平成２１年１２月２１日　判決言渡

平成２１年（行ケ）第１０１４３号　審決取消請求事件

平成２１年１０月２８日　口頭弁論終結

判　　決

原　　告　　X

被　　告　　特　許　庁　長　官

指定代理人　　丸　山　高　政

同　　江　嶋　清　仁

同　　岩　崎　伸　二

同　　小　林　和　男

主　　文

１　原告の請求を棄却する。

２　訴訟費用は原告の負担とする。

事　実　及　び　理　由

第１　請求

特許庁が不服２００６－６８５１号事件について平成２１年４月２０日にした審決を取り消す。

第２　争いのない事実

１　特許庁における手続の経緯

原告は，平成１４年１１月２８日，発明の名称を「コンピュータにおける文字型データを使用しての数値計算」とする発明について，特許出願をした（特願２００２－３８２７７９号，以下「本願」といい，その明細書を「本願明細書」という。請求項の数は１である。甲１）。

原告は，平成１８年２月３日付けで拒絶査定を受けたので（甲４），同年３月１５日，これに対する不服の審判請求をした（不服２００６－６８５１号，審判請求書は同年３月１３日付けである。甲５）。特許庁は，平成２１年４月

２０日，「本件審判の請求は，成り立たない。」との審決をし，その謄本は，同年５月１９日，原告に送達された。

２　特許請求の範囲

本願明細書の特許請求の範囲の請求項１の記載は次のとおりである。

人が数値として解釈できる，コンピュータ内部で保持する文字列データを，そのまま計算対象の数値として扱い演算する手段。（以下，この発明を「本願発明」という。）

３　審決の理由

(1)　別紙審決書写しのとおりである。要するに，本願発明は，本願出願前に日本国内において頒布された刊行物である「文字列による無限長演算ユニットの製作秘話」（Delphi マガジン Vol.15，ＰＳネットワーク，２００１年（平成１３年）3月１日発行，２４ないし３７ページ。以下「刊行物１」という。甲１０，乙１）記載の発明（以下「刊行物１記載発明」という。）に基づいて当業者が容易に発明をすることができたものであるから，特許法２９条２項の規定により特許を受けることができないとするものである。

(2)　審決が，本願発明に進歩性がないとの結論を導く過程において認定した刊行物１記載発明，本願発明と刊行物１記載発明の一致点，相違点は，次のとおりである。

ア　刊行物１記載発明

数値を string 型の TIntStr 型で保持することで，桁数制限のない四則演算をおこなう，Windows 上の Delphi で実現された，文字列による無限長演算ユニットであって，

string 型の TIntStr 同士の足し算は，一番下の位から，各桁の数値を足して, 繰り上がりの数があれば覚えておいて, 次の桁に足すことを繰り返し, string 型の result に文字列を追加するものであって，

TIntStr 型のＮ１のｉ桁，Ｎ２のｉ桁，及び Carry の和を integer 型の

calcResult に格納し，

calcResult の mod 10 の演算結果を string 型の result に追加するものである，ユニット。

イ　一致点

「人が数値として解釈できる，コンピュータ内部で保持する文字列データを，計算対象の数値として扱い演算する手段。」である点。

ウ　相違点

文字列データの演算が，本願発明では，文字列データを「そのまま」計算対象の数値として扱うのに対して，刊行物１記載発明では，そのようなものか明らかではない点。

第３　取消事由に関する原告の主張

審決は，次に述べるとおり，本願発明の認定の誤り（取消事由１），本願発明と刊行物１記載発明との相違点の看過（取消事由２），顕著な作用効果の看過（取消事由３），本願発明と刊行物２（特開平２－４７７８７号公報，甲１１，以下「刊行物２」といい，そこに記載された発明を「刊行物２記載発明」という。）記載発明との相違点の看過（取消事由４）があるから，違法として取り消されるべきである。

１　本願発明の認定の誤り（取消事由１）

平成２０年１２月１２日付け拒絶理由通知書（甲６）において刊行物１が示されたため，原告は，刊行物１に反論するために平成２１年２月１６日付け意見書（受付日は同月１７日，甲７）及び「桁の大きい数値を計算するシステムマニュアル」（甲１２）を提出した。しかし，審決は，本願発明を認定するに当たって，甲７，甲１２も参酌すべきであったにもかかわらず，甲７，甲１２を参酌することなく本願発明を認定した誤りがある。

２　本願発明と刊行物１記載発明との相違点の看過（取消事由２）

(1)　データ型に関する相違点の看過

刊行物１に「普通の『桁』の概念と，string 型の index は一致しません」（乙１，２６頁）との記載があることから，刊行物１記載発明は，数字の桁の概念と文字列の概念が一致しておらず，既存の文字列とは異なる新たなデータ型又は規格を必要とする。そのため，本願発明が文字列型という既存のデータ型を計算に用いており，数字の桁の概念と文字列の概念とを一致させているのに対し，刊行物１記載発明は新たなデータ型又は規格を必要とする点で相違するにもかかわらず，審決は，この相違点を看過しているという誤りがある。

