記憶の系列位置効果に関する実験制御プログラミング

内田雅人

人の記憶のメカニズムについては、心理学の一分野で古くから研究が進められている。エビングハウスの忘却曲線やジェンキンソンとダレンバッハの干渉説は、忘却や記憶の変容についての考え方に多くの示唆を与えている[8]。われわれが新しくものごとを憶え（記銘）、憶えたことを残しておき（保持）、必要に応じて思い出すこと（想起）は、容易にこなせることもあれば困難なこともある。このようなことは、さまざまな形で現れるが、その中の一つである「系列位置効果」は、系列をなす記憶材料を順序どおりに記憶するときに経験される。日常生活において、例えば、JR秋葉原駅から総武線に乗って、市川駅を目指し、途中の駅名を順番に憶えようとするとき、系列位置効果を経験することができる。「あきはばら」「あさくさばし」、あるいは「こいわ」「いちかわ」といった駅名は比較的早く憶えられるが、中間に位置する「かめいど」「ひらい」などを正しい順序で憶えることは難しいものである。受講生名簿を授業の度に読み上げていると、名簿の始めのほうにある名前と最後のほうにある名前は比較的憶えるのが早いが、中程にある名前は記憶に残りづらいことはよく経験されることである。

系列位置効果は、干渉説によってうまく説明される。すなわち、時間軸に沿って古い記憶内容と新しい記憶内容は相互に交錯し干渉し合い、「順向抑制」および「逆向抑制」を引き起こす。順向抑制とは、古い記憶内容が新しい記憶内容に干渉し、新しい記憶内容の再生を妨げるものである。逆向抑制とは、新しい記憶内容が古い記憶内容に干渉し、古い記憶内容の再生を妨げるものである。系列の中程にある項目は、順向抑制作用と逆向抑制作用を集中的に受け、記憶内容が最も干渉されやすくなり、比較的に憶えにくく思い出しにくいということになる。

心理学の基礎教材となる実験には、このような理論への傾注を促すものがいくつかある。「系列予言法」は、記憶の系列位置効果を実験的に再現する優れた方法の一つである。手続きは、単純である。実験的に統制された条件のもとで、一試行あたりリスト形式に連ねられ

た「無意味綴り単語」〔実験用に作成された連想価の低い単語〕が一語ずつ提示され、被験者がそのリストを順番どおり正確に憶えられるまでにどれだけ試行を要するか、また、それぞれの試行を通じてリスト内のどの単語がいつも記憶されにくいかを、客観的に測定する。被験者に要求されることは、次々に現れる無意味単語を憶えて、ある単語が提示されて次の単語が現れるまでの短いインターバルの間に次の単語を言い当てる（予言する）ことである。

系列予言法で必要な実験制御の条件は、一つひとつの刺激（無意味単語）が全ての試行でいつも決まった順序で、一定の時間提示され、一定の時間間隔で移り変わることである。そして、被験者の反応の正誤を正確に記録することが必要である。

このような実験は、人の手を介して、手作業で刺激提示をしていては正確な条件統制が望めないので、心理学実験機器を用いて機械的に制御される必要がある。系列予言法による実験のために、「メモリードラム」と呼ばれる記憶刺激の提示装置が開発されている。実験者があらかじめ作成した記憶素材（無意味単語）のリストをセットして、スイッチを入れると、一定の時間間隔でつぎつぎに装置の小窓に無意味単語を提示するという、きわめて単純なものである。ただし、特殊な機器であるために価格は驚くほど高価である。しかも、メモリードラムは、刺激提示に特化された機能しか持ち合わせていない。被験者の反応を記録するためには、専ら実験者の手作業による○×式の記録に頼ることが多い。被験者の反応を機械的に記録しようとすれば、さらに高価な機器を購入することになる。

