筆記課題　E-46B

筆記課題

「パソコン計測制御のための基礎知識（Visual C#）」

|  |
| --- |
| １　作業時間  　　 ９０分  ２　配付資料  問題用紙，解答用紙  ３　課題作成、提出方法  解答用紙のみを回収します  ４　注意事項  この筆記試験はVisual C#(省略表記C#)は特記がないかぎりVisual C# 2005(言語バージョン2.0) 以降に準拠した仕様です。 |

１．以下の文章中の**（１）**～**（５）**について、適切な語句を**（ア）**～**（ス）**から選択し、解答欄に記入しなさい。

　Visual C#（以下VC#）は**（１）**社のプログラミング言語であり、主にWindowsで動作するアプリケーション開発で利用される。開発したアプリケーションは**（２）**上で動作する。VC#はマウス等の操作や状態の変化等に対して動作を行なう**（３）**型の言語である。

　VC#で開発可能なアプリケーションは**（４）**、WPFアプリケーション等がある。Web関連ではASP.NETやSilverlight等でもVC#の言語が利用されている。中でも**（４）**は、ボタンやラベルといったパーツを配置して**（５）**を作成することができる。

**（ア）**Apple **（イ）**Microsoft **（ウ）**Oracle

**（エ）**CUI **（オ）**GUI **（カ）**タッチ

**（キ）**WCFアプリケーション **（ク）**コンソールアプリケーション

**（ケ）**Windowsフォームアプリケーション **（コ）**MFC

**（サ）**.NET Framework **（シ）**Java **（ス）**イベントドリブン

２．以下の文章中の**（１）**～**（１０）**について、適切な語句を**（ア）**～**（ツ）**から選択し、解答欄に記入しなさい。

　VC#はオブジェクト指向のプログラミング言語であり、全てのデータ型はSystem.Object型を継承している。データ型には値型と参照型があり、**（１）**型はメモリ上のデータに直接アクセスをすることができ、VC#の組み込みデータ型では符号付き整数型の**（２）**や符号なし整数型の**（３）**、実数型では**（４）**が該当する。

**（５）**型はメモリ上に生成されたインスタンスのアドレスが格納されており、組込みデータ型ではobjectやstringが該当する。

　クラスや構造体では、オブジェクトが持つ状態の設定と参照を**（６）**、オブジェクトが行なう手続きを**（７）**という。クラスではインスタンス生成時に呼び出す処理を**（８）**、オブジェクトを破棄する処理を**（９）**、特定の事象が発生したことの通知となる**（１０）**を実装することができる。

**（ア）**構造体 **（イ）**クラス **（ウ）**intとuint

**（エ）**ulongとubyte **（オ）**intとshort **（カ）**uintとbyte

**（キ）**ushortとdouble **（ク）**floatとdouble **（ケ）**singleとdecimal

**（コ）**string **（サ）**char **（シ）**デストラクタ

**（ス）**コンストラクタ **（セ）**メソッド **（ソ）**プロパティ

**（タ）**イベント **（チ）**値 **（ツ）**参照

３．以下のプログラムを実行したとき①で代入した数値が②～⑦で失われる箇所がどの箇所か回答欄の**（１）**に記入しなさい。またその理由について最も適切なものを以下の**（ア）**～**（オ）**から1つ選択し回答欄の**（２）**に記入しなさい。

sbyte n1; byte n2; short n3; ushort n4;

int n5; uint n6; long n7; ulong n8;

n8 = 12345; //①

n7 = (long)n8; //②

n6 = (uint)n7; //③

n5 = (int)n6; //④

n4 = (ushort)n5; //⑤

n3 = (short)n4; //⑥

n2 = (byte)n3; //⑦

n1 = (sbyte)n2; //⑧

**（ア）**左辺の型が負の値に対応しない。

**（イ）**暗黙の型変換によりデータが失われる。

**（ウ）**明示的な型変換(キャスト)で指定された型が小さいためデータが失われる。

**（エ）**明示的な型変換(キャスト)で指定された型が大きいためデータが失われる。

**（オ）**左辺の型と右辺の型が異なるため。

明示的な型変換(キャスト)で指定された型よ

**（オ）**左辺と右辺が共に負の値に対応しないため。

４．以下のメソッドはある周波数f[Hz]において、抵抗R[Ω]とコンデンサC[F]によるハイパスフィルタの電圧利得Gv[dB]を求めるメソッドを定義したものです。利得を求める式を以下のものとしたとき、以下System.Mathクラスのメンバーやメソッドを利用し適切な記述を回答欄の**（１）**～**（５）**へ記述しなさい。

