TOEIC 学習ソフトウェア開発入門

横田 壽

目次

第 1 章

ソフトウェア開発言語を選ぶ

1.1 C言語

日本の大学，企業では英語の能力を測るのに TOEIC が用いられています．そこで，TOEIC を教える側も学ぶ側も TOEIC 試験対策ができるソフトウェアが必要となります．ここでは，そんな人達の要求に答えられるような TOEIC の空欄補充問題を学べるソフトウェアの作成について説明します．

TOEIC の空欄補充問題ソフトウェア開発の対象には実際に TOEIC の授業を行っている英語の先生を中心に考えています．英語の先生の多くは，TOEIC 用のソフトウェアを自分で開発するためのプログラミング言語を学ぶ時間がありません．したがって，多くの先生方は市販のソフトを用いることとなり，物足りなさを感じている人は少なくないと思われます．しかし，自分でソフトウェアを作るとなると，特別な講習を受けたりしなければならないのではとお思いではないでしょうか．実は，作りたいソフトウェアにあった開発言語を選べば，ソフトウェアの開発は皆さんが思っているほど難しくはありません．日本で教育を受けた 40 歳以下の人は多分C言語を学んだと思います．40 歳以上の人も独学や講習でC言語を学んだ人は沢山いると思います．そのときの経験からプログラミングは面白くないと思っている人も沢山いると思います．それは，C言語でのプログラム作成では，エディタを用いておまじないのようなコマンドを一行一行キーボードから打ち込み，コンパイルするとエラーメッセージが出てくるといったことが度々あり，そのエラーメッセージを読んでもどこが間違っているのか分からない．色々とやっているうちに何がなんだか分からなくなってしまうというような経験をして来ているからだと思います．

C 言語の復習をしたい人は， または， を参照してください．

1.2 Cから C#へ

C言語は今でも大切な言語です．しかし，言語の構造がわれわれ人間のものの考え方とあまりにも違いすぎるのです．これを解決するために 1980 年代後半になると，C言語に代わってC ++ 言語とよばれるオブジェクト指向言語が生まれました．残念ながら，C ++ 言語もマスターするのは難しく一般の人に浸透することはありませんでした．1990 年の半ばになると Java 言語が台頭してきました．Java 言語は多くの人に受け入れられましたが，それでもマスターするのは簡単ではありません．そこで，Microsoft は 2002 年C ++ 言語と Java 言語のよいところを用いて新たなC#言語とよばれるものをつくり，ソフトウェアを作成するのに必要なエディタ，コンパイラ，ビジュアル環境を統合した統合開発環境 Visual C#というものを作りました．大学には Microsoft の製品を嫌う人が沢山いますが，世の中の 95% の人が利用している Windows を無視することは

できません．ましてや，ソフトウェア開発のプロフェッショナルではない人たちが自分の使うソフトウェアを作りたいと考えるときに，プロと呼ばれている人たちでも，マスターするのに何年もかかったような言語を学ばせるのは何か変です．そこで，ここでは，Visual C#を用いることにします．

2003 年に Visual Studio.NET2003 が開発され, ソフトウェアの開発環境は整いました．Visual Studio.NET の中に，Visual C#.NET は含まれています．ただ，残念なことに Visual Studio.NET2003(現在は Visual Studio.NET2005) は無料ではありません．無料のものがよい人は，Visual C# 2005Express が Microsoft から無償で提供されています．