(2)　プログラム言語に関する相違点の看過

本願発明は特定のプログラム言語に依存しないように配慮しているのに対し，刊行物１記載発明は Delphi という特定のプログラム環境で動作するものである点で相違しており，審決は，この相違点を看過しているという誤りがある。

３　顕著な作用効果の看過（取消事由３）

(1)　桁落ちに関する作用効果の看過

刊行物１の「TIntStr 型と Integer との変換」の節（乙１，２７頁）に記載されているように，刊行物１記載発明は，既存のデータ型が扱える範囲の数値しか扱っていないから，桁落ちを解決するという効果を奏し得ない。これに対し，本願発明は文字列数値をそのまま計算対象とすることによって，桁落ちを解決するという顕著な効果を奏するが，審決は，このような本願発明の顕著な作用効果を看過した点で誤りがある。

(2)　相互変換のための関数を作成する必要性等に関する作用効果の看過

本願発明は，既存のデータ型である文字列型の数値をそのまま計算に用いているので，文字列型と整数型及び実数型との相互変換のための関数や型変換プログラムを別途作成する必要がないという効果，計算の入出力を文字列として行えるという効果，新たなデータ型や規格を国際標準として採用させ

る手間が不必要であるという効果，及び様々なコンピュータで利用ができるという効果を奏するものであるが，審決は，これらの顕著な作用効果を看過している点で誤りがある。

4　本願発明と刊行物２記載発明との相違点の看過（取消事由４）

本願発明は桁落ちを防ぐために文字列型のまま計算をするものであるのに対し，刊行物２記載発明は文字列型のまま計算をするものではない点において相違するにもかかわらず，審決は，同相違点を看過している点で誤りがある。

第４　被告の反論

審決の認定，判断に誤りはなく，原告主張の取消事由は，いずれも理由がない。

1　本願発明の認定の誤り（取消事由１）に対し

原告の平成２１年２月１６日付け意見書（甲７）における「式を解釈する機能が幹で，計算を行なう機能は枝葉です。」（甲７，４頁４ないし５行）との主張は，本願明細書の記載と整合しない。特許請求の範囲の請求項１は，実数の処理についての限定はなく，また，本願明細書に実数を処理することの具体的な記述はないから，甲７における「実数を処理する際に小数点の存在をどう扱うのか。」（甲７，１頁２５行）との内容は，本願明細書には記載されていない。また，甲１２の内容も，本願明細書には記載されていない。したがって，本願発明を認定するに当たり甲７，甲１２を参酌すべきでない。

2　本願発明と刊行物１記載発明との相違点の看過（取消事由２）に対し

(1)　データ型に関する相違点の看過に対し

刊行物１記載発明は，数値のデータとして，既存の string 型と同じ TIntStr 型という文字列型を用いているから，新たなデータ型又は規格を必要としない。刊行物１の「普通の『桁』の概念と，string 型の index は一致しません」（乙１，２６頁）との記載は，数字の文字列の左から何番目かということと何桁目かということが一致しないことを記述したものであり，刊行物１記載

発明が新たなデータ型又は規格を必要とすることを記述したものではない。したがって，刊行物１記載発明が新たなデータ型又は規格を必要とする点で本願発明と相違することを前提とする原告の主張は失当である。

(2)　プログラム言語に関する相違点の看過に対し

本願発明は特定のプログラム言語に依存するような限定はないのに対し，刊行物１記載発明は，Delphi という特定のプログラム環境の Object Pascal というプログラム言語で記述されているから，刊行物１記載発明は，本願発明の下位概念に該当し，本願発明は刊行物１記載発明を含む発明である。したがって，審決には，本願発明が特定のプログラム言語に依存しないように配慮しているのに対し，刊行物１記載発明が Delphi という特定のプログラム環境で動作するものである点を相違点としなかったことについて誤りはない。

３　顕著な作用効果の看過（取消事由３）に対し

(1)　桁落ちに関する作用効果の看過に対し

刊行物１には，刊行物１記載発明が常に大きな桁の数値を扱わないということは記載されておらず，刊行物１記載発明は，桁数の制限すなわち桁落ちの問題を解決するために文字列型を用いて数値を保持するものであるから，刊行物１記載発明は，本願発明と同様に桁落ちを解決するという効果を奏する。したがって，刊行物１記載発明が桁落ちを解決するという作用効果を奏しないことを前提とする原告の主張は，失当である。

(2)　相互変換のための関数を作成する必要性等に関する作用効果の看過に対し

刊行物１記載発明は，文字列型と整数型との相互変換のための関数として Delphi の標準関数を用いているから，刊行物１発明においても，文字列型と整数型との相互変換のための関数は別途作成する必要はない。