昨今のIT技術の向上は、心理学の研究教育場面における実験環境の向上にもつながっている。パーソナルコンピュータは、今や日常生活道具となり、文房具、あるいはガジェット（便利な小物）として研究教育現場にも普及している。一方では、身近かな存在でありながら、活用範囲は、ワープロ、表集計、プレゼンテーション、インターネットなどの域を出ることがなく、パーソナルコンピュータはその可能性を十分に発揮しているとは言い難い。上に述べた心理学の実験機器を例にとっても、パーソナルコンピュータを活用すれば、コストは恐らく十分の一以内に納まるはずである。著者は、心理学の実験環境を効率的に運用するために、パーソナルコンピュータを心理学の実験機器として活用することを目指してきている[1)2)]。その一環として、本研究では、人の記憶過程における系列位置効果を検証するための実験環境の構築を試みた。

1．本研究の目的―コンピュータによる記憶実験制御の自動化―

人の記憶あるいは学習の過程を解明するような基礎的な心理学の実験においては、当該の刺激と反応の関係を見出すにあたっての混乱要因を極力排除するべく、実験条件を可能な限

り単純化する努力が払われる。それに伴って、実験刺激の提示及び反応の測定に関わる人的ミスも極力排除されなければならない。単純化された条件設定と正確な刺激提示および反応測定を実現するには、実験用機器を導入し機械化した制御を実現することが望まれる。しかし、既製の実験用機器では実験者が意図する個々の実験事態に適正化された条件を設定するうえで必ずしも最適ではないこともある。実験者が実験機器の制約に合わせて実験計画を立て直すようなことがあっては本末転倒である。コンピュータのプログラミングを工夫し、パーソナルコンピュータを導入することによって、実験者本位の実験制御環境を実現することができる。しかも、現今のプログラム開発環境は大幅に進歩し、マシン語やアセンブラ言語の文法と奮闘しながらのプログラミングは過去のものとなっている。

本研究では、我々が日常経験する記憶事象の一般的な特徴を理解するために心理学の基礎実験として広く実施されている「系列予言法」による記憶のシミュレーション実験にパーソナルコンピュータを導入することを目指して、実験制御のためのプログラミングを試みた。

まず、実際に行われる系列予言法の実験手続きの例を見てみる。

題目：「系列位置効果」に関する実験

目的：系列をなす記憶材の記銘保持再生における特徴を把握する

方法：

（装置）　パーソナルコンピュータ　MacOSver8.1以降搭載のマッキントッシュ（Apple社）、およびWindows98以降搭載のウィンドウズ・コンピュータ（各社）

プリンター　StyleWriter2400（Apple社）など

（手続き）「系列予言法」

- 被験者は、コンピュータ画面上に表示される実験インストラクションに従い、自ら実験セッションを開始する。
- 実験セッション開始に伴い、記憶素材である無意味単語12語のリストが、順次一定の時間間隔で画面上に提示される。２秒間の刺激提示と２秒間の刺激提示間隔〔この間は、何も表示されない〕が設けられる。
- 実験の１試行は、12の単語が順次提示されることによって成る。
- 被験者は、２秒間の刺激提示間隔の時間内に次に提示される刺激を言い当てながら（予言しながら）、12の単語を順番どおり全て間違いなく記憶していく。

・実験の試行は、被験者が間違いのない試行を３回連続して遂行できるまで繰り返し続けられる。
・３試行連続無失策が成立すると同時に、画面上に実験終了のメッセージが表示され、つぎに実験結果が表示される。

２．制御プログラム

以上の実験をパーソナルコンピュータで制御するために、プログラム開発環境として「REALbasic（ver3.5）」[3)5)6)7)9)10)11)]を用いた。

(1)　REALbasicについて

REALbasicは、プログラム記述のスクリプトは旧来の「ベーシック（BASIC)」言語に似ているが、コンパイルして実行形式のファイルに作り上げる過程が速く、また作成中のプログラムを即時に並行的に実行しデバッグすることが可能である。しかも、コンピュータOSの深い部分に関わる作業のさまざまなモジュールが用意されており、プログラム開発者はOSとユーザーとのインターフェースに関する知識を必要とされない。つまり、誰でも簡単に自分に適した制御用プログラムを完成させることができるということである。

