令和元年（ワ）第１０９４０号　損害賠償請求事件

原告　森　次　茂　廣

被告　株式会社[redacted]

第５準備書面

令和３年１月１４日

大阪地方裁判所第２６民事部合議係　御中

原告訴訟代理人弁護士　[redacted]

同　[redacted]

同　[redacted]

（担当）同　[redacted]

第１　被告準備書面４に関する主張

１　ソースコードの不一致について

（１）　本件プログラム１及び本件プログラム２のソースコードについて（甲３号証，第２準備書面別紙２，甲５号証，第３準備書面別紙２について）

本件プログラム１及び本件プログラム２はもともと一緒に使用されることが想定されていたことから，本件プログラム１と本件プログラム２には共通ルーチンであるソースコードが存在してい

た。その共通ルーチンについて，これまで証拠として提出するにあたって一部加えるのを失念しているものがあり，欠落や齟齬が生じてしまった。

したがって，共通ルーチンを加えた本件プログラム１及び本件プログラム２のソースコードを追って提出する。

（２）　本件プログラム３について（甲８－１，甲８－３（左側））

本件プログラム３は，焼山トンネルの発破振動計測のために使用されたものであるが，第４準備書面の３で述べたとおり，高山トンネル工事の騒音振動測定プログラム，志津見ダムの発破振動測定プログラムと順次改良されて作成されている。

そして，原告は，本件訴訟提起前に，甲８－３の右側に記載された被告プログラムのソースコードについて，どのプログラムからコピーしたものか調べるために，プログラム３のソースコードだけではなく，高山トンネル工事の騒音測定プログラムのソースコード，志津見ダムの発破振動測定プログラムのソースコードについてもそれぞれ比較検討している。甲８－３の左に記載されているソースコードは，志津見ダムの発破振動測定プログラムのソースコードと比較したものであり，本件プログラム３と類似の点があることから誤って本件プログラム３と被告プログラムを比較検討したものとして提出したものである。

したがって，本件プログラム３と被告作成プログラムを比較した資料を改めて甲２７号証として提出する。なお，乙１５号証「Ｐ２～Ｐ１７」の削除と記載されている箇所については，アナログデジタル変換ボードメーカーからの定義ファイルであり，かつ被告プログラムに存在しないファイルであるため，あらかじめ比較

の検討から外したものである。甲２７号証においても，検討から外している。

（３）　本件プログラム５について（甲１３－１，甲１３－３（左側））

被告が準備書面４の７頁で指摘する５つのファイルについては，開発ツール（ＶＢ６）が自動で生成するファイルであり，著作物に該当しないことが明らかな箇所であるため，訴訟提起時に甲１３－１号証として出力しなかった。甲１３－３号証として，そのファイルが出力されてしまったのは，比較ツールで一括処理したものをそのまま提出したことに起因している。

甲１３－１と甲１３－３（左側）に被告準備書面４で指摘される齟齬が生じたのは以上の経緯によるが，甲１３－１について修正したものを追って提出する予定である。

第２　原告の主張

１　本件プログラム４の著作物性

（１）　本件プログラム４の概要

本件プログラム４は，橋梁などで風を常時観測し，定期的な測定と臨時的な測定を行うプログラムである。このうち，定期的な観測とは，連続的に時間を１０分に区切り，その時間内の平均風速，最多風向，最大風速等を算定することや１時間ごとに正時１０分前のデータを元にして平均風速，最多風向，最大風速等を算出することを指す。また，臨時的な測定とは，平均風速及び瞬間風速が既定の値を超えた際に，その前後１０分間の測定データを記録することを指す。

（２）　ソースコードの構造について

ア　本件プログラム４の代表的な処理は，①ＡＤ変換器を制御するサブルーチン，②風観測データを回収処理するルーチン，③回収処理された風観測データを演算及び収録するルーチンである。

以下，①乃至③について述べる。

イ　①ＡＤ変換器を制御するサブルーチンについて（甲１１－１及び第３準備書面別紙４　１２頁～１８頁参照）

本件プログラム４は，ＡＤ変換器（外部装置からのアナログ信号をデジタルデータに変換するためのデバイスのこと）を使用していることから，ＡＤ変換器の制御のためのソースコードを記載する必要がある。

