この文書は13ページあります

易しくない パソコン リテラシー(3/4)
インタフェース編

3. パソコンとの付き合い(ユーザインタフェース)

3.1 インタフェースの解説

3.1.1 インタフェースと環境

環境は、漢和辞典にありますので、古くからある熟語のようです。「或るものを取り囲む外界」と抽象的な説明があります。英語の environment の訳語として使われるようになりました。人から見るとき、大きく、**自然環境**と**社会環境**との二つに分けます。人(ユーザ)がパソコンを利用する場合の付き合い方がユーザインタフェースです。擬人化したパソコンとの付き合い方ですので、社会環境が構成されている、とみます。インタフェース(interface)とは、化学で使い始めた新しい用語です。漢字は**界面**と当てました。直訳すれば、**顔合わせ**、または**面合わせ**です。界面活性剤の用語があります。パソコンの環境で使うインタフェースは、三つの区分を考えます。第一は、電気・電子装置をコネクタやコードで接続することの用語に使われ、詳しくはハードウエアインタフェースと言います。接続装置をバス(bus)と言うことがあります。乗り合いバスが人をまとめて運ぶことに例えて、データをまとめて運ぶ用語にしたようです。1993 から、**USB**(universal serial bus)の利用が普及してきました。第二に、ソフトウエアインタフェースの区分があります。無体物としてのデータを受け渡しする方式を指します。直列・並列インタフェースでの信号の性質を言うときが一例です。そして第三がユーザインタフェースです。人とパソコンとの付き合い方を言う用語です。最初は、マン・マシン・インタフェース、ヒューマンインタフェースとも言いましたが、性差(gender)のない用語のユーザインタフェースに落ち着きました。人(ユーザ)からパソコンへの話し掛けは、キーボードからの文字入力です。これに、**ポインティングデバイス**として、**マウス**が加わるようになりました。指でナゾル方法も開発されました。一方、パソコン側からは、モニタに、画像出力で返すことが基本です。文字も含めます。人は眼で見て確認します。ブザーやベルを鳴らすこと、その発展として、音声で知らせることも応用されてきました。モニタは具体的な有体物ですが、モニタに表示されている方は一過性の無体物です。電源を切れば、何も残りません。モニタ上の画像は、列車の窓枠(ウインドウ)から景色を眺めることに似ていて、現場に在るカメラを介して間接的に外界を観察しています。この外界を仮想(virtual)の**世界**と言います。Windows システムは、パソコンのモニタに「仮想の世界を仮想のカメラを介して見た画像を表示する」とモデル化する考え方を採用した OS です。**統合開発環境**(**IDE**; Integrated Development Environment)の下での作業を言うようになりました。この用語は、マイクロソフト社が、Windows の OS の下でのプログラミングをサポートする目的で、1997 年に使い始めてから、一般的になりました。

3.1.2 マルチメディアを楽しむ

音楽を楽しむための装置は、1877 年のエジソンに始まりました。グラモフォン(蓄音機)と言いました。絵画や写真を含め、現代では、パソコンを利用する楽しみ方は、事務処理や数値計算などの実用的な利用とは独立して、デジタル技術を応用するマルチメディアとして扱うようになりました。モニタ上での画像の鑑賞は、静的なインタフェースです。動画、また、音楽の楽しみ方は、或る継続時間を取る動的なインタフェースです。どちらも、ユーザ側の態度は、受動型です。積極的な参加型のインタフェースの一つがゲームです。テレビゲームを作りたい、人工知能を持たせたいロボットを作りたい、などがそうです。プログラミングは、参加型のインタフェースです。音楽を参加型で楽しむ場合、単に歌う場面と、楽器、つまり道具(ハードウエア)を使う場面とがあります。持ち運びができる楽器と、据え置きで使う楽器との区別もあります。パソコンを利用する音楽制作、つまり、プラグラミングは、DTM(Desk Top Music)の頭字語で言い、1980 年代から始まっていました。楽譜を書いて作曲や演奏に使うプロ級のツールには、アップル社のマッキントッシュ(1984)が多く利用されました。普通のパソコンでも、映像・音声なども含めて、マルチメディア処理と総称する利用方法がサポートされています。スポーツなど、パソコンから離れて別の趣味を楽しみたいとき、パソコンに仕事を任せることも必要です。デモンストレーション、また、広告などに利用するときは、繰り返しの実行が工夫されています。一回だけの利用では「①OS の管理下で利用できるコマンドの入力による方法、②拡張子(*.exe)の付いたアプリケーションを呼ぶこと、③拡張子(*.bat)の付いたバッチ処理による方法」があります。システムは、拡張子(*.exe) (*.bat)の付いたファイルを、どちらも、実行形式のファイルと理解してくれます。

3.1.3 疑似デバイスを操作するプログラミング

パソコンの OS が Windows の方式に変ったことで、パソコンの概念が革命的に変りました。それまでのコンピュータは、電気・電子的な機械装置(マシン)を扱う、という感覚で接していました。これに対して、Windows のモニタ画面は、装置や部品の図柄を表示しておいて、ユーザはそれらを模擬的(シミュレーション)に扱う方式になりました。Windows がどのようなプログラミングをサポートするかの例には、電卓が使われました。個別部品の図柄を**コンポーネント**と言います。演算子(+ − ＊ /),数字キー(0～9)などのボタン、表示窓などの図柄を総称的に言う用語がオブジェクト(object)です。英語の意味は**物**です。英文法では**目的語**と言います。機能を定義する**事**の方を**コントロール**と言います。プログラミングでは、まず、電卓の図形をデザインします。ボタンなどの図柄(疑似的な物)にカーソルを合わせてクリックすると、何をするかの**処理**(事の方)が機能します。**メソッド**と言う用語が使われます。ユーザは、キーワードを決めて、自作のプログラムを追加することもできます。この全体(図形と処理)を総称した英語が**オブジェク**トです。オブジェクト指向プログラミングとは、オブジェクトを**利用する**プログラム作成の意味です。図柄とメソッド、つまり、物と事との二つを含めたプログラミングの意味です。

3.1.4 キーボードが基本的なデバイス

キーボードから文字を選択してタイピングすることは面倒な作業です。好ましくない付き合い方になる例を二つ挙げます。第一は、会計業務などにコンピュータを利用するときの、データタイピング作業でした。初期のメインフレーム利用の時代、カードや紙テープに鑽孔する作業に当たる技能職をキーパンチャーと呼びました、パンチング装置は、機械式のキーボードでしたので、指先の力仕事でした。職業病として、鍵症炎が問題になりました。電気接点に替えたキーボードは指の負担が軽くなります。しかし、接点の構造に問題が起きました。微妙な指先のタッチ感覚を持たせることが必要ですが、耐久性も重要です。NEC のマイクロコンピュータ PC-8001(1979)は、コンピュータゲームを楽しむツールとしても人気商品になりました。その使い方に、できるだけ早い時間間隔でキーを打つことが競われるゲームがあって、特定のキーが早く壊れることが起こりました。

第二は、モニタを見る場面が増えて、近視が進み、眼底疲労などの健康問題が起こることです。パソコンを、業務だけでなく、趣味にも利用するときは、ほどほどの使い方ができるように、パソコンに仕事を任せ、パソコンから離れる時間を持てるインタフェースを考えます。企業では、その管理方法の工夫が必要です。個人の場合は、自分が責任を持つ自覚が必要です。近年では、スマホでのゲーム遊びが問題になっています。なお、スマホは、英語 Smartphone のカタカタ用語です。モバイル(持ち運びができる)携帯電話電話の総称であって、1998 年から普及が始まりました。簡易な OS を内蔵していて、多機能化が図られるようになりました。大型画面を使う製品をタブレットと言い、義務教育の機材として検討されています。ゲーム機としての利用が人気のようです。

3.1.5 ゲームパッドを使うインタフェース

ミニコンの PC－8001 は、ゲームも楽しめることでも人気商品でした。発売価格は、当時¥168000 でしたので、子供用のツールとしては高価です。親が買い与えることには抵抗があります。この解決が、ゲーム専用にミニコンを応用し、モニタに家庭用テレビを使う任天堂の**ファミコン**(ファミリーコンピュータ)であって、1983 年以降、テレビゲームのブームを巻き起こしました。徹底的な低価格化が図られ、PC-8001 の 1/10 の価格でした。インタフェースには、**ゲームパッド**がポインティングデバイスとしての標準になりました。プログラムは ROM を内蔵したカセットから読み込ませます。

図 3.1 任天堂のファミコン(1983)

3.1.6 マウスを使うインタフェース

パソコンは、英数字の入力用のキーボードが必須のハードウエアです。大きさは、持ち運びを考えても A4 版までの横幅です。電子辞書では、ハガキ大の英字キーボードを指先で押す使い方をします。操作性は犠牲になります。**マウス**(mouse)でクリックするインタフェースは、子供でもパソコンを利用できます。形がしっぽ付きのネズミに似ていることから付けられた愛称です。1968 年、エンゲルバードが開発しました。マッキントッシュ用はボタン一つ、マイクロソフト社はボタン二つ、の違いがあります。電池で駆動できるノート型のパソコンでは、マウスを外付けのデバイスとして使いますが、**タッチパッド**を指でナゾる方法をサポートするようになりました。これは、画面を汚すことと繋がる使い方ですので、電気・電子・機械装置に接する技能職は、使いたくないと感じる方式です。また、手をかざすと反応して、パソコンが誤作動をすることがあります。

3.2 道具としての人

3.2.1 人の脳も記録媒体である

人の脳も記録媒体と考えることができます。非常に特殊な機能が利用されますので、自動化ができないインタフェースを構成します。人の脳にデータ入出力をするデバイスの代表がコンソールでした。これとの交流がユーザインタフェースです。FORTRAN で作成したプログラムは、実行中に、ユーザの判断を問い合わせするメッセージ出力文と、ユーザがそれに答えて何かのデータ入力するため入力文とがあります。このときの機番は、6と5が決められています。コンソールは、見掛け上、一台で入出力をするデバイスであるため、機番を＊で使うこともします。人の脳は、大量のデータを、正確に記憶しておくことができません。メモ出力を補助に使います。COBOL は、定形的な処理が多いこともあって、処理の実行中にユーザが関わることがなく、ユーザインタフェースはありません。パソコンでは、多くのプログラムが利用できるようになりました。そのため、逆に、使い方が分からなくなることが起こります。HELP を参照する使い方がありますが、HELP の使い方の HELP が必要になる皮肉な場面も起こります。単純な機能に絞ったソフトウエアが使い易いのです。ユーザインタフェースにグラフィックスの利用が普通になり、それが文書の編集にも応用されるようになりました。単純な方から言うと、**メモ帳**・**ワードパッド**・**ワードプロセッサ**の順に使い分けます。メニューのデザインは、この順で複雑になっています。