また，刊行物１記載発明は，string 型で保持した数値をそのままの形で出

力することを想定している。

さらに，刊行物１には，言語仕様が公開され周知となっている Delphi の Object Pascal というプログラム言語で組まれたプログラムリストが記載され，そのプログラムリストの文字列型は，Object Pascal の標準データ型である string 型と同一の TIntStr 型であるから，刊行物１記載発明を実施するためには，当業者は，そのプログラムリストを読めば足り，アルゴリズムや特殊なデータ型は必要としない。そのプログラムリストを読めば，アルゴリズムを理解することができ，そのアルゴリズムを種々の言語でプログラムし直せば，様々なコンピュータで利用ができる。

したがって，既存のデータ型である文字列型を用いているので，文字列型と整数型及び実数型との相互変換のための関数を別途作成する必要がないという効果，計算の入出力を文字列として行えるという効果，新たなデータ型や規格を国際標準として採用させる手間が不必要であるという効果，及び様々なコンピュータで利用ができるという効果は，いずれも刊行物１記載発明から予測可能な程度のものであり顕著な作用効果ではない。

４　本願発明と刊行物２記載発明との相違点の看過（取消事由４）に対し

審決は，仮に本願発明が数式の解釈を包含するものであるとして，数式の解釈の部分について進歩性がないことを示すために，刊行物２を挙げたものであるから，審決の判断に誤りはない。

第５　当裁判所の判断

１　本願発明の認定の誤り（取消事由１）について

原告は，平成２０年１２月１２日付け拒絶理由通知書（甲６）において公知文献として刊行物１が示されたため，刊行物１に反論するために甲７，甲１２を提出したが，審決は，本願発明の内容を認定するに当たって，甲７，甲１２も参酌すべきであったにもかかわらず，甲７，甲１２を参酌することなく本願発明を認定した誤りがあると主張する。

しかし，原告の上記主張は，採用することができない。その理由は，以下のとおりである。

(1)　平成２０年１２月１２日付け拒絶理由通知書（甲６）において公知文献として刊行物１が示され，これに対し，原告は，平成２１年２月１６日付け意見書（甲７）及びその添付資料として甲１２を提出した。

以下，甲７，甲１２の記載内容について検討する。

ア　本願明細書の特許請求の範囲には，実数を演算する場合に小数点位置を整合（桁合わせ）する必要があるとの記載はなく，また，本願明細書の発明の詳細な説明にも，小数点位置を整合（桁合わせ）させて実数を処理することについて具体的な記述はない。

この点について，甲７の「①実数の扱いについて」（甲７，２ないし３頁）との項には，実数を扱う場合，小数点位置を整合（桁合わせ）する必要があるとの記載があるものの，「本願明細書に例１から例４までの仕様を具体的に記述することは，同業者を利する結果となるため，避けなければなりません。」（甲７，３頁７ないし９行）と記載され，本願明細書にそのような具体的な記述が存在しないことが明確に示されている。

したがって，甲７の上記記載部分は，本願発明の内容を理解する上で何らかの意味を有するものとはいえない。

イ　本願明細書の「発明が解決しようとする課題」の欄には，次のとおりの記載がある。

「ハードウェアによる演算は高速かつ，正確な内容が期待できるが，その数値演算のためのハードウェアの設計段階で数値の桁数に制限が設けられるため，桁数の大きな演算を行う場合に桁落ちが生じる。」（【０００４】）

「コンピュータ内で，計算の対象となる数値データ，もしくは計算途中に算出された数値データに一度桁落ちが生じると，その計算結果における数値の信頼性は大きく損なわれる。」（【０００５】）

「本発明は大きな桁を扱う数値データで，四則演算のうち加算，減算，乗算においては桁落ちの発生しない手法を示す事，除算においては任意の算出桁数を指定することにより，納得のいく計算結果を示す事を最終的な目的としている。」（【０００６】）

「例えば，『９７＋３４』という文字列の記述をコンピュータに与えれば，『９７』と『３４』は数値として解釈が可能であり，間にある『＋』演算子は左右に並ぶ２つの数値を加算するための記号と解釈可能であり，また式そのものの意味を理解するアルゴリズムは既に存在する。」（【００１４】）

「式のデータの解釈とは，『１２×（３４＋５６）』の文字列を，３４と５６を先に加算して，その演算結果と１２を乗算する等，計算の優先順位や手順を式から読み取ることで，この技術は上記にもあるとおり，既存している。」（【００１７】）

上記の本願明細書の記載によれば，本願明細書には，本願発明の課題として，大きな桁を扱う数値データで，四則演算のうち加算，減算，乗算においては桁落ちの発生しない手法を示すこと，除算においては任意の算出桁数を指定することにより，納得のいく計算結果を示すことが記載されており，計算を行うことが本願発明の課題であることが認められる。他方，式のデータの解釈は，周知であることが記載されており，本願明細書の他の箇所を参酌しても，式を解釈することについて具体的な記載はない（なお，本願明細書において，「桁落ち」とは，計算に用いようとする数値の桁数が，コンピュータの数値計算において扱うことのできる数値の桁数を超えることを意味すると解される。）。