　　

Using System;

public partial class Form1 : Form 'クラス(問題には関係なし)

{

private （１） HpfGain(double f,double r, double c )

{

double gv, w, wcr; //利得Gvと角周波数ωとωCR算出用の変数

w = 2 \* （２） \* f; //角周波数ω算出

wcr = w \* c \* r; //ωCR算出

gv = 20 \* Math.Log10(（３）((（４）) / (1 + (（４）))); //電圧利得算出

（５） gv; //利得を戻り値として渡す

}

５．以下のソースコードはプログラムの実行時に2つの数値の比較値compValueが基準値stdValueに対してどれだけの誤差があるかをパーセントで算出し、誤差なし、誤差が±3％以内、±5％以内、誤差が±5%を越えている旨をメッセージボックスで表示します。**（１）**～**（６）**について、適切な記述を解答欄の**（１）**～**（６）**に記入しなさい。

short compValue, diffPercent;

**（１）** short stdValue = 10000; //定数宣言

compValue = short.Parse(InputNumber.Text); //テキストボックスより文字列を取得して変換

diffPercent = (compValue - stdValue) \* 100 / stdValue; //比較値と基準値の差分パーセント

//以下のifステートメントは"{}"を省略

if (diffPercent **（２）** 0)

MessageBox.Show("誤差なし");

else if (diffPercent **（３）** 3 && diffPercent **（４）** -3)

MessageBox.Show("誤差が±3%以内");

else if (diffPercent **（５）** 5 || diffPercent **（６）** -5)

MessageBox.Show("誤差が±5%越え");

else

MessageBox.Show("誤差が±5%以内");

６．以下の処理を実行した、ラベルShowSumLabelに表示される数値を以下の**（ア）**～**（オ）**から1つ選択し回答欄に記入しなさい。

int[] iArray = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};

int[] x = new int[5]; int[] y = new int[5];

int i = 0, sum = 0;

for (i = 1; i <= x.Length - 1; i += 1)

{

x[i] = iArray[i \* 2 + 1];

y[i] = iArray[i / 2];

}

sum = y[i % 2];

foreach (int j in x)

{

sum += j;

}

ShowSumLabel.Text = sum.ToString(); //sumの値をラベルに出力

７．以下のメソッドはパソコンから外部のディジタル入出力（DIO:Digital Input/Output）とアナログ入出力（AIO:Analog Input/Output）対応するDevice.IOクラス（独自に用意したクラス）のメソッドです。これらのメソッドとIOの外部に接続された各種機器を制御するプログラムについて以下の問いに答えなさい。

**（ア）**６ **（イ）**１２ **（ウ）**２４ **（エ）**３６ **（オ）**４５

|  |  |  |
| --- | --- | --- |
| IOクラス　コンストラクタ | | |
| 名前空間 | Device.IO | |
| 宣言 | public IO(byte **id**, ref int **errNo**) | |
| パラメータ1 | byte id | 使用するIOのID（ハードウェア番号1～127）を指定する。 |
| パラメータ2 | ref int errNo | インスタンス生成時に初期化を実行し、成功した場合は、0を、失敗した場合はエラー番号をerrNoに格納される。 |
| 動作 | 使用するIOポートのIDを指定して初期化する。通常はデバイスの使用開始時に実行する。 | |