3.2.2 人の眼の不思議な機能

人の眼は、見ているようでも、視線から外れると、よく見たことにならないことを、先の第 1.2.9 項で紹介しました。ここでは微妙な色違いを感じる機能について説明します。人の眼の視神経は3系統です。ここに、先天的または後天的に障害があるとき、色盲と言います。差別用語とされるため、**色覚異常**の用語が使われます。色違いの点の集合を見せて、数字の見え方で色覚異常の検査ができます。色は、三原色を重ね合わせることで、人の眼は多様な色遣いを区別しています。これは、脳の中で、言わば、ごまかされている認識です。**点描画**（てんびょうが）と言う絵画の技法があります。混ぜないままの**油絵の具**を小さな点単位で描きます。隙間が空き、下の白地が見えるように残ります。結果として、明るい画像を描くことができます。風景画に向いた画法です。フランスの画家 **ジョルジュ・スーラ**（1859−1891）が始めた作画技法とされ、新印象派の創設者とされています。点の集合で描きます。ただし手作業で生の絵の具を擦り付ける技法ですので、画像の精細度で言えば、25DPI 程度です。やや離れ見ると精細な色遣いの画像に見えます。錯視の一種です。

スーラが作画に使った絵の具の色種は、わずか１１種だったと言われています。現代のカラー発色法では、3 原色を混合するだけです。それと較べると、色種は多いのですが、当時は画期的な技法でした。スーラは 31 歳の若さで亡くなりましたので、点描画法は普及しませんでした。近年、この作画法が注目されるようになりました。その背景には、カラープリンタやカラーモニタを、たったの三種類の原色ピクセルの集合で表示できることに応用されているからです。図 3.2 は、スーラの代表作品です。元の油絵は 208×308cm の大きさであって、実物ならば点描の画法が確認できます。この図は、背景に橋がありますので、参考のために、インターネットから採図しました。

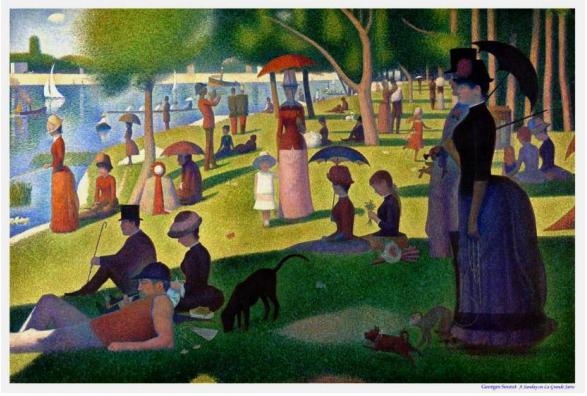

図 3.2 グランド・ジャット島（パリ近郊セーヌ河）の日曜日の午後（スーラ）、シカゴ美術館、

3.2.3 数式はグラフィックス扱いをする

科学技術を扱う出版社は、数式の**組版**（マークアップ）を扱う関係で、普通の出版社とは違う独特の存在意義があります。算術は歴史が古いこともあって、算術式は、数字並びを書いて、読み上げる方法もあります。代数式は、数の代わりに文字や記号を使う方法です。文書にする、また、それを読み上げることに文法がありません。絵、つまり、グラフィックスとして見る使い方をしています。現代風に言えば、オブジェクトです。幾つかの図形部品を幾何学的に並べた構造になるからです。鉛の活字を使って数式を組み上げていた時代の組版は、特種な部品や、特別な字形の活字を使います。特殊な技能を持った植字工が作業に当たりました。JIS では、数学記号を規格化しています。これは、数式の組版に必要だからでした。しかし、数式の原稿を書く科学技術者からは、印刷の品質について校正の段階で不満も多く出ました。パソコンを利用する時代になって、一般的な文書の組版言語が開発されてきましたが、数式の扱いは難しい課題でした。また、数値計算の手順を説明する言語もないのです。数式を読むことは、一種の読図ですが、不思議な理解をしています。

3.2.4 数式編集言語

活版印刷の時代、代数式の組版は、数式の意味が分からなくても、特殊な技能をもった植字工（人）が作業をしました。ドナルドクヌース「Donald Knuth（1938−）」は、数式を含む文書を自分のパソコンで編集できる組版言語 TeX を 1978 年に発表しました。思っているような組版表現にならないことに不満を持ったことが TeX の開発動機です。特種な記号や文字は、図形部品扱いをして、幾何学的な位置を指定して、並べる作業ですので、**画像処理**です。ただし、完全な機械化・自動化はできません。オブジェクト指向プログラミングで対話的(interactive)な作業（IDE）で作成します。作業結果は、画像データとしてファイルに保存できます。

一方、ワードプロセッサとして写真製版用の組版を作成する**文字処理**ソフトが LaTex です。TeX を利用しています。こちらは、書式を決めれば、或る程度の自動化ができます。1984 年、L.B.Lamport(1941−)が発表しました。科学技術系の論文作成の DTP として利用されています。MS−Word には、以前、簡単な数式エディタがありました。現在のバージョンでは使い方が特殊になってしまいました。

3.2.5 数式作図用言語

高校の数学で教えている初等数学・初等関数の簡易な作図用には、グラフ電卓があります。専門的な数値計算のプログラミング言語は、FORTRAN が代表です。これは、静的に数値を扱います。関数をグラフに描くのは動的インタフェースと考えることができます。そのツールは、別のサブルーチンライブラリを使います。1970 年代、図表示用のデバイスは、Calcomp 社の機械的なデバイスであるプロッタか、プリンタを備えた Tectronix 社の CRT モニタが利用されていました。後者用の線図描画用コマンド（move/draw）が普及しました。ISO による標準化言語の提案は 1977 年の GKS（Graphical Kernel System）です。非常に洗練された数値計算とグラフ化のツールは、スティーブン・ウルフラム（Stephen Wolfram; 1959−）の Mathematica(1988)があります。

3.2.6 ビジネスグラフ

事務処理関係では、MS−EXCEL は、最初、会計業務に利用する表計算ソフト（spread sheet software）として開発されました。印刷して利用したい需要がありますので、ワードプロセッサとしての使い方ができるようになりました。橋梁は、力学原理を応用して安全な設計であることを確認する計算が必要です。そのため、美術的な造形をするデザイナは、力学計算のできる技術者と共同作業をする必要があります。東京オリンピックの会場としての国立代々木競技場は、丹下健三が吊橋の構造をヒントにしましたが、構造計算者としての、坪井善勝とのペアで設計されたことは、海外では広く知られています。橋の場合、設計者は、設計図を描くことと平行して、計算書の作成をします。古い橋梁耐荷力の計算を EXCEL で再現設計することが見られるようになりました。EXCEL の書式仕様は、通信回線で文書の送受信に使う HTML 型式のファイルに利用されています。また、算術計算を始め、種々の数値計算の関数が利用できるように進化してきましたので、以前利用されていた Basic の利用に代わる使い方ができるようになりました。この中の組み込みソフトして、通称で言うビジネスグラフがサポートされています。表に構成されている数値データを、円グラフ・折れ線グラフ・棒グラフなどに作図させるツールです。こちらは、定型化が図られています。したがって、文科系・理科系の区別なく、使い易いアプリケーションになりました。

3.2.7 高速処理と雨だれ式処理

コンピュータは、高速の数値計算をさせることが目的で開発が進められてきました。メインフレームは、多種類の高速計算を、順序良く連続的に処理させる方法と、多くの人が同時に利用できる方法とを工夫しました。前者が**バッチ処理**(batch)、後者が**時分割処**理(**TSS**: time sharing system)です。パソコンは個人専用のコンピュータですので、パソコンから見れば、ポツンポツンと**雨だれ式**の処理です。電卓は、曲がりなりにもコンピュータですが、高速性能を生かす使い方ではありません。実行用プログラム単位があって、個別に実行させ、切れ目が目立たない連続処理をさせることをバッチ処理と言いす。ところが、本来のバッチ処理は、連続処理の対義語です。バッチは、パンを焼く一窯分の材料、と説明があります。小麦粉を練って、イースト菌を加え、発酵させるため、しばらく寝かせておく材料ですが容れ物の意味でも使います。建設関係では、コンクリートを練るミキサ作業単位分のセメント・砂利・砂・水を混ぜて練り、打ち込み場所まで運びます。或る時間経過で硬化が始まりますので、工業製品のような保存や再利用ができません。品質についても、言わばアナログ的であって、品質を揃えることに難しさがあります。この品質管理には、デジタル的な考えを応用して、種々の計量法が工夫されています。この方法は、実用主義（プラグマティズム）ですので、日本では、アメリカでの研究・開発に多くを学びました。生コンの製造は、機械化が進められ、工場の形態をとるようになりました。また、輸送と打ち込みとは、生コン車が利用されるようになりました。ただし、生コンは、或る時間が経過すると硬化が始まりますので、輸送時間を考えて、生コン工場は大規模化ではなく、地域単位に分散しています。

3.3 インタフェース用デバイス

3.3.1 装置依存と装置非依存

装置は、英語の device の訳語です。カタカナのデバイスの方が一般的に使われています。プログラミング言語は、OS に依存した文字処理言語として販売されています。IDE は、個別に扱うプログラム単位、または作業単位を扱います。全体をひとまとめにして、実行形式のプログラムに作成することをビルド(build)と言い、バッチ処理をさせます。特定の処理をさせることが目的の、単独に実行できる形式のプログラムを**ユーティリティ**と言います。システムの管理用、特に利用頻度の高いディスクとファイルの処理を**コマンド**と言い、それをまとめたプログラムが OS です。DOS は、disc operating system の頭字語です。UNIX では、コマンドの集合(容れ物)扱いですので、軟らかい(ソフト)が入る容れ物(ウエア)の意味を持った**シェル**(貝殻)であるとも言います。