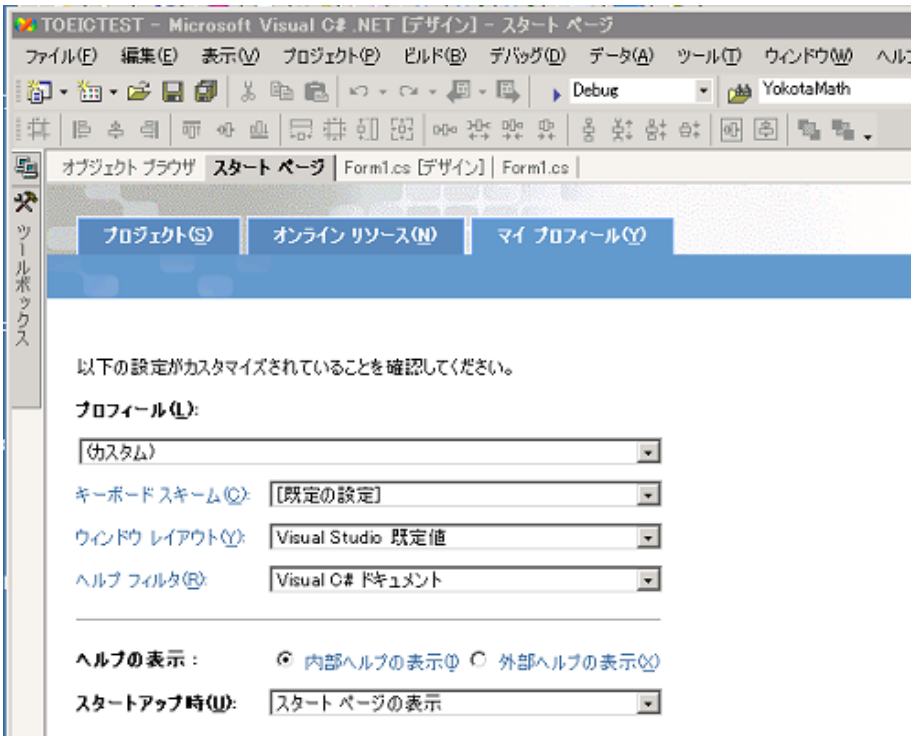
第 2 章

Visual Studio.NET で Windows アプリケーションを作成する

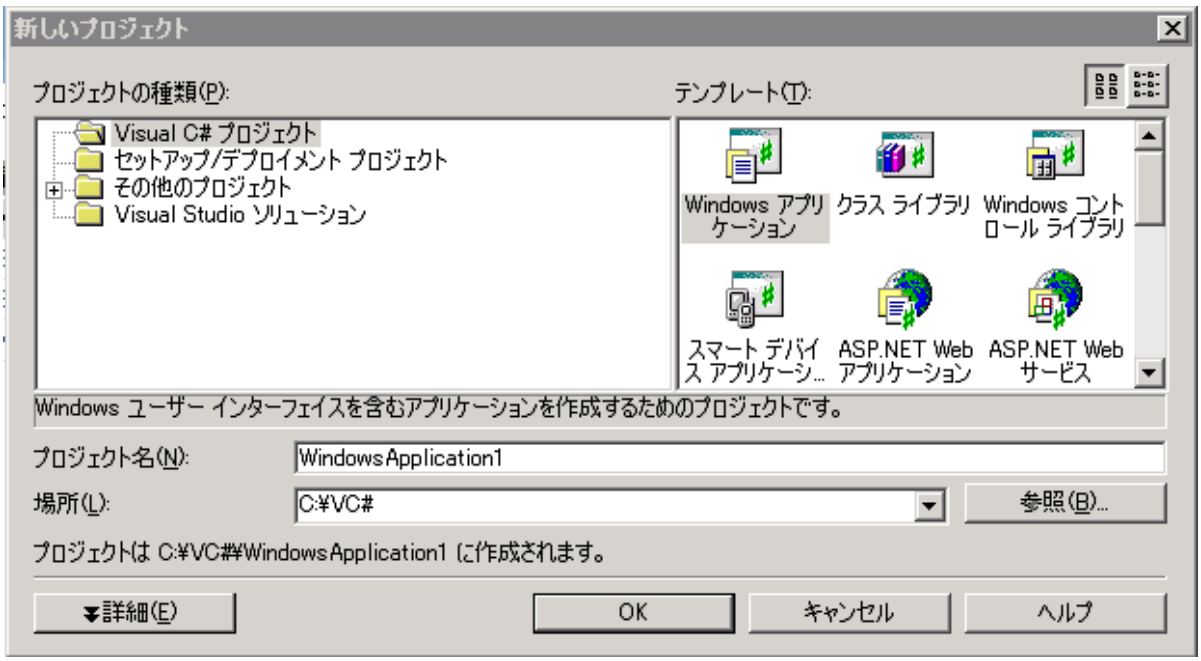
ソフトウェア開発に用いる言語が決まったら，実際にソフトウェアを作るのが，言語を学ぶ最良の方法であると言われています．