(2)　REALbasicの構造

REALbasicでは、一つの実験の一連の手続きを記した実行プログラムを作り上げるために、ユーザーは〝プロジェクト〟と呼ばれるソースプログラムを作り上げていく。

まずREALbasicを起動すると、何も描かれていない白紙のウィンドウ（〝ウィンドウエディタ〟あるいは〝ウィンドウデザイナ〟）が現れる。同時に現れる〝ツールパレット〟から、ユーザーが構想する作業に対応したアイテムを適宜マウスでドラッグして白紙ウィンドウに貼り付けていく。例えば、画像を貼り付けたいなら、ツールパレットから〝キャンバス〟というアイテムを白紙のウィンドウに貼り付ければ、その位置に必要な画像を提示する準備ができる。あるいは、白紙ウィンドウに文章を表示したければ、その位置に〝スタティックテキスト〟というアイテムを貼り付ける。因みに本実験プログラムで使用したアイテムは、基本的には〝キャンバス〟、〝スタティックテキスト〟、〝ボタン〟および〝タイマー〟の各アイテムだけである。貼り付けたアイテムの細かな特性は、３番目のウィンドウ(〝プロパティウィンドウ〟あるいは〝プロパティパレット〟）に一覧表示されている。このウィンドウの項目を操作して、ユーザーの目的に合ったアイテムに調整することができる。例えば、本実験プログラムでは、〝タイマー〟の〝プロパティウィンドウ〟を操作して、２秒間隔で繰り

返し信号が発信されるように設定している。

そして、全体的な実験の構想を実行するためには、ユーザーの側から個々のアイテムの働きの内訳について指示を与えていかなくてはならない。ユーザーは、指示の内容を〝コード〟と呼ばれる命令言語によって書き記していく。つまり、プログラムの本体を作成するわけであるが、REALbasicでは、広範囲にわたる基本的な作業内容がサブルーチン化したユニットとして既に用意されいるので、ユーザーは、それらのユニットを適当に組み合わせて望む作業の内容を記述していけばよい。

(3)　実験制御プログラムの概要

１つの実験は、１つの〝プロジェクト〟と呼ばれるプログラムにまとめられる。〝プロジェクト〟はいくつかのサブプログラムの集成体である。本実験制御プログラム（プロジェクト）は、実験開始を制御するための〝オープン〟と呼ばれるイベント*のサブプログラムと、実験セッションを開始するための〝ボタンダウン〟、一定時間毎に記憶用の刺激を提示するための〝タイマー〟、記憶刺激を提示するための〝キャンバス〟などの〝コントロール〟と呼ばれるアイテムと、マウスのクリックを検知するための〝マウスダウン〟イベントを制御するサブプログラム、そして、正答を処理するための〝カウンタ〟、実験終了と結果を処理するための〝ストップ〟といったユーザーによって名付けられた〝メソッド〟と呼ばれるサブプログラムから成っている。〔*ユーザーがコンピュータに対して何らかの操作を行ったとき、それを検知し然るべき処理に向かわせる働きを予め備えた命令ユニットを〝イベント〟と呼ぶ。〕

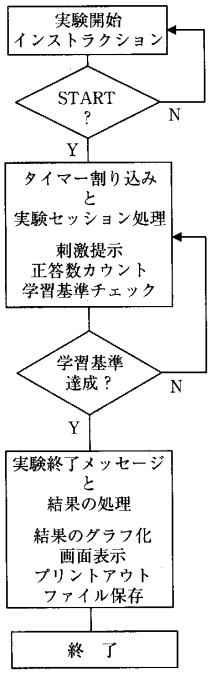
なお、記憶実験材料が実験以外の場で人目に触れることは、実験事態での刺激統制の意味をなさないので、本報告では仮想の実験制御を想定して、提示刺激の内容および刺激数を実際の実験事態とは異なるかたちで構成し掲載することとした。著者が行う実際の系列予言法実験では、予め調査され作成された「無連想価分類表」[4]に基づいて、連想価の低い綴りの組み合わせを12語使用するが、本稿で紹介する記憶材刺激は、「わようじょしだいがく」を逆さ読みして２文字ずつペアにした単語５語を用いたことをお断りしておく。

(4)　実験制御プログラムのフローチャート

系列予言法の実験は、まず実験の概要を示すインストラクションから始まる。被験者は、それを読み、実験への態勢が整った時点で自ら「START」ボタンを押して、実験を開始する。