ユーザのプログラムでは、デバイスは、外部の**周辺装置**(peripheral)の扱いをしていて、OSを介して間接的に利用します。キーボード・ディスク・プリンタは、基本的なデバイスです。ディスクがやや特殊でしたので、ディスク違いでユーザプログラムの書き換えの手間を無くす方法が、DOS の設計思想でした。テレタイプライタは、キーボードとプリンタとの複合装置であって、**標準入出力装置**と言いました。これも抽象化してディスク扱いをしました。装置違いは、番号などで区別しました。ディスク違いでも、文字の読み書きが主な利用でしたので、デバイスの相違をユーザのプログラミングで考える必要は、殆どありませんでした。この思想を**装置非依存**(device non independence)と言います。例外がグラフィックスの表示でした。

工業では、図面の作成(製図)は重要な作業です。図面は、**線図**(line drawing)で描き、**濃淡図**(painting)は使いません。カラー(色)違いで区別をすることは、原理的には濃淡図による作図です。青写真などによる複写は、カラーの再現ができません。また、線図では濃淡表現ができません。これに代わる方法が**ハッチング**です。テレビ放送のカラー受信機が普及してきた 1955 年頃から、パソコン専用に使う **CRT**(cathode ray tube)モニタが開発されるようになりました。グラフィックス装置は、原理違いで種々のデバイスがありますので、プログラムは**装置依存**(device dependence)です。装置に依存しない標準化の考え方が ISO で提案された GKS(graphical kernel system)です。ユーザは、疑似的なグラフィックス装置を使うプログラミングをすれば、OS が実装置を機能させるようにします。OS には、デバイス製作会社が提供するデバイスドライバを組み込みます。この考え方は、プリンタを始め、外部装置すべてのインタフェースに使われるようになりました。**USB**(ユニバーサル・シリアル・バス)のコネクタを持った外部装置が増えてきて、パソコン違いでも差し替えて利用できます。

3.3.2 巨大プログラムの管理

実行形式のプログラムコードは、CPU から見れば、外部装置に保存されています。実行時は、RAM に読み込んで処理速度を上げます。RAM のバイト容量は、システムが決めた、或る上限があります。OS も常駐しますので、残りの領域に入り切れない大きなプログラムコードの場合、ソフトウエア的に解決する方法が必要になります。16 ビットのパソコン時代、実行時のオブジェクトコードは(*.exe)ファイルのほかに、複数の(*.ovl)ファイルがあって、互いに独立した実行ができるようにプログラミングされ、同じメモリ領域に入れ替えて読み込んで実行させました。これを**オーバーレイ**と言います。ただし、(*.ovl)ファイルの寸法は、16 ビットの符号なし整数できまる 65KB 以内に抑える必要がありました。ファイルからの読み込みの時間は RAM の処理速度に較べて遅いので、プログラム全体の処理速度は下がります。パソコンが 32ビット、64ビットのプロセッサを採用するようになったことと、大容量の RAM が利用できるようになって、実用上は問題にならなくなりました。オーバーレイの考え方は、**DLL**(dynamic link library:ダイナミックリンクライブラリ)に引き継がれています。

もう二つの課題があります。大寸法のデータは、連続した配列にして読み書き(アクセス)します。プログラムのソースコードを、複数のテキストファイルに分けてコンパイルするとき、第一の課題は、共通に利用するデータの構造、第二がそのデータへのアクセス方法の設計です。初期の FORTRAN は、COMMON 文で共通領域を宣言し、型の異なるデータの集合を配列の形で利用する方法を採りました。C/C++言語では、構造体として、struct 文で宣言し、extern 文で参照するデータ並びを定義します。具体的には EXCEL の表形式の行方向のデータ構造を定義する場面を考えると理解できるでしょう。16 ビットのパソコンでは、論理的に一続きのデータ領域の寸法を 65KB 以内に抑える必要がありました。そのため、幾何モデリングでは、実用形状を表したい場面は限られ、複雑な形状を作成できませんでした。しかし、教育用ツールとしては、好評でした。パソコンの CPU が、32 ビット、64 ビットのレジスタを利用するようになって、寸法制限の方はあまり問題にならなくなりました。プログラミングの注意としては、宣言文は、定義文より前に書くことです。

3.4 流れ図による全体の把握

3.4.1 流れ図の特徴

ハードウエア・ソフトウエアは、静的な物です。インタフェースは、動的な性質も持ちます。流れ図(flow chart)は、動的な手順を図に表す方法です。その例として、説明用に作成した図 3.3 を見て下さい。線で表した流路は、プログラムの実行順とデータの流れが、上から下に流れるような経路を取ることを示すと同時に、扱うデータの処理手順も示します。四角の枠は、入り口が一つ、出口も一つと置ける処理単位を抽象化したものです。中身は複雑なサブプログラム単位も考えることができます。この単位に、英字名(A, B, …)を付けています。枠を結ぶ線には向きがあって、矢印がなければ、上から下に流れるとします。上に戻る逆流は、処理の繰り返しのときです。もし不注意な逆路があると無限ループになります。丸印と菱形の枠は、データ流れの離合集散の箇所を示します。菱形の箇所は番号を付けました。論理判断を使って分岐先を決めます。菱形を使う図 3.3 は、最大で3分岐しか表現できません。多分岐は、ツリー構造の場合があります。多分岐の例は、後の図 3.5 で説明します。

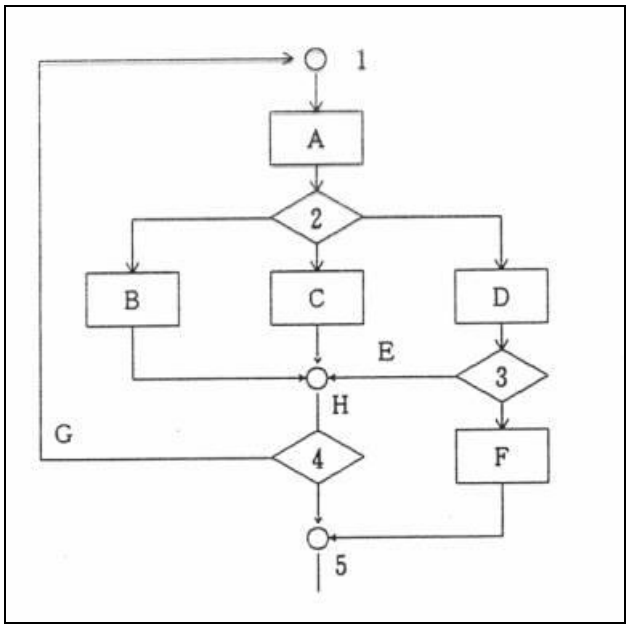

図 3.3 Flow Chart (流れ図)

3.4.2 インタラクティブな処理を考える場合

図 3.3 の流れ図が提案された頃のプログラミングは、バッチ処理が主体でした。入り口一つ、出口一つの処理単位をまとめ、分岐の判断を内部のデータで行わせます。これに対して、対話型(インタラクティブ)でコンピュータを使う場合には、丸印の箇所で処理を一時停止して、ユーザが介入して分岐先を決定して再開させます。判断のルールを図 3.3 と同じにすれば、同じ作業ができます。しかし、停止する度にユーザが指示していては鬱陶しいので、デフォルトの選択を提案しておいて、変更が必要でなければパスさせると能率がよくなります。この方法には種々の工夫があります。単純に改行キーを押す、または、タイマーを介して、ある時間内にユーザが指示しなければデフォルトで次の処理に移る、などの方法があります。このようにしておくと、教育目的やデモンストレーションをさせるときに便利です。

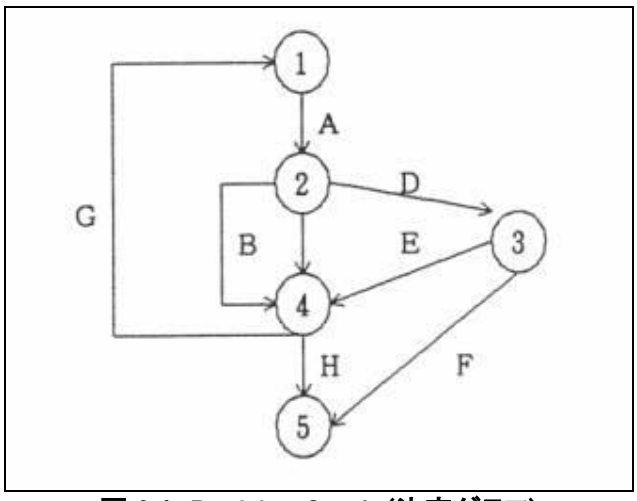

図 3.4 Decision Graph (決定グラフ)

3.4.3 繰り返し試行を組み込むことができる

図 3.4 の決定グラフの表現方法は、丸印の箇所でデータの確認をするだけでなく、必要があればデータを差し換えて再開させるように処理を追加することができます。通路は処理を伴う場合も含みますので、同じ場所から出て同じ場所に戻る道草ループを構成することもできます。これは、結果の一部をプリンタに書き出すことや、条件を変えて試行を繰り返すことを可能にしますので、設計計算のプログラミングに応用できます。この方法をコンピュータ側に立って考えてみると、処理を停止し、ユーザが介入する箇所が図 3.4 の丸印のように複数あっても、物理的には一ケ所であることです。DOS の環境では、それはキーボードだけでした。Windows の環境では、少し複雑ですが、キーボードの他にマウスを使うことができます。後で解説するプロトタイプのプログラムでは、同じ場所から出発して元の場所に戻るループ処理は、なるべくメニューの方で選択し、別の丸印に移動することを選択するときには、ステータスバーのパネル番号をクリックするように設計しました。この考え方をプログラミングするときには、次節で説明する選択分岐の手法を応用します。

3.4.4 通路をバラバラにして並列に繋ぎかえる

図 3.4 のグラフでは、丸印が複数点になるのですが、処理の流れを考えると、その場所はハードウエア的には、すべて同じ場所、キーボードです。したがって、これを一ケ所と考え、その場所から選択的に通路を選び、再び同じ場所に戻るような図に描き直したものが図 3.5 です。これは、図 3.4 の二つの丸印間の接続線をバラバラにして、個別にループを構成させます。これを**通路の選択グラフ**と(Path Selector)と呼ぶことにしました。この図は、例えば、左側にバス停車場があって、そこから並列的にバス路線が準備されているようなモデルです。路線はループになっていて、再バス停車場に戻ります。路線名に(A, B, …)の英字名が付けられています。どの路線を、どの順に回れば良いかをプログラムすれば、図 3.3、図 3.4 と同じ処理ができます。バスは一台しか考えていませんが、複数台のバスの運行を考えると、並列処理を表す図と見ることができます。Windows のメニューバーは、図 3.5 選択グラフであって、入れ子式にサブメニューが利用できます。CPU は一つですので、言わば一台しかバスが回ることになり、並列処理は疑似的な実行です。