2.1 プロジェクトの作成

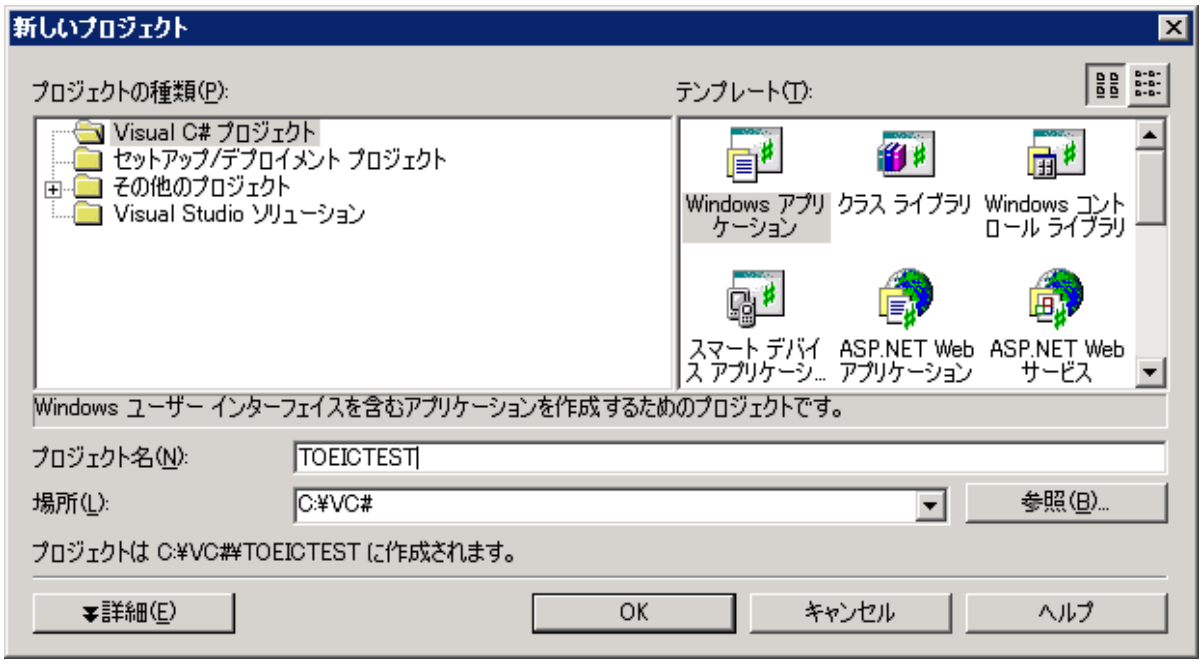
1. 「スタート」→「すべてのプログラム」→「Microsoft Visual Studio.NET2003」をポイントする．
   ⇒ サブメニューが表示されます．
2. 「Microsoft Visual Studio.NET2003」をクリックする．
   ⇒ Visual Studio.NET2003 が起動します．



3.「ファイル」→「新規作成」→「プロジェクト」をクリックする.
⇒「新しいプロジェクト」ダイアログボックスが表示されます.

