboBox1);

 this.Controls.Add(this.label1);

 this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedSingle;

 this.MaximizeBox = false;

 this.Name = "Help";

 this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;

 this.Text = "Help";

 this.TopMost = true;

 this.ResumeLayout(false);

 }

 #endregion

 private void button1\_Click(object sender, System.EventArgs e){this.Close();}

 private void comboBox1\_SelectedIndexChanged(object sender, System.EventArgs e)

 {

 string temp;

 switch(comboBox1.SelectedIndex)

 {

 case (0):

 {

 richTextBox1.Text="";

 temp="Машина Поста (как и машина Тьюринга) является одним из способов формализации понятия алгоритма. Машина Поста является абстрактной машиной, предназначенной для доказательства различных утверждений о свойствах программ для них. Машина Поста является полностью детерминированной машиной, читающей начальные данные и, после выполнения программы, выдающей результат. На содержательном уровне машина Поста представляет собой бесконечную ленту, разделенную на одинаковые клетки, каждая из которых может быть либо пустой либо заполненной меткой, вдоль которой может перемещаться головка. За один такт или за одну команду головка может переместиться на одну клетку вправо или влево (или остаться на месте). Кроме того головка может записать в клетку, над которой она находится, метку (если ее там не было) или стереть метку (если она там была). В результате работы машина Поста перерабатывает содержимое ленты в соответствие с заданной программой.В общем виде отдельная команда машины Поста имеет следующий вид: n,K,m1,m2; где n – порядковый номер команды, К – тип команды, m1, m2 – порядковые номера следующих команды. Имеется шесть типов команд машины Поста, которым для удобства можно дать номера от 1 до 6. Ниже дано описание команд машины Поста.\n1 - Движение головки на одну клетку вправо и переход к команде m1.\n2 - Движение головки на одну клетку влево и переход к команде m1.\n3 - Нанесение метки в ячейку и переход к команде m1.\n4 - Стирание метки в ячейке и переход к команде m1.\n5 - Проверка наличия метки в клетке, над которой находится головка. Если метка отсутствует, то управление передается с номером m1, если метка есть, то с номером m2.\n6 - Остановка машины.\nПример команды машины Поста: 1,1,2,0;. Данная команда имеет порядковый номер 1, тип команды – 1 (движение головки на одну клетку вправо). После выполнения команды управление передается команде, имеющей порядковый номер 2.\nРассмотрим пример простой программы, которая стирает первую метку, записанную на ленте. Предположим, ";

richTextBox1.Text = temp + "что в начале работы головка находится над крайне левой ячейкой ленты. Поставленная задача может быть выполнена с помощью следующего набора команд.\n• 1,5,2,4; 1-я команда.\n• 2,1,3,0; 2-я команда.\n• 3,5,2,4; 3-я команда.\n• 4,4,5,0; 4-я команда.\n• 5,6,0,0; 5-я команда.\nРазберем работу программы.\n1-я команда – это команда типа 5, т.е. условный переход. Если ячейка пустая, то управление передается команде 2, в противном случае – 4-й команде.\n2-я команда – это сдвиг головки на одну ячейку вправо и передача управления 3-й команде.\n3-я команда – это опять условный переход. Если метка отсутствует, то возврат ко второй команде и переход на 4-ю в противном случае.\n4-я команда стирает метку (очевидно, она там есть, т.к. мы попадаем на 4-ю команду только при наличии метки) и передает управление 5-й команде, которая является командой остановки машины.\nПорядок записи и выполнения программы следующий:\n1.В окно «Входная строка машины Поста» введите произвольную строку, состоящую из нулей и единиц, для которой поставленная задача имеет смысл. Например, если задача состоит в стирании 1-й метки, то, чтобы проверить правильность работы программы, во входной строке должна присутствовать хотя бы одна метка, т.е. единица.\n2.В нижележащее окно введите в соответствии с выше перечисленными правилами набор команд, решающих поставленную задачу.\n3.Нажмите кнопку «GO», после чего записанные команды будут выполнены и в окне «Выходная строка» появится результат работы машины Поста в виде выходной строки символов, состоящей из нулей и единиц.\nВ качестве упражнения задайте произвольную входную строку и введите рассмотренную выше программу, стирающую 1-ю метку. Запустите программу на выполнение. Результат работы программы покажите преподавателю.";

 break;

 }

 case (1):