ＡＤ変換器の制御は，メインフォームのイベント（動作，出来事又はそれらを表現する信号）から発生するものが多いため，ソースコードの記載の方法の選択肢として，メインフォーム内にプログラムコードを記載するという方法等があるが，原告は，メインフォームのソースコードのボリュームを減らすために，ソースコードを分離して記載する方法を採用している。

ソースコードを分離する方法を採用した場合，ソースコードの記載方法として，一般的な方法としては，メインフォームのインスタンスを引数として渡す方法が選択肢として挙げられる。

この方法をとる場合，すべてのＡＤ変換器関係のサブルーチンで、AnalogInputInitialize(Me)の様に、引数でMeを指定しなくてはならなくなる。

メインフォームが複数存在する可能性がある場合は，この方法を採用することも考えられるが，原告は，本件プログラム４についてメインフォームを一つにすることを前提に設計していたことか

ら，以下の＜原告が作成したソースコード＞のとおり，Ｗｉｔｈ文を使ってメインフォームのインスタンスを参照する方法を採用した（甲１１－１及び第３準備書面別紙４　１２頁乃至１８頁参照）。

＜原告が作成したソースコード＞

```
Public Function AnalogInputInitialize() As Boolean
  Dim mRet As Long, mErrFlag As Boolean
  Dim AiMaxChannels As Integer, i As Integer, j As Integer
  With fMainForm
    '初期化処理
    mRet = AioInit(.txtAioDeviceName, pvAioId)
    If DispAioMessage(mRet, "AioInit") = False Then mErrFlag = True
    For i = 0 To AiInpChN - 1: For j = 0 To AiAveDataN - 1: pvData(i, j) =
    NullData:
    Next: Next
    'デバイスのリセット
    mRet = AioResetDevice(pvAioId)
    If DispAioMessage(mRet, "AioResetDevice") = False Then mErrFlag = True
    '最大チャネル数の取得
    mRet = AioGetAiMaxChannels(pvAioId, AiMaxChannels)
    If DispAioMessage(mRet, "AioGetAiMaxChannels") = False Then mErrFlag =
    True
    If AiMaxChannels = 0 Then
    ListBox_StatusAdd .lstAioStatus, "アナログ入力の機能を持っていません"
    mErrFlag = True
```

End If

Debug.Print "AioGetAiMaxChannels" + Str(AiMaxChannels)

ListBox_StatusAdd .IslAioStatus, "初期化 : " + IIf(mErrFlag = False, "正常終了", "異常終了")

End With

End Function

このようなＷｉｔｈ文を使用することにより，AnalogInputInitialize()の様に、Me を記述しなくてもよい状態になる。

上記のＷｉｔｈ文は，甲１１－１及び第３準備書面別紙４の１２頁に記載されているものであるところ，１３頁から１８頁にも同様にＷｉｔｈ文が用いられている。

したがって，本件プログラム４のＡＤ変換器の制御のソースコードだけでも，記載する箇所やソースコードの記載の方法について複数の選択肢があり，ソースコードの表現順序，組み合わせ，表現自体に選択の幅がある。

また，原告は，本件プログラム４において，ＡＤ変換器としてコンテック社のアナログ入力ボードを用いているところ，当該アナログ入力ボードに搭載されたトリガー機能やメモリー機能のプログラムを用いず，トリガー機能やメモリー機能について以下の独自のプログラムを作成している。（甲１１－１及び第３準備書面別紙４　１５頁６行乃至１６頁３９行参照）。