これに対して，甲７の「②式の解釈について」（甲７，３ないし４頁）との項には，「本願発明の意図とする全体のレイアウトは，式を解釈する機能が幹で，計算を行なう機能は枝葉」（甲７，４頁４ないし５行）であり，それによって本願発明は実用性を有する旨記載されている。

したがって，甲７の上記記載部分は，本願明細書の記載と整合性がなく，本願発明の内容を理解する上で何らかの意味を有するものとはいえない。

ウ　甲７の実質的内容は，前記アの「①実数の扱いについて」（甲７，２ないし３頁）との項及び前記イの「②式の解釈について」（甲７，３ないし４頁）との項に記載されているものと認められ，甲７に，その他に本願発明の内容を理解する上で意味のある事項が記載されているとは認められない。

また，甲１２は，甲７の添付資料として提出されたものであり，その記載内容は，甲７に述べられたのと同じ事項を説明したものにすぎず，以上の判断を左右するものとはならない。

(2)　以上のとおりであるから，審決が，甲７，甲１２を参酌せずに本願発明を認定した点に誤りはない。

２　本願発明と刊行物１記載発明との相違点の看過（取消事由２）について

(1)　データ型に関する相違点について

原告は，刊行物１に「普通の『桁』の概念と，string 型の index は一致しません」（乙１，２６頁）との記載があることから，刊行物１記載発明は，数字の桁の概念と文字列の概念が一致しておらず，既存の文字列とは異なる新たなデータ型又は規格を必要とするとの前提に立った上で，本願発明は文字列型という既存のデータ型を計算に用いており，数字の桁の概念と文字列の概念とを一致させているのに対し，刊行物１記載発明は新たなデータ型又は規格を必要とする点で相違するにもかかわらず，審決は，この相違点を看過している点で誤りがあると主張する。

しかし原告の上記主張は，採用することができない。その理由は，以下のとおりである。

ア(ア) ａ　「プログラマのための Delphi ２入門」（小山裕徳訳　学校法人東京電機大学　１９９７年（平成９年）３月２０日第１版１刷発行，乙

２）には，次のとおりの記載がある。

「Delphi は Microsoft　Windows アプリケーションを開発するための環境であり，ツールです。」（「訳者まえがき」の頁１２ないし１３行）

「Delphi のプログラミング言語は，Object　Pascal が基本になっています。」（５頁２４行）

「Delphi の両方のバージョンとも，変数の型としておよそ１５の標準型があります。Delphi　2.0 には，（数え方にもよりますが）さらに7つの型があります。通常は，Delphi の型をいくつかに分類して考えます。論理型，文字型，整数型，実数型などです（次の章で説明するように，あなた自身で型を定義することもできます)。以下に示すのは，Delphi の両方のバージョンで共通な型について概略を説明したものです。」（１１８頁２８行ないし１１９頁2行）

「string　string（文字列）型の変数は，Delphi　1.0 では２５５字までの ASCII 文字を保持します。一方，Delphi　2.0 では文法オプションの［長い文字列を使う］がオンになっていて，このデフォルトを変更しないかぎり文字数に基本的に制限がないことを意味します（実際の制限は，２Ｇバイトです)。どちらのバージョンでも，数字を大かっこ[]でかこって，２５５字より少ない文字を保持するようにすることができます。たとえば，次のようにします。

var

　　Name: string[45];」（１２０頁３０行ないし１２１頁2行）

ｂ　プログラマのための入門書である乙２に前記ａの記載があることからすると, string 型が Delphi のプログラミング言語である Object Pascal の変数の型の一種であり，テキストを扱う文字列型であることは，プログラミングの技術分野の当業者であれば一般的な知識として有していたものと認められる。

(イ)　また，刊行物１（乙１）には，次のとおりの記載がある。

ａ「どうせならば，桁数制限なしに，そのまま四則演算のできる数値型を作ることはできないものなのでしょうか？　桁数が増えれば，そのぶんだけ自動的に長さが延長されてメモリが確保される型, というと，Delphi では，string 型が思い当たります。

数値を string で保持しておけば，桁数がどれほど増えようとおそれることはありませんし（＊４）, label や Edit に表示することも容易です。

　というような発想でもって，文字列による無限長演算ユニットの製作はスタートしました。」（２５頁「◇そういうわけで，無限桁数演算の可能性を考える」の節）

ｂ「データの保持については string 型と決めていますが，string そのままでは味気ないので，TIntStr 型という型を定義しました。

type

　TIntStr = string;

つまり string 型そのままですが，こうしておくことで，将来何か変更するときに他のソースまで書き換える手間を省くことができます。」（２６頁「◇一般的な定義」の節」）

(ウ)　前記(ア)の当業者の一般的知識と上記(イ)ａ，ｂの刊行物１の記載によれば，刊行物１記載発明は，Delphi のプログラミング言語である Object　Pascal で記述され，string 型を数値データの保持に用い，データ型として使用するものであることが認められる。