 {

 richTextBox1.Text="";

 temp="Понятие двоичного кода.\nДля контроля правильности передачи информации, а также как средство шифрования информации используются различные коды. Коды, использующие для передачи информации два различных элементарных сигнала, называются двоичными. Эти сигналы удобно обозначать символами 0 и 1. Тогда кодовое слово будет состоять из последовательности нулей и единиц.\nПри выполнении арифметических операций в цифровом автомате правильный результат будет получен только в случае, если машина работает без нарушений. При возникновении какой-либо ошибки в двоичном коде результат операции будет неверным, однако пользователь об этом не узнает, если не будут предусмотрены меры, сигнализирующие о появлении ошибки. То есть должна быть разработана определенная система контроля работы цифрового автомата. Для этой цели и служат различные методы кодирования двоичной информации. Кодирование по методу четности-нечетности. В случае кодирования по методу четности-нечетности к каждому двоичному числу добавляется один избыточный разряд и в него записывается 1 или 0 с таким условием, чтобы сумма цифр в числе была по модулю 2 равна 0 в случае кодирования по методу четности и 1 – по методу нечетности. Другими словами число единиц в двоичном числе (вместе с контрольным разрядом) должно быть четным числом в случае четности и нечетным – в случае нечетности. Появление ошибки в двоичном коде обнаружится по нарушению этого правила. Например, число 10011011 будет ошибочным в случае кодирования по четности, так как число единиц в числе нечетно и равно 5. При таком кодировании допускается, что может возникнуть только одна ошибка. Очевидно, что если число содержит две ошибки (или вообще четное число ошибок) то его код по методу четности будет правильным. То же самое можно сказать и о кодировании по методу нечетности. Пример реализации метода четности:\nI.Цифровой разряд - 1011101\nКонтрольный разряд - 1\nПризнак правильности числа - 0\n\nII.Цифровой разряд - 1110100\nКонтрольный разряд - 0\nПризнак правильности числа - 0\n\nIII.Ц";

richTextBox1.Text = temp + "ифровой разряд - 1000011\nКонтрольный разряд - 0\nПризнак правильности числа - 1\n\nIV.Цифровой разряд - 1110111\nКонтрольный разряд - 1\nПризнак правильности числа - 1.\nПризнак правильности равный 0 означает, что код числа правильный и 1 - неправильный. Такое кодирование не позволяет в случае появления ошибки определить какой именно разряд является ошибочным. Избыточность информации при таком кодировании минимальна и равна одному биту (один дополнительный разряд). Можно представить видоизмененный метод контроля по методу четности. Цифровые разряды двоичного числа разбиваются на группы, каждая из которых содержит одинаковое количество разрядов. Контрольные разряды добавляются к каждой группе по строкам и по столбцам. Увеличение избыточности информации (вместо одного дополнительного разряда для 9 разрядного числа мы имеем 6 дополнительных разрядов) приводит к тому, что появляется возможность не только обнаружить наличие ошибки, но и место ее возникновения.";

 break;

 }

 case (2):

 {

 richTextBox1.Text="";

 temp="Как и в последнем варианте метода четности, коды Хэмминга позволяют не только обнаружить наличие ошибки, но и место ее нахождения и следовательно дают возможность ее исправить. Однако, в отличие от предыдущего методы, коды Хэмминга обладают меньшей избыточностью, т.е. количеством дополнительных контрольных разрядов. Предположим, что имеется код, содержащий m информационных и k контрольных разрядов. Все разряды, включая контрольные, разбиваются на k групп по определенным правилам, о которых будет сказано ниже. Каждая группа, содержащая один контрольный разряд, проверяется на четность. Пусть были проведены все k проверок. Если результат данной проверки свидетельствует об отсутствии ошибки, то записывается 0, если есть ошибка, то записывается 1. В результате получается последовательность, состоящая из k нулей и единиц. При отсутствии ошибки в коде получается последовательность нулей. Полученное k-разрядное двоичное число может содержать 2k различных комбинаций нулей и единиц. С помощью этой информации нужно определить ошибочный разряд в коде, содержащем m+k разрядов. Для того, чтобы это было возможно должно выполняться неравенство:2^k=>(m+k+1)\nОпределить максимальное значение m для данного k можно из следующей таблицы.\n n|1,2,3,4…|8,…,15|16,…,31|…|\nm|0,0,1,1…|4,…,11|11,…,26|…|\nk | 1,2,2,3 | 4,…,4 | 5,…,5 |…|\nИз таблицы видно, для 16-ти разрядного числа требуется 5 контрольных разрядов. В качестве сравнения, в случае модифицированного метода четности потребовалось бы 8 контрольных разрядов. Позиции контрольных разрядов в методе Хэмминга определены заранее, это разряды 1,2,4,8,… Разряды, входящие в каждую группу проверки представлены в следующей таблице (1-й разряд в каждой группе является контрольным).\n\nНомер группы проверки - 1.\nПроверяемые разряды - 1,3,5,7,9,11,13,15,…\n\nНомер группы проверки - 2.\nПроверяемые разряды - 2,3,6,7,10,11,14,15,18,19,22,23,…\n\nНомер группы проверки - 3.\nПроверяемые разряды - 4,5,6,7,12,13,14,15,20,21,22,23,…\n\nНомер группы проверки - 4.\n";