＜原告が作成したソースコード＞

Public Function AnalogInputGetOneData(ByRef rData() As Double) As Long

```
Dim mRet As Long, mErrFlag As Boolean
Dim i As Integer, j As Integer, mTopRow As Integer, mRow As Integer
Dim mGetAiSamplingCount As Long, mHeadRow As Integer
Dim m 風速 As Double, m 風速 Ave As Double, m 風向 As Double, m 風向 No As
    Integer
Dim mVoltAve As Double, mVoltCentre As Double
If gAiRunFlag = False Then Exit Function
AnalogInputGetStatus
If pvAiSamplingCount < 1 Then AnalogInputGetOneData = 0: Exit Function
mGetAiSamplingCount = 1
'変換データ取得
mRet = AioGetAiSamplingDataEx(pvAioId, mGetAiSamplingCount,
    pvAiData(0))
If DispAioMessage(mRet, "AioGetAiSamplingDataEx") = False Then mErrFlag =
    True
For i = LBound(rData) To UBound(rData)
  If i >= LBound(pvAiData) And i <= UBound(pvAiData) Then
    rData(i) = pvAiData(i)
  End If
Next
AnalogInputGetOneData = mGetAiSamplingCount
With fMainForm
'   .txtTotalSamplingCount = Str(Val(.txtTotalSamplingCount) +
    mGetAiSamplingCount)
  .acxInputVolt(0).Value = rData(0)
```

```
.acxInputVolt(1).Value = rData(1)
g 風速 Ave.PushData rData(0)
mVoltAve = g 風速 Ave.GetAveData
mVoltCentre = g 風速 Ave.GetCentreData
.txtDataW = Format(rData(0), "##0.0000")
.txtDataWA = Format(mVoltAve, "##0.0000")
.txtDataWC = Format(mVoltCentre, "##0.0000")
m 風速 Ave = CompAiData(0, mVoltCentre)
m 風速 = CompAiData(0, rData(0))
m 風向 = CompAiData(1, rData(1))
m 風向 No = ConvDegToDirNo(m 風向)
.txtAD 瞬間風速 = Format(m 風速, "####0.00")
.txtAD 瞬間風向 Num = Format(m 風向, "####0.00")
.txtAD 瞬間風向 Str = GetDirStr(m 風向 No)
With .gridAiData
  mHeadRow = .FixedRows + .FrozenRows
  mRow = pvDataPos + mHeadRow - 1
  If fMainForm.chkViewScanData.Value = 1 Then
    mTopRow = pvDataPos - 10
    mTopRow = IIf(mTopRow >= .Rows, .Rows - 1, mTopRow)
    mTopRow = IIf(mTopRow < mHeadRow, mHeadRow, mTopRow)
    .TopRow = mTopRow
  End If
  .Cell(flexcpBackColor, mHeadRow, 1, .Rows - 1, .Cols - 1) = vbWhite
  .Cell(flexcpBackColor, mRow, 1, mRow, .Cols - 1) = RGB(255, 200, 200)
```

```
        .TextMatrix(mRow, 1) = Str2(m風速Ave)
        .TextMatrix(mRow, 2) = Str2(m風向)
        .TextMatrix(mRow, 3) = Str2(m風向No)
        .TextMatrix(mRow, 4) = GetDirStr(m風向No)
        .TextMatrix(mRow, 5) = Str2(rData(0))
        .TextMatrix(mRow, 6) = Str2(mVoltAve)
        .TextMatrix(mRow, 7) = Str2(mVoltCentre)
        .TextMatrix(mRow, 8) = Str2(rData(1))
      End With
      CompGrid_AiData
      .txtAD平均風速 = Format(Val(.gridAiData.TextMatrix(3, 1)), "####0.00")
      .txtAD最大風速10min = Format(Val(.gridAiData.TextMatrix(4, 1)),
          "####0.00")
      .txtAD最多風向10min.Text = GetDirStr(.gridAiData.TextMatrix(4, 3))
      .txtAD最多風向10min.Tag = .gridAiData.TextMatrix(4, 3)
      pvDataPos = (pvDataPos Mod 2400) + 1
      If mErrFlag = True Then
        ListBox_StatusAdd .lstAioStatus, "変換データ取得 : 異常終了"
      End If
    End With
End Function
```

原告は，プログラマーとしての経験に基づき，アナログ入力ボードに搭載されているもの以上の機能を実現するために上記のとおり独自のソースコードを作成していることから，原告以外のプ