実験開始と同時に、２秒ごとにタイマー信号が起こり、そのたびに画面上に、まず系列の始まりを合図する「？」マークが提示され、つぎに画面は暗転〔何も表示されていない状態

図1　実験プロジェクトのフローチャート



に変化〕し、次に第1刺激(「よわ」)が提示される。以後、暗転—第2刺激(「しう」)—暗転—第3刺激(「しよ」)……と続き、第5刺激(「くか」)の提示が終わると1試行が終了する。引き続き暗転に移り、それに続いて始めに「?」マークが提示され、次の試行へと移行する。このように試行が繰り返され、被験者は、刺激提示の間の暗転時に次に現れる刺激を予言することを要求される。刺激が提示されるたびに正答が確認されればマウスがクリックされる。提示される全ての刺激に対してマウスがクリックされれば1試行無失策で完了したことになる。このような誤答のない試行が3回連続すれば、実験セッションが終了される。

タイマー割り込み処理プログラムのルーチンでは、つねに刺激提示のたびにマウスのクリックをチェックしていて、クリックがあれば正答反応としてカウントしていく。試行が終わるたびに提示された全刺激に対して正答であったかがチェックされ、3回連続かどうかもチェックされる。3回連続であることが確認されれば、実験終了プログラムルーチンへ処理が移る。

実験終了プログラムルーチンでは、まずタイマーの割り込みが停止され、続いて、変数に保存された正答反応の処理が進められる。提示された5枚の刺激それぞれに対する正答反応の合計数が算出され、棒グラフとして画面上に表示される。つぎに、5枚の刺激それぞれに対する反応の正誤の散布状態が一覧表のかたちで画面上に表示される。

画面上に表示されたこれらの結果は、必要に応じてプリンターより印刷され、あるいは反応の正誤一覧表示がテキストファイル形式でデスクトップに保存されるようになっている。

以上の実験プロジェクトのフローチャートは、図1のとおりである。

3．実験制御プログラムの実際

上に述べた実験制御を実行するためのプログラムを具体的に見ていく。

(1)　実験開始インストラクション画面の作成（ソースコード1，2）

先に述べたように、REALbasicではソースプログラム作成時から平行して、逐次実行形式に移行してプログラムの動作を即時的に確認できることが大きな利点である。さらに、実

行形式にコンパイルされたプログラムは、スタンドアロンのアプリケーションとして生成されるため、REALbasic本体のアプリケーションソフトがインストールされていないコンピュータ上でも実行できるという利点もある。

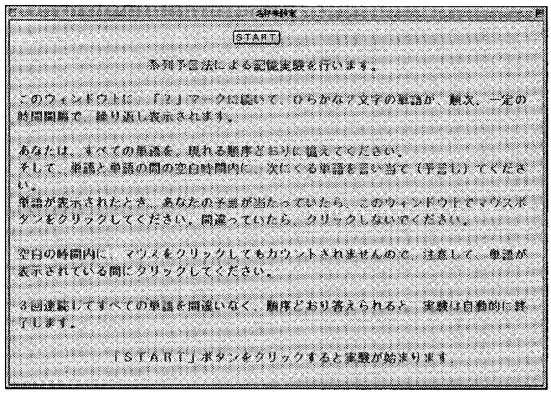
本実験制御プログラムアプリケーション（実行プログラム）は、立ち上がるとまず実験開始インストラクションが表示される（"staticText1"）。実験開始インストラクションの画面は、"open" イベントに必要なコードを書き込むことによって作成する（ソースコード1、図2）。画面上の、STARTボタンは、"PushButton" コントロールの "Action" イベントに必要なコードを書き込むことによって制御する（ソースコード2）。STARTボタンがクリックされると、タイマーが2秒ごとに繰り返し信号を発信するように設定され（"timer1.mode＝2"）、STARTボタン自体は見えなくなる（"me.visible＝false"）。