richTextBox1.Text = temp + "Проверяемые разряды - 8,9,10,11,12,13,14,15,24,…\nИз таблицы видно, что если например код Хэмминга содержит 9 разрядов, включая контрольные, то 1-я группа проверки содержит 1,3,5,7,9 разряды. 2-я группа проверки содержит 2,3,6,7 разряды. 3-группа проверки содержит 4,5,6,7 разряды и 4-я группа – 8,9 разряды. Каждой группе проверки приписывается 1, если проверка на четность обнаруживает ошибку и 0, если ошибки нет. Полученное двоичное число дает номер ошибочного разряда. Рассмотрим в качестве примера 5-ти разрядное двоичное число 10011. В этом случае, как следует из вышеприведенной таблицы, 1-я группа проверки состоит из 1,3, и 5-го разрядов. 2-я группа проверки состоит из 2 и 3-го разряда. 3-я группа проверки состоит из 4 и 5-го разрядов. Результат проверки на четность 1-й группы дает 0 (101), проверка 2-й группы дает 0 (00), проверка 3-й группы дает 0 (11). Таким образом, данное число не содержит ошибки. Искусственно введем ошибку, заменив, например, 4-й разряд на 0. В этом случае 1, 2 и 3-я проверки дадут соответственно 0, 0, 1. Полученное двоичное число 100 дает номер ошибочного разряда, т.е. 4.";

 break;

 }

 }

 }

 }

}

Error

using System;

using System.Drawing;

using System.Collections;

using System.ComponentModel;

using System.Windows.Forms;

namespace Study\_Programs

{

 public class Error : System.Windows.Forms.Form

 {

 private System.Windows.Forms.Button button1;

 private System.Windows.Forms.Label label1;

 private System.Windows.Forms.PictureBox pictureBox1;

 private System.ComponentModel.Container components = null;

 public Error(string error)

 {

 InitializeComponent();

 label1.Text=error;

 }

 protected override void Dispose( bool disposing )

 {

 if(disposing) if(components != null) components.Dispose();

 base.Dispose( disposing );

 }

 #region Windows Form Designer generated code

 private void InitializeComponent()

 {

 System.Resources.ResourceManager resources = new System.Resources.ResourceManager(typeof(Error));

 this.button1 = new System.Windows.Forms.Button();

 this.label1 = new System.Windows.Forms.Label();

 this.pictureBox1 = new System.Windows.Forms.PictureBox();

 this.SuspendLayout();

 //

 // button1

 //

 this.button1.Font = new System.Drawing.Font("MS Reference Sans Serif", 8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((System.Byte)(204)));

 this.button1.Location = new System.Drawing.Point(136, 56);

 this.button1.Name = "button1";

 this.button1.Size = new System.Drawing.Size(72, 24);

 this.button1.TabIndex = 0;

 this.button1.Text = "OK";

 this.button1.Click += new System.EventHandler(this.button1\_Click);

 //

 // label1

 //

 this.label1.Font = new System.Drawing.Font("MS Reference Sans Serif", 8.25F, System.Drawing.FontStyle.Regular, System.Drawing.GraphicsUnit.Point, ((System.Byte)(204)));

 this.label1.Location = new System.Drawing.Point(48, 8);

 this.label1.Name = "label1";

 this.label1.Size = new System.Drawing.Size(272, 40);

 this.label1.TabIndex = 1;

 this.label1.Text = "label1";

 this.label1.TextAlign = System.Drawing.ContentAlignment.MiddleCenter;

 //

 // pictureBox1

 //

 this.pictureBox1.Image = ((System.Drawing.Image)(resources.GetObject("pictureBox1.Image")));

 this.pictureBox1.Location = new System.Drawing.Point(8, 16);

 this.pictureBox1.Name = "pictureBox1";

 this.pictureBox1.Size = new System.Drawing.Size(32, 32);

 this.pictureBox1.TabIndex = 2;

 this.pictureBox1.TabStop = false;

 //

 // Error

 //

 this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);

 this.BackColor = System.Drawing.SystemColors.InactiveBorder;

 this.ClientSize = new System.Drawing.Size(338, 88);

 this.ControlBox = false;

 this.Controls.Add(this.pictureBox1);

 this.Controls.Add(this.label1);

 this.Controls.Add(this.button1);

 this.FormBorderStyle = System.Windows.Forms.FormBorderStyle.FixedSingle;

 this.Name = "Error";

 this.StartPosition = System.Windows.Forms.FormStartPosition.CenterScreen;

 this.Text = "Error";

 this.TopMost = true;

 this.ResumeLayout(false);

 }

 #endregion

 private void button1\_Click(object sender, System.EventArgs e){this.Close();}

 }

}