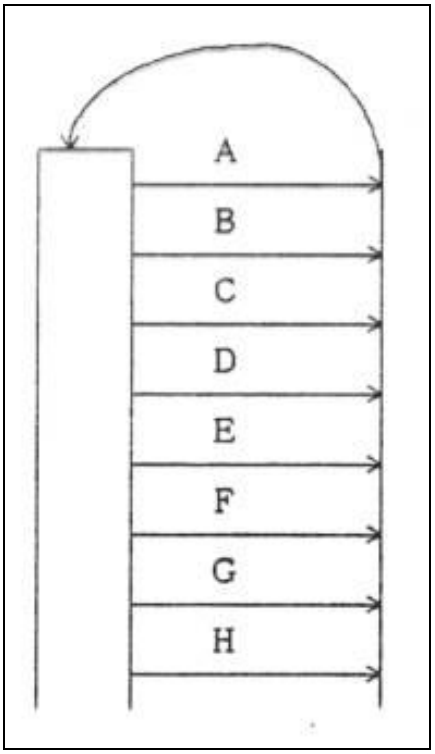

図 3.5 Path Selector (選択グラフ)

3.4.5 コマンド起動型プログラム

図 3.5 のような処理の組み立ては、コマンド起動型のプログラム(command driven program)を作成するときに応用されています。ユーザインタフェースの部分は、キーボードから、あらかじめ定義された何かのキーワードと、必要に応じて引き数の並びを入力すると、右側の処理のどれかを選択して実行し、再びユーザの次の入力を待ちます。したがって、図 3.5 の左の欄は、文字列解読のプログラムです。DOSの環境では、モニタ画面に案内表示に続けて、キーボードからの入力要請のプロンプトが表示されます。このとき、関連のあるデータを個別に要請することもあって、図 3.5 を入れ子構造に構成します。この制御方式が**対話型**(インタラクティブ)です。ユーザが、毎回キーボードから指示を与えるのが鬱陶しいときには、コマンド並びをテキストファイルに作成しておいて、キーボードの入力をファイル入力に切り替える方法があります。このテキストファイルが**バッチファイル**(batch file)です。また、対話形式でキーボードから入力した作業を、そっくりファイルに記録する方法を組み込むことがあります。このファイルを**ログファイル**(log file)と言います。ログファイルを編集すれば、バッチファイルが簡単に作成できます。

3.4.6 プログラミング文書

単に**プログラム**と言えば、小学生向けの国語辞書にも載るように、学芸会・運動会・音楽会などの、組み合わせや順序を書いたものです。したがって、コンピュータの実行手順を書いたものは、コンピュータ・プログラムと断ります。プログラミングと言うときは、コンピュータ・プログラムの作成作業を言う用語と解釈されます。プログラミングの英字綴りは、-ing を付けた動名詞ですので、動的な意義を持ちます。**プログラミング文書**は、テキスト形式にしたものを**ソースコード**、ファイルに保存すると、ソースファイルです。単に**プログラム文書**と言うときは、マニュアルなども含めた全体を総括的に言う用語です。

インタフェースは、プログラム文書が動的な実行段階で、人とどのように関わるか、の方に視点を向けます。これには、二つの区別があります。パソコンと直接に対話する**インタラクティブ**な実行方式と、**コンパイル**して間接的に実行させる方式とです。NEC の PC-8001 に搭載されていた N－BASIC は、この二つの方法を使うことができました。しかし、その後は、高級言語化が進められ、インタラクティブな方法が利用できなくなってしまいました。これは、パソコンの利用が広範囲になって、専門別に特化させた実行形式のプログラム(アプリケーション)の作成に進んできたことが一つの原因です。次節から、専門性の高いアプリケーションを作成するツールの幾つかを紹介します。予約語にどのようなスペルの語があるかのリストをまとめてあります。「予約語の数が少ない言語は易しく、多い言語は高級な処理ができますが難しい」と判断できるようにしました。

3.5 シェルと言う概念

ユーザがパソコンを機能させる方法は、文字並びの入力です。これには種々の名前で呼ばれています。総称はキーワード(または予約語)です。実行形式のプログラム名は、識別子(*.exe)が付きます。一方、コマンドと言うときは、OS または実行中のプログラムの中でユーザが動的に使う命令であって、引き数を持つ複数の語句の場合があります。プログラミングの文構造は、主に、他動詞を先行させた命令文の集合で書きます。引き数として、直接目的語、日本語では(…を)に当たる語が続きます。日本語は、動詞の活用に命令形があります。しかし、動詞は文末に使う SOV ですので、プログラミング言語には向かない文型です。

パソコンに電源を入れると、真っ先に実行するソフトウエアが ROM から読み出されて走ります。UNIX では、シェル(shell)と言いますが、他の OS では特に名前を付けた区別をしていません。シェルは、OS をROMから読み出して、RAM に書き込み、ユーザインタフェースの環境を構成します。したがって、読み出しに一分程度の時間が掛かります。シェルは軟らかい意味を持つソフトウエアの容れ物(貝殻)であるとする考え方から呼ぶ名称にしたようです。この節から、幾つかの OS とプログラミング言語で使われているキーワードの簡単な紹介をまとめました。

3.6 アセンブリ言語

最も原初的な文字並びでパソコンを機能させるコードが**機械語**であって、複数バイトの並びです。CPU の製作会社ごとに違いがあります。人にとっては全くの暗号ですので、憶えやすいように3字の英字並びを当てます。これをニーモニック(mnemonic)と言います。引き数などを加えて、プログラミング言語化したものを、**アセンブラ言語**と言います。一つの文例を挙げます;

MOV AL, 61h …… 意味の英語 : Load AL with 97 decimal (61hex)

MOV がニーモニック、AL は CPU の AL レジスタ名、61h は 10 進数 97 の 16 進数表示です。処理は、レジスタ AL に、引き数として使っている 97 のビット並びのバイトをコピーします。アセンブラ言語で書いたソースプログラム(テキストファイルです)を翻訳(コンパイル)して、機械語並びのファイル(バイナリファイル)に変換するソフトウエアがアセンブラです。何かの機械語でプログラムを調べたいとき、ファイルをダンプ(dump)して、16 進数並びと英数字並びのリストを作成することをします。このリストをニーモニック並びに翻訳することが**逆アセンブル**です。この解読作業は、一般的には**リバースエンジニアリング**(reverse engineering)と言います。知的財産権の侵害であるとの批判もある微妙な処理ですが、倫理的なことに注意して利用することは許されています。コンピュータウイルスの対策では、必須の研究手段です。

3.7 C/C++言語

アセンブラ言語で書いたプログラムは、人が処理内容を理解するには難しいので、**低級言語**とされています。FORTRAN や COBOL は、人が読んで理解できるように、機械語から作成した、専門環境に特化したプログラム言語であることから、**高級言語**であると言います。しかし、他の目的でプログラミングをしたいとき、CPUの機能を賢く利用できるアセンブラ言語の利用が勝ります。C言語は、機械語との相性がよい高級言語として、1972年に発表されました。言語構造のくせが強く、一般利用には使い難さがあります。C 言語は、OS のような管理用ソフトや別仕様のプログラミング言語の開発に向いた言語です。多目的の利用ができるますので、**汎用言語**と言うことがあります。対義語は**ドメイン固有言語**です。C++言語は、C 言語の拡張版です。C/C++言語は英字の大文字・小文字は別文字として使います。読み上げて伝える文書ではなく、眼で見て理解する文書であることも特徴の一つです。

予約語 (keyword) 英大文字を使うとエラーになります	**C 言語の予約語** auto break case char const continue default do double else enum extern float for goto if int long register return short signed sizeof static struct switch typedef union unsigned void volatile while **C++言語の予約語**:上記に加えて下に示すキーワードが追加されています。 asm bad_cast bad_typeid bool catch class const_cast delete dynamic_cast except explicit false finally friend inline mutable namespace new operator private protected public reinterpret_cast static_cast template this throw true try type_info typeid typename using virtual xalloc

ディレクティブは、13 語あります。

#define #elif #else #endif #error #if #ifdef #ifndef #import #include #line #pragma #undef

C言語の標準ライブラリとヘッダーを下にまとめます。定数名などは多く定義されています。C/C++言語が難解であることの理由の一つは、ライブラリの関数名がキーワードに準じるように使われることです。

診断	<assert.h>		assert
文字テスト	<ctype.h>	文字テスト	isalnum, isalpha, iscntrl, isdigit, isgraph, islower, isprint, ispunct, isspace, isupper, isxdigit
		文字変換	tolower, toupper
数学エラー定数	<errno.h>		
実数型サイズ	<float.h>	処理系依存の定数	浮動小数点演算に関係する定数
整数型サイズ	<limits.h>	処理系依存の定数	最大値、最小値などの値
ロケール関数	<locale.h>		localeconv, setlocal
数学関数	<math.h>		sin, cos, tan, asin, acos, atan, atan2, sinh, cosh, tanh, exp, log, log10, pow, sqrt, ceil, floor, fabs, ldexp, frexp, modf, fmod
非局所的ジャンプ	<setjmp.h>		setjmp, longjmp,
シグナル	<signal.h>		signal, raise,
可変引き数リスト	<stdarg.h>	マクロ	(va_list, va_start, va_arg, va_end)
	<stddef.h>		
入力と出力	<stdio.h>	ファイルの操作	clearerr, fopen, freopen, fflush, remove, fclose, rename, tmpfile, tmpnam, setvbuf, setbuf, feof
		書式付き出力	fprintf, printf, sprintf, vprintf, vfprintf, vsprintf
		書式付き入力	fscanf, scanf, sscanf,
		文字入出力	fgetc, fgets, fputc, fputs, getc, getchar, gets, putc, putchar, puts, ungetc,
		直接入出力	fread, fwrite,
		ファイル位置	fseek, ftell, rewind, fgetpos, fsetpos,
		エラー関数	ferror, perror
ユーティリティ	<stdlib.h>	型変換	atof, atoi, atol, strtod, strtol, strtoul,
		乱数,絶対値,剰余	rand, srand, abs, labs, div, ldiv,
		文字数計算	mblen, mbstowcs, mbtowc, wcstombs, wctomb
		記憶領域割り当て	calloc, malloc, realloc, free,
		システム制御	abort, exit, atexit, system, getenv,
		ソートなど	bsearch, qsort,
文字関数	<string.h>	str 関数	strcpy, strncpy, strcat, strncat, strcmp, strncmp, strchr, strrchr, strspn, strcspn, strpbrk, strstr, strlen, strerror, strcoll, strcspn, strtok, strxfrm
		mem 関数	memcpy, memmove, memcmp, memchr, memset
日付けと時刻	<time.h>		clock, time, difftime, mktime, asctime, ctime, gmtime, localtime, strftime