(2)　タイマー割り込みと実験セッションの処理（ソースコード3，4，5）

実験制御プログラム作成時に、あらかじめ刺激提示と暗転の時間を "timer" コントロール・アイテムの"プロパティ"ウィンドウで2秒に設定しておく。これによって、コンピュータに内蔵された時計からプログラムに2秒ごとの信号（割り込み信号）が届くようになる。この信号が届くタイミングに合わせて、刺激提示時と暗転時に必要な処理内容を "timer" コントロールの "Action" イベントの中に書き込んでいけばよい。書き込む内容には以下のようなことが必要である（ソースコード3）。

"timer" の信号を数えて（"tim＝tim＋1"）、1なら実験開始インストラクションからの切り替わった直後なので、暗転〔何も表示されていない状態〕の2秒間とする。2なら、系列刺激の先頭を示す「？」マークを提示する（図3）。3番目の信号ならまた暗転にする、4番目の信号になって系列刺激の第1番目「よわ」を提示する（図4）。以下同様に進み、5

図2　実験開始インストラクションの画面



番目の刺激「くか」を提示した後の暗転が済んだときに1試行が完了したことになる。カウンタ変数timが偶数のときには画面上に系列刺激のいずれかが提示されていることになる。「?」マークから、「くか」(5番目の刺激)まで提示が終わった時点でtimを御破算すれば、全ての試行で提示される刺激にはいつも同じ通し番号を割り当てることができる。この番号は、変数〝amari〟に保存される。また、第1刺激(「よわ」)が提示されるたびに変数〝cumul〟に試行数が保存される。変数〝amari〟と〝cumul〟は、刺激提示中の反応の有無を全試行全刺激項目ごとに記録するための2次元配列変数〝kiroku〟の要素変数となる。そして、正答すれば〝kiroku(amari,cumul)=1〟、正答できなければ〝kiroku(amari,cumul)=0〟として記録される。

なお、本実験プログラムで使用する提示刺激は、他のアプリケーションで作成した画像ファイルを〝キャンバス〟内に取り込んだものである。

1試行が完了したら、その試行内の全ての提示刺激に正答できたかを調べることが必要である。これは、その試行内での正答数と提示刺激数を比較することによって、同じなら全刺激に正答であったことが分かる。1つの刺激の提示中は、マウスクリックは1回しかカウントされないようになっているので(ソースコード4)、1試行内の刺激提示数とマウスクリック数が一致すれば(ここでは、1試行に5枚の刺激が提示されるのでマウスクリック数が5であれば)、その試行は全刺激に正答であった(無失策であった)ことが確認される(ソースコード5)。全ての刺激に正答できなければ、つぎの試行が繰り返される。全刺激に正答であれば、つぎにそれが3回連続して起こったかが調べられる。まず、全ての刺激に正答できれば特定の変数(〝ok〟カウンタ)が加算される。連続が確認されなければ〝ok〟カウンタは御破算され、次の試行が繰り返される。たとえ前の試行が全正答であっても、今試行に誤答があれば〝ok〟カウンタはやはり御破算され、再度全正答が連続されることを待つこ

図3　系列刺激提示開始を告げるマーク

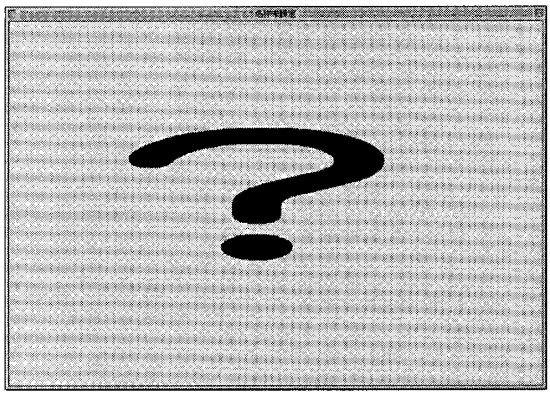
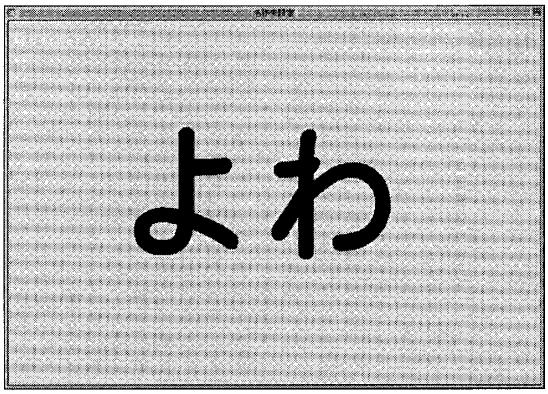