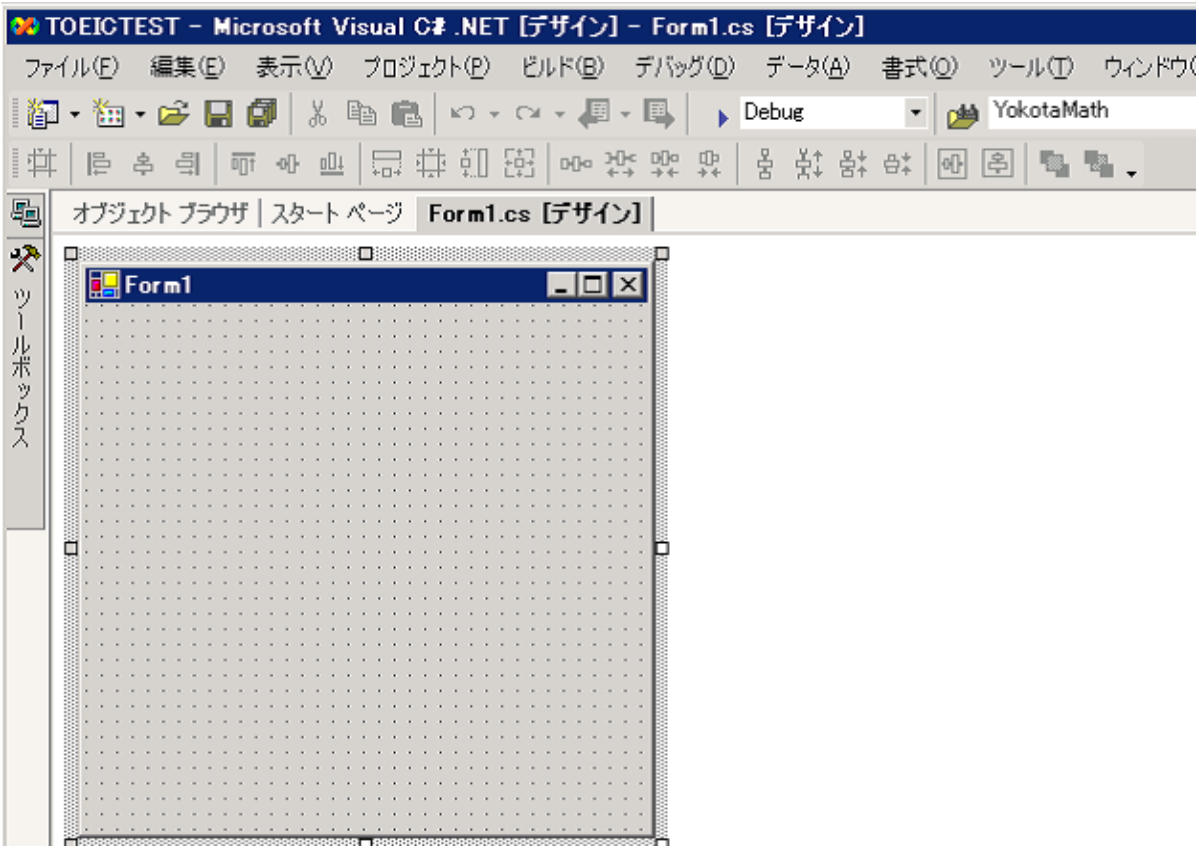


4.「プロジェクトの種類」ボックス一覧から,「Visual C#プロジェクト」をクリックする.
5.「テンプレート」ボックスの一覧から,「Windows アプリケーション」をクリックする.
6.「プロジェクト名」ボックスに TOEICTEST を入力する.
7. 場所ボックスに，C:¥VC#と入力する.