3.8 UNIX のコマンド

ディレクトリ操作系

cd	(ディレクトリを移動する)	cd ホームディレクトリ($HOME)に移動する cd .. 一つ上のディレクトリに移動する cd [dir] dir に移動する
ls	(ファイルやディレクトリ名を表示する)	-a 通常のファイルの他、ドットファイルも含めて表示する -F ディレクトリやシンボリックリンク等の分類記号を付加して表示 -l ファイルやディレクトリの所有者、保護モード、最終更新日時を付加して表示する -d ディレクトリを表示
mkdir	(ディレクトリを作成する)	mkdir [dir_name] pwd (カレントディレクトリを表示する)

ファイル操作系

cat	(ファイルの中身を表示す	cat [ファイル]
less	(ファイルの内容を一画面ずつ表示する)	less [ファイル...] less 中のコマンド [j]一行下送り [k]一行上送り [数字 G]数字行にジャンプ [/]下方向検索(n,p で次前候補) [?]上方向検索(n,p で次前候補) [Space]次画面 [-N]行番号表示 [:n]複数ファイルを指定した時に次のファイルへ [:p]複数ファイルを指定した時に前のファイルへ [q]で終了
head	(ファイルの先頭部分を表示)	head [オプション] [ファイル...] -n 先頭から指定した行数 n 行だけを表示する
tail	ファイル末尾部分を表示する	tail [オプション] [ファイル...] -n ファイル末尾から指定した行数 n 行だけを表示する
grep	(ファイルから文字列を検索する)	grep [オプション] [文字列パターン] [ファイル...] -v 指定した文字列パターンを含まない行を表示する -l 指定した文字列を含むファイル名を表示する
wc	(テキストファイルの文字数や単語数等数える)	wc [オプション] [ファイル n] -l 行数を表示する -w 単語数を表示する -c 文字数を表示する
diff	(2 つのファイルの内容の違いを調べる)	diff [オプション] [ファイル 1(ディレクトリ 1)] [ファイル 2(ディレクトリ 2)] -i 大文字と小文字の違いを無視する -r サブディレクトリまで比較する -s 違いを調べたファイルを表示する
chown	(ファイルの所有権を変更する)	chown [オプション] [ユーザー名] [ファイル(ディレクトリ)]
chgrp	(ファイルの所有グループを変更する)	chgrp [オプション] [ユーザー名] [ファイル(ディレクトリ)] -R サブディレクトリ以下もまとめて変更する
chmod	(ファイルモードを変更する)	chmod [オプション] [モード] [ファイル...] -R サブディレクトリ以下もまとめて変更する
tar	(ファイルを保管・復元する)	tar [オプション] [ファイル n] tar -cvzf hoge.tgz hoge1 hoge2 hoge/ (hoge1,hoge2,hoge/を圧縮して hoge.tgz を生成する) tar -xvzf hoge.tgz (hoge.tgz を展開する)

		tar –tvzf hoge.tgz (hoge.tgz に圧縮されている内容を表示する。) –f 新しく保管するファイル名を指定する –c 新しく保管ファイルを作成する –x 指定したファイルを保管ファイルから復元する –t 指定したファイルを保管ファイルから探し、あればそのファイル名を表示し、無ければすべてを表示する –v 保管・復元時の情報を表示する –z gzip 形式の圧縮・展開を同時に行う –C 展開先を指定する
ln	(リンクを作成する)	ln [オプション] [オリジナルファイル(ディレクトリ)] [リンクファイル名] ln –s /usr/local/bin/hoge . (カレントディレクトリに /usr/local/bin/hoge のシンボリックリンク hoge を作る) –s シンボリックリンクで別名をつける
find	(ファイルを検索する)	find [開始ディレクトリ] [検索条件] [処理方法] find . –name ″*.txt″ –print (カレントディレクトリ以下の*.txt ファイルを探す)
cp	(ファイルのコピーする)	cp [オプション] [コピー元ファイル…,ディレクトリ…] [コピー先ディレクトリ] –R ディレクとごとコピーする –p 日付、フラグ等のファイル情報をそのままコピーする
mv	(ファイルを移動する、名前を変更する)	mv [オプション] [移動元ファイル,ディレクトリ] [移動先ディレクトリ]
rm	(ファイルやディレクトリを削除する)	rm [オプション] [ファイル, ディレクトリ] –r ディレクトリの削除を行う –f 確認なしに削除する。

数字操作

seq	(連続する数字を生成)	seq 1 2 10 1,23,5,7,9 の数値を生成 bc (電卓) bc –l

シェル記号

|	(標準出力を別のプログラムに渡す)	ls | less ls の結果を less で見る
>	(標準出力をファイルに書き込む)	ls > hoge.txt ls の結果を hoge.txt に書き出す
>>	(標準出力をファイルに追加で書き込む)	ls > hoge.txt ls の結果を hoge.txt に追加で書き出す

システム管理

which	(プログラムの存在するパスを表示する)	which [コマンド]
echo	(変数の値を表示する)	echo [$環境変数] echo $HOME (HOME にセットされている値を表示する)
export	(環境変数をセットする)	export [環境変数]=値 export PATH=/usr/local/bin:$PATH (PATH の頭にに/usr/local/bin

		をセットする) df (ファイルシステムの容量を表示する) du (ディスク使用量を表示する)
du	ディスク使用量を表示する	du [オプション] -b 表示するディスク使用量の単位をバイト単位で表示する -s 合計サイズだけ表示する
ps	(プロセスの状態を表示する)	ps [オプション] -a 全てのユーザーのプロセスを表示する -x 制御端末のないプロセス情報を表示する
man	(コマンドに関するオンラインマニュアルを表示する)	man [コマンド] -k マニュアルのキーワードを指定
kill	(プロセスを終了させる)	kill [プロセス番号] -KILL プロセスの強制終了

3.9 MS-DOS のコマンド

下の表は、16 ビットのパソコン NEC―PC9801(1982～1995 年代)で利用されていた MS-DOS のコマンド名の一覧です。Windows の環境でも、すべてではありませんが、CUI の環境を実現して実行できます(第 2.3.4 項参照)。使い方についてのヘルプは、コマンド名に続けて「/?」と入力すると説明がスクリーンに表示されます。
なお、CUI(character user interface)は、GUI が使われ始めたことに対応させて、後から決めた頭字語です。

MS-DOS コマンド名 (PC-9801 で使われた)

ディスクとファイル	ディスクの管理	CHKDSK, DISKCOPY, DISKINIT, FDISK, FORMAT, LABEL, SCANDISK, SHARE, SMARTDRV, VOL
	ファイルの管理	COPY, DEL, DIR, ERASE, FIND, MOVE, REN (RENAME), TYPE
	ディレクトリの管理	CD (CHDIR), DELTREE, MD (MKDIR), PATH, RD (RMDIR), XCOPY
	メモリの管理	MEM, SETVER
システムとデバイス	システムの制御	DATE, DOSKEY, EXIT, PROMPT, SET, SYS, TIME, VER, WIN
	デバイスの制御	BREAK, CLS, CTTY, VERIFY, MODE
	リスト制御	FIND, SORT, MORE
その他	内部コマンド	LH (LOADHIGH), LFNFOR, LOCK/UNLOCK, SHIFT
	外部コマンド	ADDDRV/DELDRV, ATTRIB, COMMAND, EMM386, EXTRACT, FC, IPEXTRACT, JVIEW, REGEDIT
バッチ処理	*.BAT ファイル	AUTOEXEC、CALL, CHOICE, ECHO, FOR[IN/DO], GOTO, IF, PAUSE, REM
	*.SYS ファイル	CONFIG.SYS、BUFFERS, DEVICE, FILES
ユーティリティ	DEBUG	A(アセンブル), C, D(ダンプ), E, F, G, H, I, L, M, N, O, P, Q, R, S, T, U(逆アセンブル)
	テキスト編集	EDIT
	日本語処理など	CHCP/CHCV, JP/US, MSIMEKEY, MSIMELST, MSIMERGN, MSIMESET,SELKKC
	通信	NBTSTAT, NET, NETSTAT, ROUTE

3.10 MS-EXCEL の関数名

数学/三角関数(数値の集計、平方、丸め、剰余、公約数、公倍数、符号、組み合わせ、ベキ級数、平方根、指数関数、対数関数、円周率、三角関数、双曲線関数、行列、乱数、検索)

SUM, SUMIF, SUMIFS, SUBTOTAL, AGGREGATE, PRODUCT, SUMPRODUCT, SUMSQ, SUMX2PY2, SUMX2MY2, SUMXMY2, ROUNDDOWN/TRUNC, ROUNDUP, ROUND, INT, FLOOR.MATH, FLOOR, LOOR.PRECISE, MROUND, CEILING.MATH, CEILING, ISO.CEILING/CEILING.PRECISE, EVEN/ODD, QUOTIENT, MOD, ABS, SIGN, GCD, LCM, FACT, FACTDOUBLE, PERMUT, PERMUTATIONA, COMBIN, COMBINA, MULTINOMIAL, SERIESSUM, SQRT, SQRTPI, POWER, EXP, LOG, LOG10, LN, PI, RADIANS, DEGREES, SIN, COS, TAN, CSC, SEC, COT, ASIN, ACOS, ATAN, ATAN2, ACOT, SINH, COSH, TANH, CSCH, SECH, COTH, ASINH, ACOSH, ATANH, ACOTH, MDETERM, MINVERSE, MMULT, MUNIT, SEQUENCE, RANDBETWEEN, RAND, RANDARRAY,