|  |  |  |
| --- | --- | --- |
| IO.DigitalOutメソッド | | |
| 名前空間 | Device.IO | |
| 宣言 | public int **DigitalOut**(byte **port**, byte **data**) | |
| パラメータ1 | byte port | 使用するIOのポート番号を指定する。 |
| パラメータ２ | byte data | 出力するデータを1Byteで指定する。出力成功時は対応するビットが'1'のときHighの電圧が、'0'のときLowの電圧が出力される。 |
| 戻り値 | int | 0のとき出力成功、0以外の時はエラー番号を格納。 |
| 動作 | IOクラスのオブジェクトを使用して指定されたデータを信号として出力する。 | |

|  |  |  |
| --- | --- | --- |
| IO.DigitalInメソッド | | |
| 名前空間 | Device.IO | |
| 宣言 | public int **DigitalIn**(byte **port**, ref byte **data**) | |
| パラメータ1 | byte port | 使用するIOのポート番号を指定する。 |
| パラメータ２ | ref byte data | 入力されたデータを1Byteで取得する。入力成功時は対応する入力がHighのとき'1'を、Lowのとき'0'が所定のビットに格納される。 |
| 戻り値 | int | 0のとき入力成功、0以外の時はエラー番号を格納。 |
| 動作 | IOクラスのオブジェクトを使用して入力された信号をデータとして取得する。 | |

|  |  |  |
| --- | --- | --- |
| IO.AnalogOutメソッド | | |
| 名前空間 | Device.IO | |
| 宣言 | public int **AnalogOut**(byte **port**, short[] **data**) | |
| パラメータ1 | byte port, | 使用するIOのポート番号を指定する。 |
| パラメータ２ | short[] data | 出力する電圧をshort型の配列（data[0]～data[3]）がDA0～DA3に対応する4要素）で指定する。成功時の電圧はアナログ出力変換特性による。 |
| 戻り値 | Integer | 0のとき出力成功、0以外の時はエラー番号を格納。 |
| 動作 | IOクラスのオブジェクトを使用して指定されたデータをアナログ電圧として出力する。 | |

|  |  |  |
| --- | --- | --- |
| IO.AnalogInメソッド | | |
| 名前空間 | Device.IO | |
| 宣言 | public int **AnalogIn**(byte **port**, ref short[] **data**) | |
| パラメータ1 | byte port | 使用するIOのポート番号を指定する。 |
| パラメータ２ | As Short | 入力された電圧をShort型の配列（data(0)～data(3)がAD0～AD3に対応する4要素）で取得する。成功時の値はアナログ入力変換電圧による。 |
| 戻り値 | int | 0のとき入力成功、0以外の時はエラー番号を格納。 |
| 動作 | IOクラスのオブジェクトを使用してアナログ電圧をデータとして取得する。 | |

|  |  |
| --- | --- |
| IO.Disposeメソッド | |
| 名前空間 | Device.IO |
| 宣言 | public void Dispose() |
| 動作 | IOオブジェクトが使用するリソースを明示的に開放する。 |

回路図1：PORT0とPORT1の機器の接続

回路図１に関するソースコード

|  |
| --- |
| using Device;  public class Form1  {  //IOクラスのオブジェクト  private IO myIO;  private int errorNo;  private void Form1\_Load(object sender, EventArgs e)  {  myIO = new IO(1,ref errorNo); //プログラム起動時IOをID1で初期化  if (errorNo != 0) System.Windows.Forms.Application.Exit(); //初期化失敗時プログラム終了  **DisplayLED(0xaa); //初期化後のLEDの出力状態**  }  private void Form1\_FormClosing(object sender, FormClosingEventArgs e)  {  **（１）**　　　　 //プログラム終了時IOを開放する  }  private void ShowSwitchState()  {  byte inData = 0; //スイッチから取得したデータ  if (myIO.DigitalOut(0, inData) == 0) //スイッチ取得成功時  {  **SwitchStateLabel.Text = inData.ToString("X2"); //データを16進数2桁大文字で表示**  }  }  private void DisplayLED(byte data)  {  byte outData = 0; //引数に指定されたデータをLEDへ出力する  outData = (byte)((0x3f & data) | 0x05);  if (myIO.DigitalOut(1, data) != 0)  {  MessageBox.Show("出力エラー発生");  }  }  } |