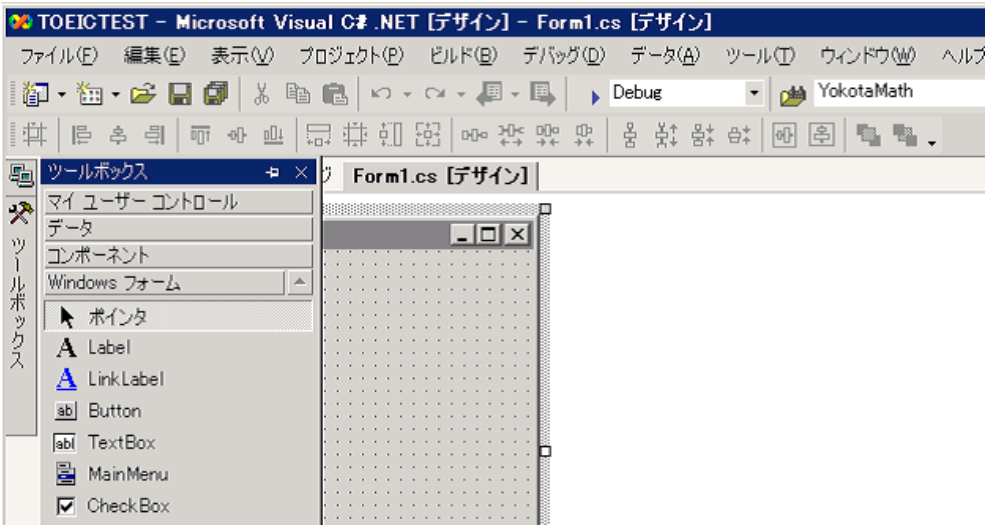
8.「ＯＫ」をクリックする．

⇒ 新しいプロジェクトが作成されます．

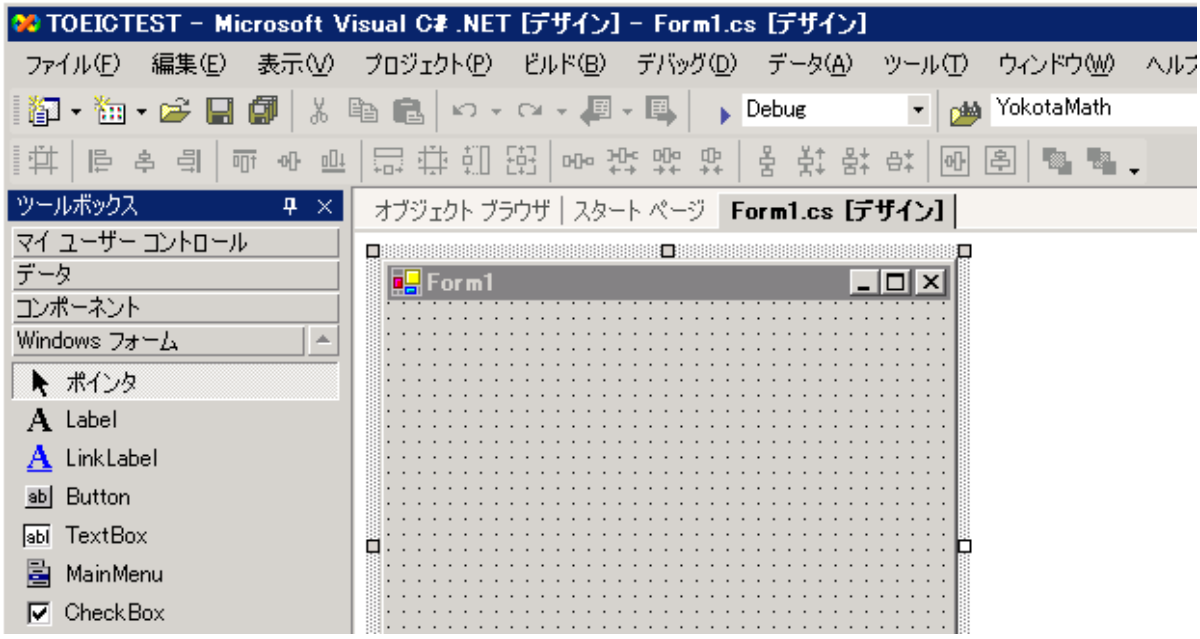


2.2　フォームの作成

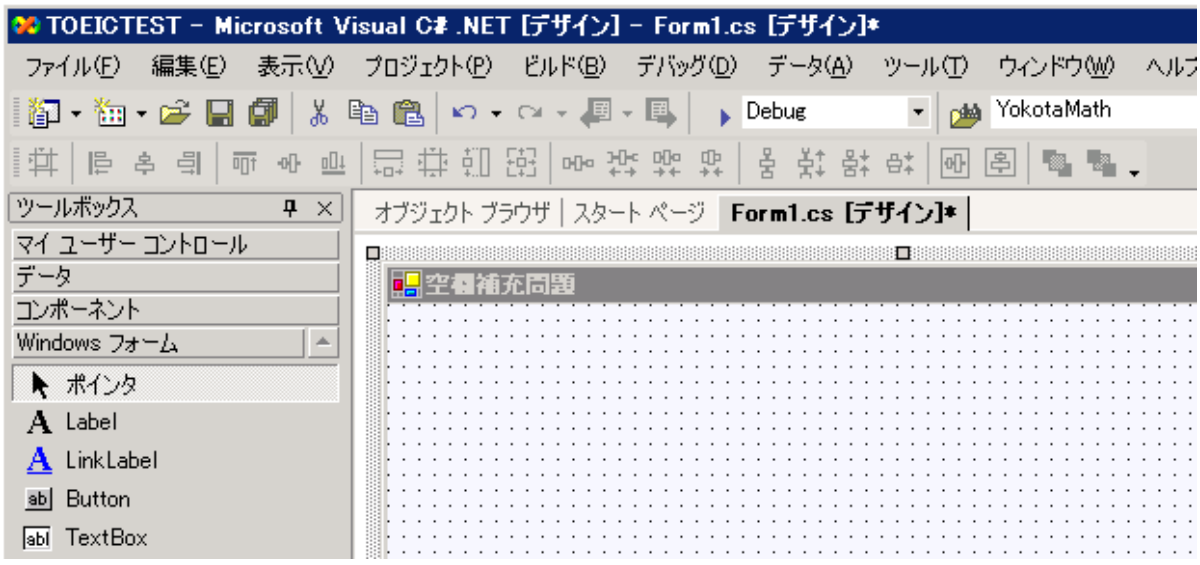
1.「ツールボックス」をクリックすると，ツールボックスの中が見られるようになります．