イ(ア)ａ　さらに，「[改訂版]Delphi 入門」（青山学著　ソフトバンク株式会社１９９６年（平成８年）８月２４日初版発行，乙３）には，次のとおりの記載がある。

「type 文は，新しい型を宣言するための予約語です。type 文を使えば，任意のデータ型を宣言することができます。

◇ type 文の構文

type

型名 = 宣言する型;

たとえば，２０文字の文字列型を TNamae という型として新たに登録したいときには，次のように宣言します。

type

TNamae = String[20];

type 文で型を宣言すると，それを新しい型として，Integer 型や Real 型などの他の組み込み型とまったく同様に，変数の宣言をすることができます。次に，その例を示しておきましょう。

var

AoyamaHikari: TNamae;

この例で AoyamaHikari は，TNamae 型，つまり先ほど type 文で宣言した String[20]という型で宣言したことになります。」（５４頁１２ないし２７行）

ｂ　プログラマのための入門書である乙３に上記の記載があることからすると，type が，Delphi においてプログラムで使用する変数に対して用いられる新しいデータ型を宣言するための予約語であることは，プログラミングの技術分野の当業者であれば一般的な知識として有していたものと認められる。

（イ）　前記(ア)の当業者の一般的知識と前記ア(イ)ｂの刊行物１の記載によれば，刊行物１記載発明では，新たな型として TIntStr 型を宣言しているが，TIntStr 型は string 型と同じ型として宣言しており，TIntStr 型と string 型が実質的に同じであることは，当業者の一般的知識に照らして明らかである。

ウ　原告は，刊行物１に「普通の『桁』の概念と，string 型の index は一致

しません」（乙１，２６頁）との記載があることから，刊行物１記載発明は，数字の桁の概念と文字列の概念が一致しておらず，既存の文字列とは異なる新たなデータ型又は規格を必要とすると主張するが，「普通の『桁』の概念と，string 型の index は一致しません」との記載を根拠として，刊行物１記載発明は，数字の桁の概念と文字列の概念が一致しておらず，既存の文字列とは異なる新たなデータ型又は規格を必要とすると解することはできない。

(ア)　すなわち，刊行物１（乙１）の２６頁には次のとおりの記載がある。

「TIntStr 型は string 型そのままですので，Intstr[3]という具合にしてアクセスすれば，特定の桁の数字をとってくることができます。しかしながら，図２を見ていただければ分かるように，普通の『桁』の概念と，string 型の index は一致しません。

そこで，次のような関数を用意しました。」（「◇桁の数値を取り出す」の節）

図２には，左より１から７までの七つの数字が横書きされ，右から三つ目の「５」が「３桁目の数字」として指示され，左から三つ目の「３」が「Intstr[3]」として指示されている。

(イ)　前記(ア)の刊行物１の記載によれば，刊行物１記載発明において，ある数値から特定の桁の数字を取り出す場合，Intstr[　]（string 型の index）は，数字列の左から数えて１文字目を Intstr[1]，２文字目を Intstr[2]，３文字目を Intstr[3]・・・のように扱うものであり，文字数がｎ個の数字列では，普通の位取りでいう１桁目，すなわち１の位の数字は，左から数えてｎ文字目であるから，Intstr[1]ではなく Intstr[n]に当たる。このように，普通の位取りでいう桁と，左から数えて何文字目にあるかということ（Intstr[　]，string 型の index）は一致せず，このことを，刊行物１では，「普通の『桁』の概念と，string 型の index は一致しません」

と記述したものと認められる。

したがって，刊行物１の「普通の『桁』の概念と，string 型の index は一致しません」との記載を根拠として，刊行物１記載発明は，数字の桁の概念と文字列の概念が一致しておらず，既存の文字列とは異なる新たなデータ型又は規格を必要とすると解することはできない。

エ　このように, 刊行物１記載発明は, 既存の string 型と実質的に同じ TIntStr 型の文字列型をデータ型として計算に用いており，新たなデータ型又は規格を必要とするものではない。そうすると，本願発明は文字列型という既存のデータ型を計算に用いており，数字の桁の概念と文字列の概念とを一致させているのに対し，刊行物１記載発明は新たなデータ型又は規格を必要とする点で相違するとは認められず，審決に，この相違点を看過しているという誤りがあるとの原告の主張は，採用することができない。

(2)　プログラム言語に関する相違点について

原告は，本願発明は特定のプログラム言語に依存しないように配慮しているのに対し，刊行物１記載発明は Delphi という特定のプログラム環境で動作するものである点で相違しているが，審決は，この相違点を看過しているという誤りがあると主張する。

しかし，原告の上記主張は，採用することができない。その理由は，以下のとおりである。

すなわち，本願発明は，もともと特定のプログラム言語に依存することのない，「人が数値として解釈できる，コンピュータ内部で保持する文字列データを，そのまま計算対象の数値として扱い演算する手段。」との技術思想であり，いかなるプログラム言語によるものであっても，上記の技術思想を実現するものであれば，本願発明と同一の発明と認められ，採用されているプログラム言語が何かということは，本願発明の技術思想の実現の有無とは直接の関係がないものと解される。そうすると，審決が，本願発明と刊行物