７－１．回路図1に関する問題（スイッチとLED）

（１）以下の回路図１に関するソースコードにおいて空白部分　**（１）**　はDevice.IOクラスのリソースをプログラム終了時に開放するための処理です。この空白部分に当てはまる適切な記述（オブジェクト名とメソッド名や引数も含め）を回答欄**（１）**に記入しなさい。

（２）初期化成功時PORT1に接続されているLED0～LED7の状態について、点灯しているものに**○**、消灯しているものに×を回答欄**（２）**に記入しなさい。なお出力処理中エラーが発生しないものとします。

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| LED | LED7 | LED6 | LED5 | LED4 | LED3 | LED2 | LED1 | LED0 |
| 状態 |  |  |  |  |  |  |  |  |

（３）PORT0に接続されているスイッチSW0〜SW7が以下の状態でShowSwitchStateメソッドを実行したときにラベルに表示される内容を**（ア）**～**（オ）**から選択し回答欄**（３）**に記入しなさい。

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| スイッチ | SW7 | SW6 | SW5 | SW4 | SW3 | SW2 | SW1 | SW0 |
| 状態 | OFF | ON | OFF | OFF | ON | ON | OFF | ON |

**（ア）**Ｂ２ **（イ）**０ｘＢ２ **（ウ）**Ｂ２Ｈ **（エ）**ｂ２ **（オ）**＆Ｈｂ２

７－２．回路図２に関する問題（モータ制御）

先ほどと同じクラスを使用し、PORT2からPORT5に対しモータドライバ、タコジェネレータをそれぞれ接続したものです。これによりモータの回転数を制御するプログラムについて以下の問いに答えなさい。



|  |  |  |
| --- | --- | --- |
| 信号名 | 機能 | 備考 |
| VRPM | モータの回転数を2V/krpmで取得する。  絶対値(正の電圧)で出力する。 | 1krm(=1000rpm)のとき2Vが入力される。  回転方向は検出できない。 |
| RPMV | モータの回転数を1000rpm/Vで制御する。 | 1V出力でモータが1000rpmで回転する。 |
| DIR | モータの回転方向を取得する。 | 停止中の状態は保証しない。 |
| VRPM | モータに過負荷がかかっているかを検出する。 | 過負荷を検出してもモータ制御の変化なし。 |
| CCW | モータを逆転制御する。 | モータの制御はCW、CCW、BRAKEの組み合わせで行なわれる。ただし停止状態からの回転はRPMVの電圧が0.5V以上必要となる。 |
| CW | モータを正転制御する。 |
| BRAKE | モータにブレーキをかける。 |

※rpm(revolution per minute)で単位分あたりの回転数を表し1krpm=1000rpmの場合1分に1000回転

回路図２に関するソースコード（次ページに続きあり）

|  |
| --- |
| public partial class Form3 : Form  {  private int errorNo;  private IO myIO;  private short[] da = new short[4]; //DA0～DA3に出力するための配列  private void Form3\_Load(object sender, EventArgs e)  {  myIO = new IO(1, ref errorNo); //プログラム起動時IOをID1で初期化  if (errorNo != 0) System.Windows.Forms.Application.Exit(); //初期化失敗時終了  myIO.DigitalOut(3, 0); //モータを停止状態  da = new short[4] { 0, 0, 0, 0 }; //配列の初期化  myIO.AnalogOut(5, da); //配列daのアナログ出力  }  private void MotorNormalRotate()  {  da[0] = 　　**（１）**　　; //3000rpmとなる電圧を出力  myIO.DigitalOut(　　**（１）**　　); //正転となるデータをPORT5へ出力  myIO.AnalogOut(（　　**（１）**　　); //配列daをアナログ出力  }  private void MotorState(ref short rpm, ref bool cw)  {  short[] ad = new short[4]; //電圧を格納する配列  byte dir = 0; //DIRが接続されているポートのデータ  //アナログとディジタル電圧の取得  myIO.AnalogIn(2, ref ad); //ad0～ad3の取得  myIO.DigitalIn(4, ref dir); //dirの取得  rpm = (short)(ad[0] / 　　**（２）**　　); //回転数をrpmで算出  if ((dir & 　　**（３）**　　) == 0) { //DIRが格納されているビットを1でマスク  cw = true; //モータが正転状態  } else {  cw = false; //モータが逆転状態  }  }  private void MotorSequential()  {  //このメソッド呼び出し前にMotorNormalRotateを実行している  for (da[0] = 3000; da[0] <= 6000; da[0] += 60) //①  {  myIO.AnalogOut(3, da);  System.Threading.Thread.Sleep(10); //10msの待ち時間  }  System.Threading.Thread.Sleep(1000); //②1000msの待ち時間  for (da[0] = 6000; da[0] <= 2000; da[0] -= 80) //③  {  myIO.AnalogOut(3, da);  System.Threading.Thread.Sleep(10); //10msの待ち時間  }  for (da[0] =　　　　　**（４）**　　　　　) { //④  myIO.AnalogOut(3, da);  System.Threading.Thread.Sleep(10); //10msの待ち時間  }  while (　　**（５）**　　) //⑤  {  da[0] 　　**（６）**　　 160; //出力電圧を減算  myIO.AnalogOut(3, da);  System.Threading.Thread.Sleep(10); //10msの待ち時間  }  myIO.DigitalOut(5, 0x04); //モータをブレーキ  }  }  } |