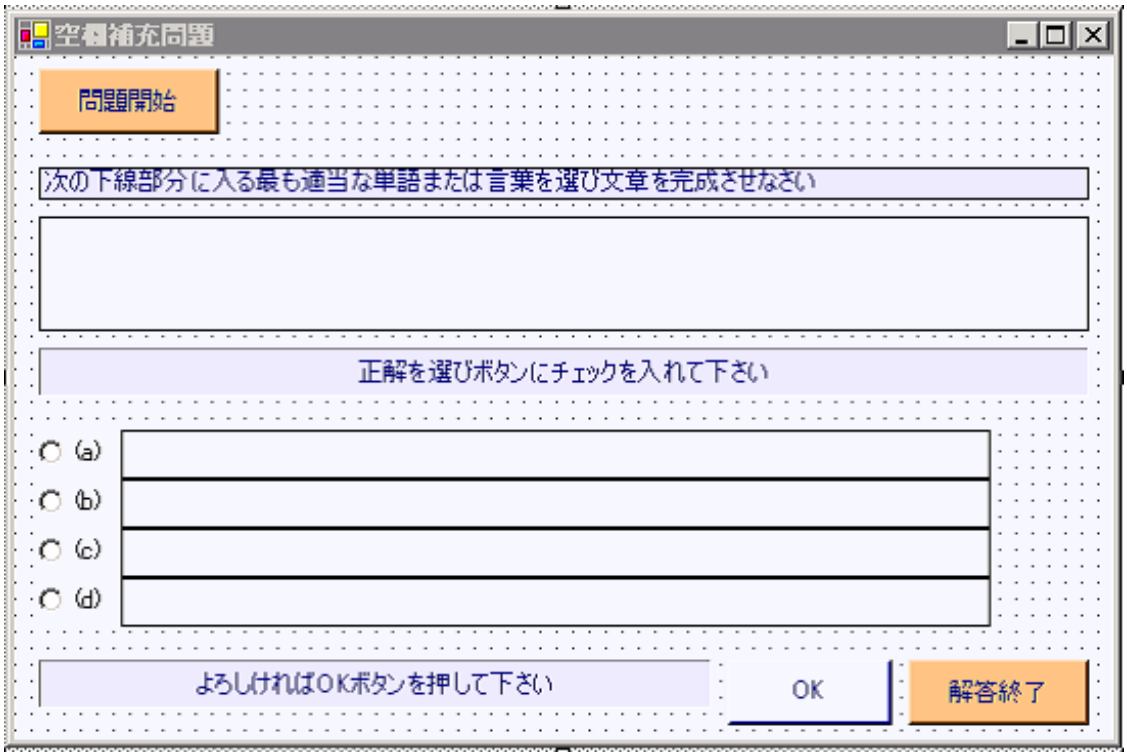
2. ツールボックス横のピンをクリックし，ピン止めをします．



3. Form1 と書いてあるのが，空欄補充問題が表示される画面になります．そこで，Form1 のプロパティの設定を行っていきます．
   (a) 画面の大きさを横 544, 縦 360 ピクセルに設定します．
   ⇒ Form1 のプロパティの Size と書いてあるところの右にある数字を 544,360 に変更
   (b) Form1 の BackColor を 244,244,255 に設定します．
   ⇒ Form1 のプロパティの BackColor の値を 244,244,255 と入力
   (c) Form1 のタイトルバーに表示される文字を変更します．
   ⇒Form1 のプロパティの Text のところに空欄補充問題と入力
4. 次のような画面になります．



5. ここで，どんな画面にするのか少し考えましょう．Windwos を使っている人たちならば，ボタンを押すことで問題が表示するのがいいかも知れませんね．そして，4 択の解答がその下に表示され，それぞれの解答の前にはラジオボタンとよばれる選択ボタンを付けるのがよいでしょう．最後に，問題の解答が済んだら，ボタンを押して正誤判定を行い，正解の場合には，正解と表示され，不正解の場合は不正解と表示されるのがよいでしょう．終了するときは，終了ボタンで終了するようにします．この要求をもとに画面設計を行うと次のようなものになります．



2.3　コントロールの追加とプロパティの設定

1. Form1 にボタンを付けるには，ツールボックスから Button を選び，ドラッグ&ドロップするか，ツールボックス上でダブルクリックすると，フォーム上にボタンができます．このボタンのプロパティ (ボタン上で右クリック→「プロパティ」) で，
   (Name) を btnStart
   BackColor を 255,192,198
   ForeColor を Navy
   Text を問題開始
   に変更すると，先ほどの画面のボタンと同じになります．
2. Form1 上で文章を表示するには，label を用います．ツールボックスから label を選び，フォーム上においてください．このラベルのプロパティで，
   (Name) を lblOutQuestion
   Size を 512,16

BackColor を 234,234,255
BorderStyle を FixedSingle
Text を下の下線部に入る最も適当な単語または言葉を選び文章を完成させなさい
と記入