日付/時刻関数(期日、日数、期間、時刻、年月日、など)

TODAY/NOW, YEAR, MONTH, DAY, MINUTE, SECOND, HOUR, WEEKDAY, WEEKNUM, ISOWEEKNUM, DATESTRING, DATEVALUE, DATE, TIMEVALUE, TIME, EOMONTH, EDATE, WORKDAY, WORKDAY.INTL, DAYS, DAYS360, NETWORKDAYS, NETWORKDAYS.INTL, DATEDIF, YEARFRAC,

統計関数(平均値、最大・最小、頻度、分散、回帰、相関、分布、検定、)

COUNT/COUNTA, COUNTBLANK, COUNTIF, COUNTIFS, AVERAGE/AVERAGEA, AVERAGEIF, AVERAGEIFS, TRIMMEAN, GEOMEAN, HARMEAN, MAX/MAXA, MAXIFS, MIN/MINA, MINIFS, FREQUENCY, MEDIAN, MODE.SNGL/MODE, MODE.MULT, LARGE, SMALL, RANK.EQ/RANK, RANK.AVG, PERCENTILE.INC/PERCENTILE, PERCENTILE.EXC, PERCENTRANK.INC/ERCENTRANK, PERCENTRANK.EXC, QUARTILE.INC/QUARTILE, QUARTILE.EXC, VAR.P/VARP, VARPA, VAR.S/VAR, VARA, STDEV.P/STDEVP, STDEVPA, STDEV.S/TDEV, STDEVA, AVEDEV, DEVSQ, STANDARDIZE, SKEW, SKEW.P, KURT, FORECAST, FORECAST.LINEAR, TREND, SLOPE, INTERCEPT, LINEST, STEYX, RSQ, FORECAST.ETS, FORECAST.ETS.CONFINT, FORECAST.ES.SEASONALITY, FORECAST.ETS.STAT, GROWTH, LOGEST, CORREL/PEARSON, COVARIANCE.P/COVAR, COVARIANCE.S, CONFIDENCE.NORM/CONFIDENCE, CONFIDENCE.T, PROB, BINOM.DIST/BINOMDIST, BINOM.DIST.RANGE, BINOM.INV/RITBINOM, NEGBINOM.DIST, NEGBINOMDIST, HYPGEOM.DIST, HYPGEOMDIST, POISSON.DIST/POISSON, NORM.DIST/NORMDIST, NORM.INV/ORMINV, NORM.S.DIST, NORMSDIST, NORM.S.INV/NORMSINV, PHI, GAUSS, LOGNORM.DIST/LOGNORMDIST, LOGNORM.NV/LOGINV, CHISQ.DIST, CHISQ.DIST.RT/CHIDIST, CHISQ.INV, CHISQ.INV.RT/CHIINV, CHISQ.TEST/CHITEST, T.DIST, T.DIST.RT, T.DIST.2T, TDIST, T.INV, T.INV.2T/TINV, T.TEST/TTEST, Z.TEST/ZTEST, F.DIST, F.DIST.RT/FDIST, F.INV, F.INV.RT/FINV, F.TEST/TEST, FISHER, FISHERINV, EXPON.DIST/EXPONDIST, GAMMA, GAMMA.DIST/GAMMADIST, GAMMA.INV/GAMMAINV, GAMMALN.PRECISE/GAMMALN, BETA.DIST, BETADIST, BETA.INV/BETAINV, WEIBULL.DIST/WEIBULL,

文字列操作関数(全角・半角、大文字・小文字、文字列長さ、取り出し、連結、取り換え、検索、文字コード)

LEN/LENB, LEFT/LEFTB, RIGHT/RIGHTB, MID/MIDB, FIND/FINDB, SEARCH/SEARCHB, REPLACE/REPLACEB, SUBSTITUTE, CONCATENATE, CONCAT, TEXTJOIN, TRIM, CLEAN, PHONETIC, REPT, CODE/UNICODE, CHAR/UNICHAR, ASC/JIS, YEN/DOLLAR, UPPER/LOWER, TEXT, FIXED, VALUE, NUMBERVALUE, NUMBERSTRING, BAHTTEXT, ROMAN, ARABIC, PROPER, EXACT, T, LET, LAMBDA,

論理関数

AND, OR, XOR, IFS, SWITCH, NOT, IFERROR/IFNA, TRUE, FALSE,

検索/行列関数

XLOOKUP, VLOOKUP, HLOOKUP, LOOKUP, CHOOSE, INDEX, OFFSET, COLUMN, ROW, MATCH, XMATCH, COLUMNS, ROWS, AREAS, INDIRECT, ADDRESS, TRANSPOSE, FILTER, UNIQUE, SORT, SORTBY, HYPERLINK, GETPIVOTDATA, RTD, FIELDVALUE,

Web 関数

ENCODEURL, WEBSERVICE, FILTERXML,

データベース関数
DCOUNT, DCOUNTA, DSUM, DAVERAGE, DPRODUCT, DMAX/DMIN, DGET, DVAR, DVARP, DSTDEV, DSTDEVP,
財務関数（ローン、金利、証券、償却、など）
PMT, PPMT, CUMPRINC, IPMT, CUMIPMT, ISPMT, PV, FV, FVSCHEDULE, NPER, RATE, EFFECT/NOMINAL, RRI, PDURATION, NPV, XNPV, IRR, XIRR, MIRR, YIELD, PRICE, ACCRINT, COUPPCD/COUPNCD, COUPNUM, COUPDAYBS/COUPDAYSNC, COUPDAYS, DURATION, MDURATION, ODDFYIELD/ODDLYIELD, ODDFPRICE/ODDLPRICE, YIELDMAT, PRICEMAT, ACCRINTM, YIELDDISC, INTRATE, RECEIVED, PRICEDISC, DISC, TBILLYIELD, TBILLEQ, TBILLPRICE, DOLLARDE, DOLLARFR, SLN, DB, DDB, VDB, SYD, AMORLINC/AMORDEGRC,
エンジニアリング関数（数表示、複素数、ベッセル関数、誤差関数、など）
CONVERT, DELTA, GESTEP, DEC2BIN, DEC2OCT, DEC2HEX, BASE, BIN2OCT, BIN2DEC, BIN2HEX, CT2BIN, OCT2DEC, OCT2HEX, HEX2BIN, HEX2OCT, HEX2DEC, DECIMAL, BITAND, BITOR/BITXOR, BITLSHIFT/BITRSHIFT, COMPLEX, IMREAL/IMAGINARY, IMCONJUGATE, IMABS, IMARGUMENT, IMSUM, IMSUB, IMPRODUCT, IMDIV, IMSQRT, IMPOWER, IMEXP, IMLN, IMLOG10, IMLOG2, IMSIN, IMCOS, IMTAN, IMCSC, IMSEC, IMCOT, IMSINH, IMCOSH, IMCSCH, IMSECH, BESSELJ, BESSELY, BESSELI, BESSELK, ERF/ERF.PRECISE, ERFC/ERFC.PRECISE,
情報関数（データ型、内容調査など）
CELL, ISBLANK, ISERROR/ISERR, ISNA, ISTEXT/ISNONTEXT, ISNUMBER, ISEVEN/ISODD, ISLOGICAL, ISFORMULA, FORMULATEXT, ISREF, INFO, SHEET, SHEETS, ERROR.TYPE, TYPE, NA, N,
キューブ関数（大規模データベース、集計など）
CUBEMEMBER, CUBEMEMBERPROPERTY, CUBESET, CUBESETCOUNT, CUBEVALUE, CUBERANKEDMEMBER, CUBEKPIMEMBER,

3.11 BASIC 言語

3.11.1. 概説

BASIC または Basic 言語は、1964 年にダートマス大学の John Kemeny と Thomas Kurtz の開発です。この言語は、初心者がコンピュータのプログラミングを容易に理解できるようにという教育目的を持って提案されました。BASIC の名前は、英単語の形容詞(basic)と語呂合わせをして、Beginner's All-purpose Symbolic Instruction Code の頭文字を当てはめて命名されたものです。現在では、英大文字で始める場合はコンピュータ言語を表す固有名詞であるとして辞書に載っています。日本では、PC-8001(1979)に組み込まれていた BASIC 風のコマンドとしての利用が始まりです。マイクロソフト社の開発であって、通称は N-BASIC です。後続の BASIC は、インタプリタとしての機能を持たせながら、高級言語並みのコンパイラとしての利用をユーザに提供していました。言語仕様として BASIC を標榜していますが、プログラミングツールとしてみれば、代表的な三つの製品;N88BASIC、QuickBasic、Visual-Basic は異なった言語処理の体系(システム)です。これらは、歴史的に見ると、BASIC 言語の進化の過程を代表する言語です。

3.11.2 N-BASIC(NEC-PC8001)

8 ビットのマイクロコンピュータ PC-8001 に電源を入れると、OS の PL/M が起動します。実行形式プログラム作成ツールではではありません。ユーザがマイコンの使い方を覚えることは、BASIC の使い方を覚えることと同義でした。8 ビットのマイコンは、一般大衆に、マイクロコンピュータ、さらには、より高性能のパーソナルコンピュータ(personal computer:パソコン)の利用を普及させることに貢献しました。マイコンに電源を入れて立ち上げた状態は、**ダイレクトモード**です。マイコンはメモリ空間(RAM)が狭いので、コンパイラ型式で BASIC 言語を編集するのに代えて、実行形式に編集した**スクリプト言語**が OS のコマンドと共に ROM に保存されていました。PL/M は、この言語を、疑似的なプログラム文書に編集する**編集モード**に移行して、メモリ(RAM)に保存します。この文構造は、行単位に順番号のラベルを付けます。GOTO 文での処理の移動先を指定するために必要になります。コマンドの RUN で**実行モード**になり、スクリプト言語を解読して処理を実行してくれます。ユーザからは、見れば、コンパイラで編集した実行形式のプログラムとの区別をしなくてもよい簡便さがあります。ただし、原則として、疑似的なプログラムの文字並びに、PL/M のコマンドを含めることはできません。マイコンで BASIC を利用することが普及しましたので、BASIC の言語仕様の標準化が、後追いの形で JIS X3003-1982 電子計算機プログラム言語 基本 BASIC で規格化されました。これは、主として8ビットのマイコンに実装することを考えて制定されたものです。この規格に含まれているキーワードは、下に示すように僅か 25 語、組み込み関数は 11 種類です。