１記載発明の対比において，本願発明は特定のプログラム言語に依存しないように配慮しているのに対し，刊行物１記載発明は Delphi という特定のプログラム環境で動作するものである点を相違点として挙げなかった点に誤りはない。

3　顕著な作用効果の看過（取消事由３）について

(1)　桁落ちに関する作用効果の看過について

原告は，刊行物１の「TIntStr 型と Integer との変換」の節（乙１，２７頁）に記載されているように，刊行物１記載発明は，既存のデータ型が扱える範囲の数値しか扱っていないから，桁落ちを解決するという効果を奏し得ないとした上で, 本願発明は文字列数値をそのまま計算対象とすることによって, 桁落ちを解決するという顕著な効果を奏するが，刊行物１記載発明はそのような効果を奏し得ず，審決は，このような本願発明の顕著な作用効果を看過した点で誤りがあると主張する。

しかし，原告の上記主張は，採用することができない。その理由は，以下のとおりである。(なお,「桁落ちを解決する」とは，コンピュータの数値計算において扱うことのできる数値の桁数に制限がないようにすることを意味すると解される。)

ア(ア)　刊行物１の「TIntStr 型と Integer との変換」の節には，次のとおりの記載がある。

「TIntStr 型は独自に計算ルーチンを持っていて，独立して数値を保持しますが，そうはいっても既存の型とやりとりができないと役にたちません。

Int64 の数値を TIntStr に代入したり，TIntStr 型の数値を Int64 として返す関数を作成しました。

当然，お互い桁あふれが生じない範囲であることが前提です。

Int64 は Integer と代入互換性がありますので, 普通の integer の数値も，

この関数で TIntStr に変換することができます。」（乙１，２７頁）

（イ） 前記(ア)の記載によれば，そのうちの「当然，お互い桁あふれが生じない範囲であることが前提です。」との記載は，変換元又は変換先の数値が桁あふれを生じる場合には正しくデータ型の変換ができないことについて注意を喚起しているものと認められ，刊行物１記載発明が大きな桁数の数値を常に扱わないとの趣旨の記載であるとは認められない。

イ(ア) また，刊行物１には，次のとおりの記載がある。

「このような大きな数値の演算は，科学技術計算の分野では頻繁に出てきており，この場合の数値の扱いについても「多倍長計算」というアルゴリズムが確立されています。つまり，Integer で足りなければ Integer をふたつもってきて 64bit で数値を表そう，それで足りなければ４つで 128bit，８つで 256bit...という具合にして，integer を複数持ってくることによって演算を行うわけです。

この手法は，桁上がりの部分だけをチェックすれば残りの部分は普通の計算で処理することができるため，計算効率に優れており，多くの分野で利用されています。しかし，実際に使用するには，特殊な数値の型を利用する必要があったり，計算できる桁数に制限があったりして，普通の思考でそのまま計算するには，何かと不便な点が多くあります。」（「◇パソコンにおける，アップル社購入の計算」の節，乙１，２５頁）

（イ） 刊行物１の前記(ア)の記載及び前記２(1)ア(イ)ａの記載によれば，刊行物１記載発明は，大きな数値の演算には，桁数の制限があったり，特殊な数値の型を利用する必要があるとの問題点があったので，これらの問題点を解決するために，桁数の制限なしに普通の思考でそのまま四則計算できる数値型として文字列型（Delphi の string 型）を用いるという発想のもとに製作されたことが認められる。

そうすると，刊行物１記載発明は，文字列型（Delphi の string 型）を

用いることにより，桁落ちを解決する，すなわちコンピュータの数値計算において扱うことのできる数値の桁数に制限がないようにするとの作用効果を奏するものと認められる。

ウ　このように，刊行物１記載発明は，文字列型（Delphi の string 型）を用いることにより，桁落ちを解決する，すなわちコンピュータの数値計算において扱うことのできる数値の桁数に制限がないようにするとの作用効果を奏するから，本願発明が，文字列数値をそのまま計算対象とすることによって桁落ちを解決するという効果を奏するとしても，それは，顕著な作用効果であるとはいえない。

(2)　相互変換のための関数を作成する必要性等に関する作用効果の看過について

原告は，本願発明は，既存のデータ型である文字列型をそのまま計算に用いているので，文字列型と整数型及び実数型との相互変換のための関数や型変換プログラムを別途作成する必要がないという効果，計算の入出力を文字列として行えるという効果，新たなデータ型や規格を国際標準として採用させる手間が不必要であるという効果及び様々なコンピュータで利用ができるという効果を奏するものであるが，審決は，これらの顕著な作用効果を看過している点で誤りがあると主張する。