（１）このモータの通常回転状態を正転・3000rpmで回転させるためのメソッドMotorNormalRotateに必要な記述を以下**（ア）**～**（カ）**から1つ選択し回答欄**（１）**に記入しなさい。

**（ア） （イ）** **（ウ）**

da[0] = 1500 da[0] = 1500 da[0] = 1500

myIO.DigitalOut(5, 1) myIO.DigitalOut(5, 0x02) myIO.DigitalOut(5, 0x05)

myIO.AnalogOut(2, da) myIO.AnalogOut(3, da) myIO.AnalogOut(4, da)

**（エ） （オ）** **（カ）**

da[0] = 3000 da[0] = 3000 da[0] = 3000

myIO.DigitalOut(5, 0x05) myIO.DigitalOut(5, 0x02) myIO.DigitalOut(5, 0x02)

myIO.AnalogOut(2, da) myIO.AnalogOut(3, da) myIO.AnalogOut(4, da)

（２）モータに接続されているタコジェネレータから回転数と回転方向を取得するメソッドMotorStateについて参照渡しされる引数rpmの演算式と、DIRの状態を判別するｉｆステートメントの条件式を回答欄**（２**

**）**と**（３）**に記入しなさい。

（３）NomalState実行中（正転・3000rpm）からMotorSequentialメソッド実行し以下のタイムチャートと近似の動作になるような記述を回答欄**（４）**～**（６）**に記入しなさい。

ただしIOの仕様によりデータ出力の間隔は10msを設け実行するものとします。



解答用紙

|  |  |  |  |
| --- | --- | --- | --- |
| 入所期 | 氏名 | 合計点 | 評価判定 |
| 平成　　　　年　　　月生 |  |  |  |

１．（２点×５）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| （１） | （２） | （３） | （４） | （５） |

２．（２点×１０）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| （１） | （２） | （３） | （４） | （５） |
| （６） | （７） | （８） | （９） | （１０） |

３．（２点×２）

|  |  |
| --- | --- |
| （１６） | （１７） |

４． （３点×５）

|  |  |  |
| --- | --- | --- |
| （１） | （４） | （３） |
| （４） | （５） |  |

５．（２点×６）

|  |  |  |
| --- | --- | --- |
| （１） | （２） | （３） |
| （４） | （５） | （６） |

６．（３点）

|  |
| --- |
|  |

７－１．（４点×３）

|  |  |
| --- | --- |
| （１） | （２） |
| |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | LED | LED7 | LED6 | LED5 | LED4 | LED3 | LED2 | LED1 | LED0 | | 状態 |  |  |  |  |  |  |  |  |   （３） | |

７－２（４点×６）

|  |  |  |
| --- | --- | --- |
| （１） | （２） | （３） |
| （４） | | （５） |
| （６） | | |