基本 BASIC のキーワード

キーワード	BASE, DATA, DEF, DIM, END, FOR, GOSUB, GOTO, IF, INPUT, LET, NEXT, ON, OPTION, RANDOMIZE, READ, REM, RESTORE, RETURN, STEP, STOP, SUB, THEN, TO
組み込み関数	ABS, ATN, COS, EXP, INT, LOG, RND, SGN, SIN, SQR, TAN

3.11.3 N88-BASIC(インタプリタ型 BASIC)

マイコンが 16 ビットのプロセッサを利用するようになって、マイコンの名称から、パソコン(personal computer)と呼ばれるようになりました。この場合の処理方式は、まず DOS を立ち上げ、その管理の下で、アプリケーションプログラムの一つとして、マイクロソフト社のコンパイラ方式の BASIC プログラミングツールが機能しました。この実行は、N-BASIC と同じように、インタプリタ方式の採用でした。BASIC 言語の仕様も拡張され、ANSI、-X3113-1987 および ISO/IEC 10279:1991 などで規格化が進み、それらとの整合性をもたせた日本規格が「JIS X3003-1993 電子計算機プログラミング言語 Full Basic」に改訂されました。

Full BASIC のキーワード　　(複数ページに表示されています)

文字	英字(alphabet)	a ～ z(小文字)、A ～ Z(大文字)
	数字(digit)	0 ～ 9
	記号(symbol)	！ ＃ ＄ ％ ＆ ‘ ( ) ＊ ， ／ ： ； ＜ ＝ ＞ ？ ＾ ＿ ． ＋ － ″
	空白	スペース(SP)
	コメント	REM 文。‘または！で始まり、行末までの文字列をコメントとすることがある。

一般	単純キーワード	AT, IN, ON, OUT, FROM, IS, TO, GET, PUT, WITH
	演算子	AND, NOT, OR

編集		DELETE, EXTRACT, FIRST, LAST, LIST, RENUMBER
プログラム要素		CHAIN, END, EXTERNAL, FUNCTION, PROGRAM, SUB
宣言と定義		ARITHMETIC, BASE, DATA, DECIMAL, DECLARE, DEF, DIM, ERASE, FIXED, MAT, NUMERIC, OPTION, REM, STRING,
実行文	単純実行	LET, ASK, STOP
	制御	EXIT, FOR-TO-STEP-NEXT, SELECT CASE, DO-WHILE, IF-THEN-ELSE, LOOP, IF-MISSING/THERE, UNTIL, GOSUB/GOTO, RETURN, CALL
ファイル	内部ファイル	DATA, READ, RESTORE, IF-MISSING, IF-THERE
	入力	ELAPSED, INPUT, LINE INPUT, PROMPT, TIMEOUT
	出力	PRINT, SET, MARGIN
	ファイル入出力	OPEN, CLOSE, GET-FROM, INPUT, REWRITE, SET, TEMPLATE, WRITE
	ファイル属性	ACCESS, BEGIN, COLLATE, DATUM, DEVICE, ERASABLE, FILETYPE, IMAGE, INTERNAL, KEY, LENGTH, MARGIN, NAME, NATIVE, NUMCHARS, ORGANIZATION, OUTIN, OUTPUT, POINTER, RECORD, RECSIZE, RECTYPE, SAME, SETTER, SKIP, STANDARD, TEMPLATE, VARIABLE, ZONEWIDTH
実時間処理		CONNECT, DELAY, DISCONNECT, EVENT, FROM, MESSAGE, PARACT, PARSTOP, PORT, PROCESS, PUT, RECEIVE, SEIZE, SEND, SHARED, SIGNAL, START, STRUCTURE, TIME, URGENCY, WAIT
グラフィックス	デバイス	CHOICE, CLEAR, CLIP, COLOR, DEVICE, STATUS, VIEWPORT, WINDOW
	図形要素	AREA, CELLS, PICTURE, GRAPH, PIXEL
	文字、線種、	HEIGHT, JUSTIFY, LIMIT, RANGE, STYLE, TEXT
	図形入出力	DRAW, MULTIPOINT, PLOT, POINT, POINTS, SHIFT, SCALE, ROTATE, SHEAR, LOCATE
例外状態、デバッグ		BREAK, CAUSE, DEBUG, EXCEPTION, HANDLER, RETRY, TRACE, USE, WHEN
ベンダー仕様(一例)	宣言と定義	AS, COMMON, CONST, DEFINT, DEFLNG, DEFSNG, DEFDBL, DEFSTR, DOUBLE, ENVIRON, INTEGER, LONG, REDIM, SINGLE, STATIC, TYPE
	実行	BEEP, ELSEIF, ERROR, RESET, RESUME, RUN, SWAP, SYSTEM, WEND
	デバッグ	TRON, TROFF
	演算子	XOR, EQV, IMP
	入出力	LPRINT, POKE, PRESET, PSET, SEEK
	ファイル	BLOAD, BSAVE, FILES, KILL, LOAD, MERGE, SAVE
	グラフィックス	CIRCLE, CLS, PAINT, PALETTE, SCREEN, VIEW, WIDTH

数値関数	数の処理	ABS, INT, MOD, REMAINDER, SGN, SQR, EP, IP,
	最大、最小、切り捨て、丸め	CEIL, EPS, MAX, MIN, ROUND, TRUNCATE
	定数、乱数	MAXNUM, PI, RND,
	三角関数、逆三角関数、	COS, SIN, TAN, SEC, CSC, COT, ACOS, ASIN, ATN, ANGLE, DEG, RAD,
	双曲線関数、指数関数、対数	COSH, SINH, TANH, EXP, LOG, LOG10, LOG2,
	日付け、時間	DATE, TIME
		OPTION 文の引き数として、DEGREES, RADIANS
	その他	宣言文:RANDOMIZE、数の代入文:LET
	ベンダー仕様	CDBL, CINT, CLNG, CSNG, CSRLIN, ERL, ERR, FIX, PEEK, TIMER
文字列関数		CHR$, STR$, USING$, REPEAT$, LTRIM$, RTRIM$
		LCASE$, UCASE$,
		LEN, MAXLEN, POS, VAL, ORD

		DATE$, TIME$,
	ベンダー仕様	ASC, CDBL$, CSNG$, CVI, CVL, CVS, CVD, EOF, HEX$, INKEY$, INSTR,LEFT$, MID$, OCT$, POS, RIGHT$, SPACE$, SPC, TAB
ビット列関数		BVAL, BSTR$

括弧	( )	括弧	
算術演算子	^	ベキ乗	
	* /	乗算と除算	
	¥	整数除算	
	MOD	剰余	
	+ −	加算と減算	
文字列演算子	&	文字列の連結、+記号を用いるものもある。	
関係演算子	=	等しい	
	<> ><	等しくない	
	<	未満、小さい	
	<= =<	以下、等しいか小さい	
	>	超過、大きい	
	>= =>	以上、等しいか大きい	
論理演算子	NOT	論理否定	XOR, EQV, IMP を加える仕様もある。
	AND	論理積	
	OR	論理和	
代入演算子	=		

3.11.4 Quick-Basic(構造化プログラミング言語)

- コンピュータのプログラミング言語の一つとして、BASIC 言語も互換性が要望されるようになり、言語としての高級化、標準化、規格化へと進みました。Kemeny と Kurtz は、1983 年、先の基本 BASIC を発展させ拡張した True BASIC を発表しました。この言語仕様には、構造化プログラミングの思想が導入され、FORTRAN や C と肩を並べるコンパイラ型の高級言語へと変質しました。行番号を使わないプログラミングの方法(パラダイム)です。処理の手順に沿ってプログラム文を書けば、すべての行に行番号を付ける必要がありません。FOR-NEXT、IF_THEN_ELSE 文などを書く場合にも、キーワードを利用した構文を考えれば行番号を使わなくても済みます。また、繰り返し制御や選択のためのキーワードとして、CASE, LOOP, SELECT, WHILE なども利用します。しかし、処理の分岐や復帰は必ず必要になりますので、CALL、GOSUB、GOTO 文では、英字名のラベルを付けて利用します。
- さらに、ANSI X3.113-1987 および ISO/IEC 10279:1991 などで規格化が進み、それらとの整合性を持たせて、以前の同名の日本規格が JIS X3003-1993 電子計算機プログラミング言語 Full BASIC に改訂されました。