ログラマーが作成しても上記ソースコードのような記載にはならない。そして，上記ソースコードによって，本件プログラム４は，高速サンプリング，連続サンプリング，トリガ収録，プリトリガ収録等の複雑な処理を実現しており，原告の個性が表れている。

ウ　②風観測データを回収処理するサブルーチンについて

データを回収して記録する方法の選択肢として，一つは以下のような配列を使用する方法が考えられる。

＜選択肢１＞

配列を使用したプログラム例

```
Dim a 風速 Ave(2400) as Double, a 風向(2400) as Double, a 風向 No (2400) as
        Double,aData(2400)asDouble,aVoltAve(2400)asDouble,aVoltCentre)(240
        0) as Double
a 風速 Ave(PosNo)= m風速Ave
a 風向(PosNo)= a 風向
a 風向 No (PosNo)= m風向No
aData (PosNo)= rData(0)
aVoltAve(PosNo)= mVoltAve
aVoltCentre (PosNo)= mVoltCentre
```

このソースコードは，配列メモリにデータを代入して、後にループを使用した演算処理を行わせるものである。

また，上記以外にも何を重視するか（処理スピード，使用メモリ等）によって，ソースコードの記載方法に多様な選択肢があるところ，本件では変換速度が４Ｈｚと低速であること，１０分間データ処理が対象で，データ数が４Ｈｚ×６００（秒）＝２４０

０個と大量ではないことから，原告は収録データの見える化を重視して，以下のグリッドコントロールを利用する方法を選択してソースコードを作成している（甲１１－１　１２頁９行目～１２頁４２行目，１０５頁１６行目～１０６頁４行目）。

＜原告が作成したソースコード＞

甲１１－１　１２頁９行目～１２頁４２行目

```
Dim mRet2 As Long, mErrorString As String, buf As String
DispAioMessage = True
If mRet <> 0 Then
mRet2 = AioGetErrorString(mRet, mErrorString)
buf = mMessage & " = " & mRet & " : " & mErrorString
ListBox_StatusAdd fMainForm.lstAioStatus, buf
DispAioMessage = False
End If
End Function
Public Function AnalogInputInitialize() As Boolean
Dim mRet As Long, mErrFlag As Boolean
Dim AiMaxChannels As Integer, i As Integer, j As Integer
With fMainForm
  '初期化処理
  mRet = AioInit(.txtAioDeviceName, pvAioId)
  If DispAioMessage(mRet, "AioInit") = False Then mErrFlag = True
  For i = 0 To AiInpChN - 1: For j = 0 To AiAveDataN - 1: pvData(i, j) =
  NullData:
  Next: Next
```

```
    'デバイスのリセット
     mRet = AioResetDevice(pvAioId)
     If DispAioMessage(mRet, "AioResetDevice") = False Then mErrFlag = True
      '最大チャネル数の取得
      mRet = AioGetAiMaxChannels(pvAioId, AiMaxChannels)
      If DispAioMessage(mRet, "AioGetAiMaxChannels") = False Then mErrFlag =
True
      If AiMaxChannels = 0 Then
        ListBox_StatusAdd .lstAioStatus, "アナログ入力の機能を持っていません"
        mErrFlag = True
      End If
      Debug.Print "AioGetAiMaxChannels" + Str(AiMaxChannels)
      ListBox_StatusAdd .lstAioStatus, "初期化 : " + IIf(mErrFlag = False, "正常終
了", "異常終了")
    End With
End Function

105行16行目～106行4行目
Public Sub CompGrid_AiData()
Dim r1&, r2&, c1&
Dim i As Long, mDir(15) As Integer, mDirNo As Double, mMaxCount As Integer,
mMaxDirNo
As Integer
With fMainForm.gridAiData
r1 = .FixedRows + .FrozenRows
```

```
r2 = .Rows - 1
c1 = 1
.TextMatrix(1, c1) = .Aggregate(flexSTCount, r1, c1, r2, c1)
.TextMatrix(2, c1) = .Aggregate(flexSTSum, r1, c1, r2, c1)
.TextMatrix(3, c1) = .Aggregate(flexSTAverage, r1, c1, r2, c1)
.TextMatrix(4, c1) = .Aggregate(flexSTMax, r1, c1, r2, c1)
For i = 0 To 15: mDir(i) = 0: Next
For i = r1 To r2
mDirNo = Val2(.TextMatrix(i, 3))
If mDirNo <> NullData Then mDir(mDirNo) = mDir(mDirNo) + 1
Next
mMaxDirNo = -1
mMaxCount = 0
For i = 0 To 15
If mMaxCount < mDir(i) Then
mMaxCount = mDir(i)
mMaxDirNo = i
End If
Next
.TextMatrix(3, 3) = mMaxCount
.TextMatrix(4, 3) = mMaxDirNo
End With
End Sub
Public Function GridSetData() As Boolean
Dim i As Integer
```