しかし，原告の上記主張は，採用することができない。その理由は，以下のとおりである。

ア　文字列型と整数型及び実数型との相互変換のための関数や型変換プログラムを別途作成する必要がないという効果について

(ア)　文字列型と整数型及び実数型との相互変換のための関数や型変換プログラムを別途作成する必要がないという点は，本願発明の作用効果ということができないのみならず，仮にそのような効果があると解する余地があるとしても，その効果を顕著な作用効果とみることはできない。

まず, 以下のとおり, 特許請求の範囲及び本願明細書の記載によれば, 文字列型と整数型及び実数型との相互変換のための関数や型変換プログラムを別途作成する必要がないという効果は，本願発明によって必然的にもたらされる作用効果とはいえない。

すなわち，本願明細書には，数値のデータ型に関連して，「このときの計算手段は，それぞれ縦１桁の数値だけを対象として行うため，文字型から整数型へ変換して計算し，必要な結果内容を文字型に変換して④のエリアに置いていく方法も考えられる。」（【００２９】）との記載，及びＩＥＥＥ規格等の数値データから文字列データに変換すること（【００３３】ないし【００３５】）の記載はあるものの，文字列型と整数型及び実数型との相互変換のための手段についての記載はない。そうすると，本願発明は，文字列型と整数型及び実数型との相互変換のための関数や型変換プログラムを別途作成する技術を排除しているとは解されず，そのことからすると，文字列型と整数型及び実数型との相互変換のための関数や型変換プログラムを作成する必要がないという効果は，本願発明の作用効果と解することはできない。

（イ） のみならず，仮に，文字列型と整数型及び実数型との相互変換のための関数や型変換プログラムを別途作成する必要がないという効果があったとしても，同効果は，以下のとおり，刊行物１記載発明から予測可能な程度のものにすぎず，顕著な作用効果とはいえない。

すなわち，刊行物１には，次のとおりの記載がある。

「TIntStr 型は string 型そのままですので，Intstr[3]という具合にしてアクセスすれば，特定の桁の数字をとってくることができます。しかしながら, 図２を見ていただければ分かるように, 普通の『桁』の概念と, string 型の index は一致しません。

そこで，次のような関数を用意しました。

getdigit は, TIntStr 型の数値から Keta 桁目の数値をとってくる関数です。返り値は Tdigit となっていますが，これは

```
Tdigit = 0..9;
```

と定義されています。

```
function getdigit(N:TIntStr;Keta:integer):Tdigit;
begin
  if (Keta > length(N))or(Keta<1) then result := 0
  else result := StrToIntDef(Copy(N,length(N)-Keta+1,1),0);
end;
```

」（乙１，２６頁「◇桁の数値を取り出す」の節）

「符号の問題をここで片づけてしまうことにより，IntStrAddPositive, IntStrDecPositive といった関数の中では，引数が正の数であると想定して処理を行うことができます。

符号の問題を片づけてしまえば，あとは小学校一年生で習った足し算を忠実に実行して行くだけです。

一番下の位から，各桁の数値を足して，繰り上がりの数があれば覚えておいて，次の桁に足す，この繰り返しです。

result は TIntStr（=stirng 型）（判決注：「(=string 型)」の誤りと認められる。）ですので，result := InttoStr(calcResult mod 10) + result;は足し算ではなくて文字列の追加になるところに注意してください。

```
function IntStrAddPositive(N1, N2:TIntStr):TIntStr; {N1+N2}
var
  i,calcResult,N1Length,N2Length:integer;
  Carry:Tdigit;
begin
  result := '';
  N1Length := length(N1);
```

```
    N2Length := length(N2);
    Carry := 0;

    i:=1;
    while(i<=N1Length)or(i<=N2Length)do
    begin
      calcResult := getdigit(N1,i)+getdigit(N2,i) + Carry;
      Carry := calcResult div 10;
      result := InttoStr(calcResult mod 10) + result;
      Inc(i);
    end;
    if Carry > 0 then result := Inttostr(Carry) + result;
  end;」（乙１，２８ないし２９頁「◇足し算」の節）
```

（ウ） 前記(イ)の記載によれば，刊行物１記載発明においては，関数 IntStrAddPositive(N1, N2:TIntStr)が TIntStr 型の二つの引数 N1 及び N2 を取り，N1 及び N2 はともに正の整数であることが想定されており，N1 を被加数，N2 を加数として加算を行い，加算の結果を TIntStr 型で返すものであることが認められ，N1 と N2 の加算は，小さい桁から１桁ずつ順番に行われ，１桁の加算を行う際には，その桁の数字を一度整数型のデータに変換して加算してから，結果を文字列型に再変換していることが認められる。

そして，「プログラマのための Delphi ２入門」（乙２，１３６頁）によれば，刊行物１記載発明の上記の演算の過程で用いられている Delphi の関数について，関数 StrToIntDef（乙２，１３６頁の表の機能欄の「StringToIntDef」との表記は，「StrToIntDef」を意味するものと認められる。）は，整数を表現する文字列を，その数値（整数型データ）に変