3. Form1 上で問題文を表示するには，label を用います．ツールボックスから label を選び，フォーム上においてください．このラベルのプロパティで，
(Name) を lblOutQuestion
Size を 512,56
BackColor を 244,244,255
BorderStyle を FixedSingle
に変更
4. 正解にチェックを入れるものとして，ツールボックスから RadioButton を選びフォーム上に持ってきます．その横の (a) を表示するには label を用いています．さらに，4 択の解答を表示するには，label を使っています．各自で画面設計の画面をまねて作成してください．ソフトウェア開発のよい習慣として，それぞれのラベルには分かりやすい名前を付けておきます．ちなみに私は，lblAnswer1 から lblAnswer4 を使って (Name) プロパティの設定を行っています．
(Name) を lblAnswer1
BackColor を 244,244,255
ForeColor を Navy
Text を空白に設定．同様に，残りのラベルも設定
5. OK ボタンのプロパティは次のように変更します．
(Name) を btnOK
BackColor を 244,244,255
ForeColor を Navy
Text を OK
6. 解答終了ボタンのプロパティは次のように変更します．
(Name) を btnEnd
BackColor を 255,192,128
ForeColor を Navy
Text を解答終了

2.4　イベントハンドラの追加

Form1 にイベントハンドラの追加を行います．btnStart ボタンをダブルクリックするか，btnStart のプロパティで稲妻の記号をクリックすると，ボタンコントロールが発生することができるイベントが表示されます．その中から Click を選んでダブルクリックすると，btnStart_Click というイベントハンドラが追加されます．

コントロール名	イベント名	イベントハンドラ名
btnStart	Click	btnStart_Click
btnOK	Click	btnOK_Click
btnEnd	Click	btnEnd_Click

2.5 ソースコードの追加

- イベントハンドラをフォームに追加すると，ソースコード上に次のようなものが追加されます.

```
private void btnStart_Click(object sender, System.EventArgs e)
{
}
```

この { から } の中に，次のソースコードを記述します.

```
private void btnStart_Click(object sender, System.EventArgs e)
{
        getQuestion();
}
```

- 「OK」ボタンをダブルクリックして，次の青い部分を追加

```
private void btnOK_Click(object sender, System.EventArgs e)
{
        if(radioButtonChecked() == true)
        {
                correctAnswer();
        }
        else
        {
                MessageBox.Show(”項目のチェックをして下さい”);
        }
 }
```