With fMainForm

With .grigSetting

上記のようにグリッドコントロールを使用してのデータ管理は，演算速度に問題が生じる可能性もあるが，原告はプログラマーとしての経験から，測定スペックやマシンスペック等も考慮した上で上記のソースコードを作成し，演算速度に問題を生じさせない形でデータの回収，記憶等の複雑な処理を実現している。加えて，ソースコードが大量であり，その組み合わせも複雑であることから，原告の個性が発揮されていると考えられる。

エ　③回収処理された風観測データを演算及び収録するルーチン

風観測は，平均風速を演算する必要があり，対象期間に一つの平均風速を演算するのではなく，常にその平均風速が必要となるため，移動平均処理を行っている。また，演算したデータを収録するために，データベース処理とテキストファイル処理を行っている。

移動平均処理等については，一般的な方法をとった場合，以下のように，配列を使用する選択肢が考えられる。

＜選択肢１＞

Dim aData(11) as Double

aData(Pos)=mData

for i=0 to 11 : mTotal= mTotal+aData(i):Next

mAve= mTotal/12

また，最適化の点を重視して，以下のような方法を選択するこ

とも考えられる。以下のソースコードは，移動平均演算をクラスとして定義することで，処理のカプセル化を可能とし，オブジェクト指向に沿ったプログラミングを実現するものである。

＜選択肢２＞

```
Private pvData(11) As Double, pvDataN As Integer, pvPos As Integer, pvTotal As
Double
Private Sub Class_Initialize()
Dim i As Integer
pvDataN = 0
For i = 0 To 11: pvData(i) = 0: Next
End Sub
Public Sub PushData(ByVal vData As Double)
pvData(pvPos) = vData
pvTotal = pvTotal - pvData((pvPos + 1) Mod 12)
pvTotal = pvTotal + pvData(pvPos)
If pvDataN < 12 Then pvDataN = pvDataN + 1
pvPos = (pvPos + 1) Mod 12
End Sub
Public Function GetAveData() As Double
If pvDataN > 0 Then
GetAveData = pvTotal / pvDataN
Else
GetAveData = 0
End If
End Function
```

以上のような選択肢の中，原告は，開発ポリシーである最適化重視の理由から，甲１１－１及び第３準備書面別紙４　２５頁，２７頁乃至３４頁，３６頁乃至３８頁のソースコードを作成した。具体的には，移動平均処理についてのソースコードは２５頁に記載があるものであり，データの収録についてのソースコードは２７頁乃至３４頁に記載があるものであり，テキストファイル出力処理についてのソースコードは３６頁乃至３８頁に記載があるものである。

上記のソースコードは，最適化の観点から原告が経験を踏まえて作成したものであり，他の者が作成しても同じものにはならないと考えられることから原告の個性が発揮されているといえる。

オ　小括

以上より，本件プログラム４は，各ルーチンについて，多様な選択肢の中からそれぞれのソースコードが選択されており，プログラム全体に選択肢の幅があることは明らかである。加えて，イ乃至オで上述したとおり，ソースコードは原告の独自の考えから作成されており，他者が作成しても同様のソースコードとなることは考え難い。

よって，本件プログラム４に著作物性が認められる。

以上