換し，文字列が正しい数字の表現でないときにはデフォルトの値にする関数であり，IntToStr は，整数を文字列に変換する関数であることが認められ，そのため，刊行物１記載発明は，文字列型と整数型の相互変換のための関数として Delphi に標準的に備えられた関数を用いていることが認められる。

そうすると，刊行物１記載発明は，既存のデータ型である文字列型（Delphi の string 型）を用いており，文字列型と整数型及び実数型との相互変換のための関数や型変換プログラムを別途作成する必要はないものと認められる。したがって，文字列型と整数型及び実数型との相互変換のための関数や型変換プログラムを別途作成する必要がないという効果は，刊行物１記載発明から予測可能な程度のものにすぎず，顕著な作用効果とはいえない。

イ　計算の入出力を文字列として行えるという効果について

刊行物１には，「数値を string で保持しておけば，桁数がどれほど増えようとおそれることはありませんし（＊４），label や Edit に表示することも容易です。」（乙１，２５頁「◇そういうわけで，無限桁数演算の可能性を考える」の節）と記載されている。

そして，「Delphi プログラミング入門」（玉木彰著　２００１年（平成１３年）４月５日初版第１刷発行，乙４）には，「Label コンポーネント」及び「Edit コンポーネント」が，文字列（string）を表示出力するためのものであることが記載されている（乙４，３０ないし３１頁）。

そうすると，刊行物１記載発明は，string 型で保持した数値をそのままの形で出力することを想定しているものであると認められる。したがって，計算の入出力を文字列として行えるという効果は，顕著なものではなく，刊行物１記載発明から予測可能な程度のものにすぎない。

ウ　新たなデータ型や規格を国際標準として採用させる手間が不必要である

という効果及び様々なコンピュータで利用ができるという効果について

刊行物１には，前記のとおり，文字列型を用いた無限長の数値演算について，Delphi の Object Pascal というプログラム言語で組まれたプログラムリストが記載されている。そして，乙２ないし５によれば，Delphi の Object Pascal の言語仕様は公開され，周知である。さらに，前記のとおり，刊行物１記載発明のプログラムリストの文字列型は，Object Pascal の標準データ型である string 型と同一型の TIntStr 型である。

そうすると，当業者は，刊行物１記載のプログラムリストを読むことにより，刊行物１記載発明のアルゴリズムを理解し，その実施ができるものであって，そのためにアルゴリズム及び特殊なデータ型を必要としない。

また，刊行物１記載のプログラムリストを読むことにより，刊行物１記載発明のアルゴリズムを理解することができることから，そのアルゴリズムを種々のプログラム言語でプログラムし直すことは，当業者であれば適宜になし得る設計事項であり，それによって刊行物１記載発明を様々なコンピュータで利用することもできる。そのため，刊行物１記載発明の利用を広めるために，新たなデータ型や規格を国際標準として採用させる必要もない。

したがって，新たなデータ型や規格を国際標準として採用させる手間が不必要であるという効果及び様々なコンピュータで利用ができるという効果は，顕著なものではなく，刊行物１記載発明から予測可能な程度のものにすぎない。

エ　したがって，本願発明が，文字列型と整数型及び実数型との相互変換のための関数や型変換プログラムを別途作成する必要がないという効果，計算の入出力を文字列として行えるという効果，新たなデータ型や規格を国際標準として採用させる手間が不必要であるという効果及び様々なコンピュータで利用ができるという効果を奏するものであるとしても，これらは

顕著な作用効果であるということはできず，審決に，これらの顕著な作用効果を看過したとの誤りはない。

4　本願発明と刊行物２記載発明との相違点の看過（取消事由４）について

原告は，本願発明は桁落ちを防ぐために文字列型のまま計算をするものであるのに対し，刊行物２記載発明は文字列型のまま計算をするものではない点において相違するにもかかわらず，審決は，同相違点を看過している点で誤りがあると主張する。

しかし，原告の上記主張は，採用することができない。

審決は，念のため，本願発明と刊行物１記載発明とが「人からコンピュータに与える式のデータを解釈するプログラム」を備えているか否かとの点において相違しているとしても，同相違点に係る事項は，刊行物１記載発明に刊行物２記載発明を適用することによって，当業者が容易に想到し得ると判断した。すなわち，刊行物２記載発明は，本願発明と刊行物１記載発明との間の前記相違点に係る事項が容易に想到し得ることを示すために用いられたものであるから，本願発明と刊行物２記載発明の相違点を認定しなかったからといって，何ら審決の判断に違法を来すことはない。したがって，この点の原告の主張は，主張自体失当である。

5　結論

以上のとおり，原告主張の取消事由はいずれも理由がない。原告は，その他縷々主張するが，審決にこれを取り消すべきその他の違法もない。

よって，原告の本訴請求を棄却することとし，主文のとおり判決する。

知的財産高等裁判所第３部

裁判長裁判官　飯　村　敏　明

裁判官　中　平　　健

裁判官　上　田　洋　幸