- 「終了」ボタンをダブルクリックして，次の青い部分を追加

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

2.6 プライベートメソッドのコード

```
#region 問題の獲得
private void getQuestion()
{
    #region 問題文のデータを取得
    // 問題の数を取得
    StreamReader sr = new StreamReader(”c:¥Program Files¥yokota lab¥ENGLEASY¥EnglishEasyFillBlank.csv”);
    string text = sr.ReadToEnd();
    // text に改行記号をつけない
```

```
text = text.Replace('¥r',' ');
sr.Close();
string[] stext = text.Split("¥n');
rowNumber = int.Parse(stext[0].ToString());
// 探索回数
searchQuestionCount++;
// 問題の中から一問ランダムに抽出
Random r = new Random();
questionNumber = r.Next(2,rowNumber+1);
// 同一問題かチェック
if(sameQuestionNumber(questionNumber) == true)
{
    text = stext[questionNumber];
    // text が" "で囲まれている場合. つまり，文中にカンマが含まれている場合.
    // "When I was busy, I did not go to school.", answer,
    if(text.IndexOf("'') ¿= 0)
    {
        int ind = text.IndexOf("'');
        text = text.Substring(ind+1);
        int end = text.IndexOf("'');
        question = text.Substring(ind,end);
        stext = text.Substring(end+1).Split(',');
        selctAnswer[0] = stext[1];
        selctAnswer[1] = stext[2];
        selctAnswer[2] = stext[3];
        selctAnswer[3] = stext[4];
        // 改行文字をスペースで置き換えたので，スペースを削除する
        stext[5] = stext[5].Replace(" ","");
        selctAnswer[4] = stext[5];
    }
    else
    {
        stext = text.Split(',');
        question = stext[0];
        selctAnswer[0] = stext[1];
        selctAnswer[1] = stext[2];
        selctAnswer[2] = stext[3];
        selctAnswer[3] = stext[4];
        stext[5] = stext[5].Replace(" ","");
        selctAnswer[4] = stext[5];
    }
}
else
{
    if(searchQuestionCount < rowNumber)
    {
        getQuestion();
    }
    else
```

```
        {
        }
    }
    #endregion
    // lblOutQuesiton への書き込み
    lblOutQuestion.Text = question;
    lblAnswer1.Text = selctAnswer[0];
    lblAnswer2.Text = selctAnswer[1];
    lblAnswer3.Text = selctAnswer[2];
    lblAnswer4.Text = selctAnswer[3];
    // 問題数を 1 増やす
    questionCount++;
}
#endregion

#region 同一の問題かチェック
private bool sameQuestionNumber(int questionNumber)
{
    foreach(int qnum in sameQuestion)
    {
        if(questionNumber == qnum)
        {
            return false;
        }
    }
    sameQuestion.Add(questionNumber);
    return true;
}
#endregion

#region ラジオボタン
private bool radioButtonChecked()
{
    if(radioButton1.Checked == false && radioButton2.Checked == false
        && radioButton3.Checked == false && radioButton4.Checked == false)
    {
    return false;
    }
    else
    {
        return true;
    }
}
#endregion

#region 正誤判定
private void correctAnswer()
{
    if(radioButton1.Checked == true && selctAnswer[4] == "(a)" ||
```

```
        radioButton2.Checked == true && selctAnswer[4] == "(b)" ||
        radioButton3.Checked == true && selctAnswer[4] == "(c)"||
        radioButton4.Checked == true && selctAnswer[4] == "(d)")
    {
        MessageBox.Show("正解です");
        correctAnsCount++;
        radioButton1.Checked = false;
        radioButton2.Checked = false;
        radioButton3.Checked = false;
        radioButton4.Checked = false;
        getQuestion();
    }
    else
    {
        wrongAnsCount++;
        MessageBox.Show("不正解です");
        radioButton1.Checked = false;
        radioButton2.Checked = false;
        radioButton3.Checked = false;
        radioButton4.Checked = false;
    }
}
#endregion

#region 結果の表示
private void questionResult()
{
    questionCount = correctAnsCount + wrongAnsCount;
    MessageBox.Show("あなたの成績は" + questionCount + "回挑戦中" + correctAnsCount + "問の正解でした");
}
#endregion

#endregion
```

2.7　名前空間とフィールドの追加

この状態でソリューションのビルド (プログラムを実行できるようにする) をするとエラーがでます.

- ビルドする前にファイルの入出力を行っていますので，名前空間 IO を使えるようにします.

  ⇒ ツールバーの「表示」→「ソース」でソースコードを開いて，using System.IO; を追加
- リストを用いていますので，名前空間 Collections を使えるようにします.
  ⇒ ツールバーの「表示」→「ソース」でソースコードを開いて，using System.Collections; を追加
- public class Form1 の下に以下の変数 (オブジェクト指向プログラミングではフィールドという) を追加します.
  ⇒ // ファイル内

```
private string question = "";
private int questionNumber = 0;
private int rowNumber = 1;
private string[] selctAnswer = new string[5];
```

```
ArrayList sameQuestion = new ArrayList();
// 不正解数
private int wrongAnsCount = 0;
// 問題数
private int questionCount = 0;
// 正解数
private int correctAnsCount = 0;
// 探索回数
private int searchQuestionCount = 0;
```

2.8 ソリューションのビルドと実行

.NET Framework の中核であるアプリケーション実行環境（ランタイム環境）にあたるのが、Common Language Runtime （以下、CLR とよぶ）です． CLR は、専用の中間コードを解釈、実行するエンジン（仮想マシン）です。この中間コードは、「Intermediate Language」（ IL ）とよばれ、C# や Visual Basic.NET で開発したアプリケーションは、コンパイラによってこの共通の中間コードになります．

1. ツールバーから「ビルド」→「ソリューションのビルド」でクリックする
   ⇒ 今書いたプログラムのビルドが行われ，中間コードに変換されます．ビルドが完了すると「出力」ウィンドウに次の行が表示されます．
   ビルド　：　1 正常終了，　0 失敗，　0 スキップ
   ここで，エラーがなければアプリケーションの実行が可能となります．エラーがある場合，画面の下に表示されますので，エラーを取り除く作業を行います．この作業をバグとりまたは虫取りといいます．Visual Studio ではこの作業が簡単にできるように，表示されたエラーのところにマウスを持っていきクリックすると本文中のエラーの発生している場所にカーソルを移動します．
2. ツールバーから「デバッグ」→「開始」をクリックする
   ⇒ プログラムが開始され画面にアプリケーションが表示されます．