3.11.5 Visual-Basic 6.0(オブジェクト指向言語)

- 8 ビットのマイクロコンピュータは、16 ビットのプロセッサの利用を経て、32 ビット、さらに 64 ビットのプロセッサを採用するように進化してきました。従来の DOS のような CUI の OS 環境から、Windows のような GUI の環境に変ったことを反映して、Basic 言語も変化しました。CUI と GUI とではユーザインタフェースの設計が全く異なりますので、Windows のユーザインタフェースをサポートする関数などを大幅に追加しなければなりません。これらの関数は OS の中身を知り尽くしたベンダーが提供するライブラリを利用します。つまり、Visual Basic6.0(VB6 と略記)は、Windows の環境に合うようなプログラムを製作する開発ツールの性格を持ったアプリケーションプログラムです。1998 年に発売され、好評でしたが、2008 年にサポート体制が終了し、現在の Windows 系のパソコンでは利用できなくなりました。しかし、VB6 の利用環境がIDEで構成されていましたので、IDEの使い方を理解する教育ツールとしても利用価値がありました。
- GUI の環境では、ユーザの入力インターフェースがキーボードだけでなく、マウスが加わり、さらに入力そのものの種類と方法とが多様に選択できます。コンピュータ側は、ユーザが何をしたのかを識別して、それに対応した処理を行なわせます。そこで、標準入出力に代わって、事象駆動(event-driven:イベントドリブン)という入出力方法が採用されます。入出力が発生する箇所を、プログラミング上では一箇所にまとめ、その挙動を監視し、何かのイベントが発生したら、そのイベントの種類と性質とによって必要な処理単位を呼ぶようにプログラミングを組み立てます。イベントの解読はシステムに依存しますので、その部分はベンダーが提供する膨大なライブラリ群を利用しなければなりません。オブジェクト指向プログラミング(object oriented programming)は、部品化を進めたプログラミングです。ユーザの入力操作の生じる場所と手段ごとに、そのデータ、処理方法などを、独立した単位にまとめます。プログラミングコードの書き方は BASIC 流に記述し、操作性もインタラクティブになっています。従来の BASIC のキーワードに比べて、オブジェクトに関連した予約語が非常に多くなっています。
- プログラミングのソースコードは、拡張子として(*.bas)と(*.frm)の二種類のモジュールを利用します。前者を**標準モジュール**、後者を**フォームモジュール**と言います。この全体をコンパイルするため、VB6 は統合開発環境(IDE)の下にユーザが作業します。IDE は、種々のファイルを利用しますので、全体を管理するため一つのプロジェクトファイル(*.vbp)を利用します。Quick Basic または FORTRAN などを VB6 に移植する場合には、これらに多少の変更を加えて標準モジュールに転用することができますが、フォームモジュール部分は追加作成します。
- Windows の環境で動作する実行形式のプログラムは、**フォーム**(通称で言うウインドウ)の物理的かつ機能的デザインから始めます。フォームに載せる種々の部品(コントロールと、それを機能させる命令語(メソッド)全体をオブジェクトと総称します。**フォームモジュール**は、このデザインを記述するテキスト形式のファイルであって、フォーム単位で作成する。つまり、フォームモジュールの作成がオブジェクト指向プログラミングの骨格を構成します。フォームモジュールのテキストの大部分を裏で書き込むようになっていて、ユーザの眼に直接触れません。しかし、ユーザは、このテキストの内容について理解しておくことが必要です。フォームモジュールの中身は、テキストエディタで開けば見ることができます。しかし、勝手に変更を加えると理解できないエラーを起こす危険があります。
- フォームモジュールのテキストは、オブジェクト(フォーム、コントロール、その他の部品類全部)の定義文と、コールバック関数(イベントプロシージャ)、及びユーザが適宜プログラミングするプロシージャとから構成されます。この中、定義部分はオブジェクトの図形的なデザインを含めた属性(**プロパティ**と言う)の記述です。IDE では、プロパティは表形式になったプロパティウインドウを表示してプログラマの選択もしくは書き込み

ができます。図形に関する部分は GUI の環境で眼でみて確認しながら構成できる GUI のツールを提供してくれます。この定義部分のテキストは、システム側が裏で書き込みます

- **コールバック関数**(Callback)とは、オブジェクトに対してユーザが行うクリックなど(イベントと言う)を認識すると、システム側から処理が入る入口名です。VB6 ではこれを**イベントプロシージャ**と言い、サブルーチン形式です。イベントの種類はオブジェクト毎に固有です。このサブルーチン名は、

  Private Sub オブジェクト名・定義されたイベント名(定義された引き数)

  の形式であって、IDE が発行する。プログラマはこのプロシージャの名前などを変更することはできません。
- イベントプロシージャのコード記述部分に、プログラマは Basic の文でイベントに応じるプログラム文を記入します。この場合、そのイベントに固有のステートメント用のキーワードが用意されています。これを、そのイベントに対応した**メソッド**と言います。したがって、或るキーワードは、他のイベントプロシージャでは意味を持たない場合があります。
- フォームモジュールのイベントプロシージャ部分だけでプログラム文を書き切れない場合、**標準モジュール**にプログラム文をまとめ、そちらに処理を渡す方が、全体プログラムの管理には便利です。例えば、プログラム全体で共通に利用する定数や変数の定義文は、フォームモジュールに書くことができなくて、標準モジュールに書きます。
- 標準モジュールのテキストコードは、IDE の画面上でリストの編集ができます。ただし、このテキスト形式ファイルの先頭には管理用に一行の文字並びが書き込まれていますが、リストには表示されません。
- VB で利用するファイル名は、プロジェクトファイル(*.vbp)に記録されます。このプロジェクトファイルと同じフォルダに無い場合にはパス名が付けてあります。したがって、プログラムのソースコードをフォルダ単位でコピーする場合、必要ファイルが欠けることがあります。また、同じファイルを複数のプロジェクトで利用しているとき、あるプロジェクトで変更を加えると、他のプロジェクトでエラーを起こす危険があります。
- プログラマの作業は、まず、入出力(Input/Output, I/O)の種類と方法とを設計します。ユーザが操作する入力の物理的な装置(デバイス)はマウスとキーボードとです。ユーザが確認する出力用のデバイスはモニタとプリンタです。Windows の環境では、ユーザが単純にキーやマウスを操作しても、何の反応も起きません。プログラマは、キーやマウス操作を受け取る擬似的な I/O 装置をモニタの画面上で準備することから始めます。これが、フォームモジュールのプログラミングです。モニタに擬似的に表示する仮想のデバイスは、フォーム、メニューバー、ツールバー、各種のコントロールなどです。
- 仮想のデバイスは、種類単位ごとに一種の型名が決められています。通称ではオブジェクト、やや具体的な名称はコントロール、そして個々の分類名として、例えば下のようなキーワードの名前で区分されます。

  Label, CommandButton, TextBox, Timer, PictureBox, Line, CheckBox,Frame, RichTextBox
- 上記のキーワードは、言わば変数の型名と考えることができて、コードの中ではプログラマが覚え易い名前で利用します。コントロールはフォームに従属して定義されますので、他のモジュールから参照するときには、「フォームに付けた名前・コントロールに付けた名前・・・」のように修飾して利用します。
- DOS の標準入出力装置の考え方が無くなりましたので、特に文字データを入出力する際、変数の型変換を明示的にプログラミングします。

3.11.6 VBA(Visual Basic for Applications)

マイクロソフト社が VB6 のサポートを終了した 2008 年以降、Basic 言語を直接利用する環境が無くなりました。現在は、マクロ形式の VBA として、Office シリーズ、とりわけ EXCEL の中での利用するためにプログラミングする形になりました。ただし、ユーザは VB6 を使うことの経験が必要ですが、その機会が失われています。なお、マクロとは反復的な作業を支援するために再現実行ができる**イベントシーケンス**(キーストローク、マウス、クリック、遅延などの動作順)を表示するソフトです。

3.11.7 PBasic,GBasic,Geomap

プログラミング言語は、OS の機能を利用するようにオブジェクトコードを作成しますので、利用するコンピュータシステムの OS に依存した言語として販売されます。ユーザのプログラムでは、外部装置(デバイス)は OS を

3.12 技術の継承

3.12.1 共同利用と一人占め利用

企業内のネットワークを介して多くの人が高性能のコンピュータを共同で利用したいとき、I/O に掛かる時間の無駄を省く方法が工夫されます。そうでないと、一人の人がコンピュータを使っていると、他の人が使えなくなるからです。この場合、電話線などを介して複数の個人の端末環境(**ワークステーション**)を作ります。この構成が**ネットワーク**(network)ですが、蜘蛛の巣状の網目構造を意味する Web の用語も普及しました。端末では、あまり高性能でないコンピュータを使っても、実質的に中央のコンピュータを一人占めで使っているような効率的な利用ができます。しかし、このワークステーションでも高性能のパーソナルコンピュータ(パソコン)を使えるようになりましたので、すべての端末をネットワークで相互に結んで効率的に使う方法が工夫されるようになりました。パソコンとは、コンピュータを一人占め(パーソナル)で使う意味の用語ですので、データの入出力の処理にも同じパソコンを使い、またパソコンが長い時間待機状態になる無駄にも頓着しなくなりました。

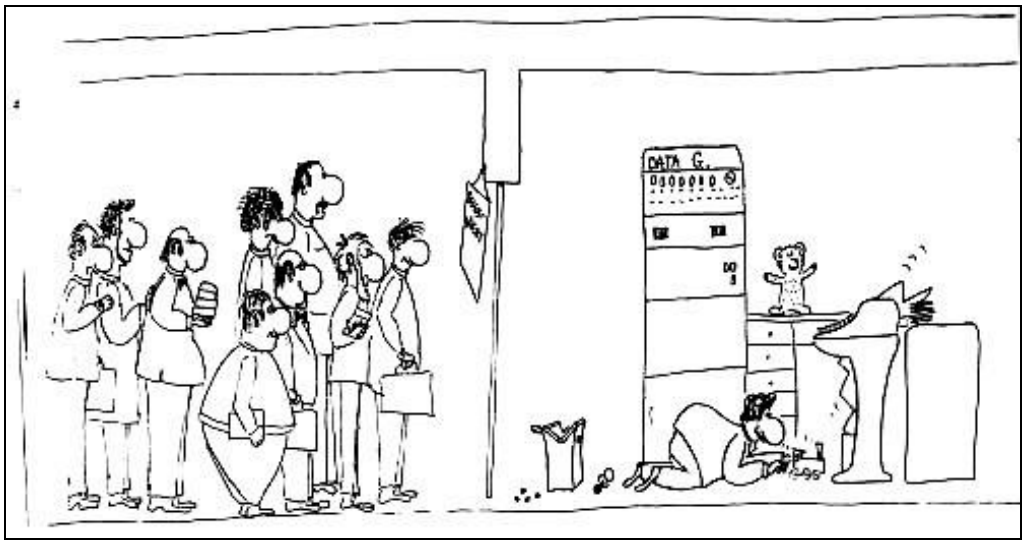

図 3.9 夢にまでみた理想的なパーソナルな環境。第三者からは わがままと見える使い方。(I. Uherkovich 筆者の友人)

3.12.2 大衆化は技術の質を下げる

日本は、技術を持つ職人に敬意を払う伝統があります。その技術名を意味する特別な家名を持つこともあり、親子代々に亘って技術を伝承します。親子関係がなくても、技術を継承してくれる人が名前を引く継ぐこともあります。武道・茶道など、(…道)は技術名に当たり、(…流、…派)は継承者名に当たります。老舗(しにせ)は和語ですが、それに相当する英語はありません。日本以外では、職人は権力者の奴隷扱いがされていました。職人が敬意を受ける社会は民主的です。技術は、創意工夫で向上します。最も経済効果の高い産業が、俗に言う**略奪産業**です。真似をする技術がリバースエンジニアリングですが、先行者のこだわりやノウハウが分からないことがあって、行き過ぎた模倣も招きます。この歯止めが、知的財産権(特許権・著作権など)の制度です。貨幣経済の社会では、経済性が優先され、実用場面では「安かろう・悪かろう」でも使われます。これが技術の質を下げ、また、文化の継承を妨げます。