図4　画面上に提示された第1刺激



とになる。全試行に正答が確認され、かつ3試行連続無失策も確認されたら、実験終了を合図するフラグを立てる（"inp＝9999"）（ソースコード3、最後の13行）。タイマー割り込み処理ルーチンの最初の処理で、"inp" フラグが立っていることが検知されると、タイマー割り込みは禁止され、上に述べた刺激提示を含めた一連の作業は停止される。そして、実験セッション終了のメッセージが画面上に3秒間表示され、次に結果の処理ルーチンへ移行する（図5）。

(3) 結果の処理（ソースコード6，7，8，9，10）

実験セッション中に変数 "kiroku" に保存されたデータは、各試行において提示刺激に対して得られた正誤反応の結果である。これは、二通りに整理され、画面上に表示される。一つは、系列位置効果を見るもので、提示された刺激ごとに正答数の合計を算出し、棒グラフで表示される。刺激ごとの合計正答数は、2次元配列変数 "kiroku" の中の "amari" 列にある1（正答マーク）の数を試行数分（変数 "cumul" の中の数）だけ足しこめば得られる。

図5　画面上に表示された実験終了メッセージ

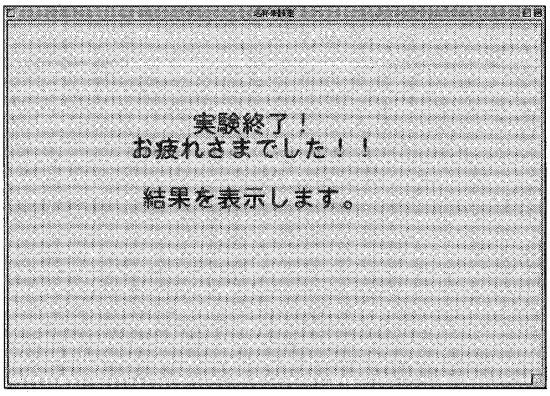
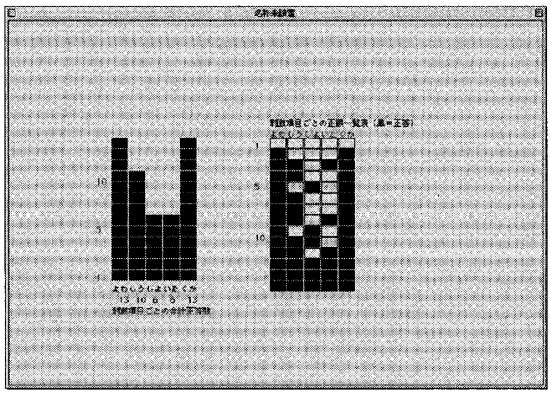


図6　画面上に表示された結果のグラフ



刺激ごとに合計された正答数は、1次元配列変数〝tot〟に保存される(〝tot (tx) =tot (tx) +kiroku (tx,ty)〟)。(ソースコード6)(図6)

グラフ表示には、〝tot〟の中の数値をもとに、画面上に小さな黒塗りの四角形を数値分だけ累積描画して棒グラフを形成し、系列位置効果を表す。

系列位置効果を示すグラフだけでは、系列内のそれぞれの位置にある刺激が学習完成基準に達するまでにどのような経緯を辿ったのかを細かく知ることができない。もう一方の結果の整理法は、そのような学習完成基準に達するまでのローカルな過程を探る助けとなるものである。そのために、実験セッション内の全試行で提示された刺激に対する反応の正誤の散布図が一覧表のかたちで表示される。

〝kiroku (amari,cumul)〟の中の数値をもとに、amari列とcumul行に対応させて画面上に設定した座標軸に沿って小さな四角形を描画していく。黒塗りの四角形は正答を表し、白抜きの四角形は誤答を表す(〝canvas8. graphics.FillRect 350+gx*25, 150+gy*16, 23, 15〟)。

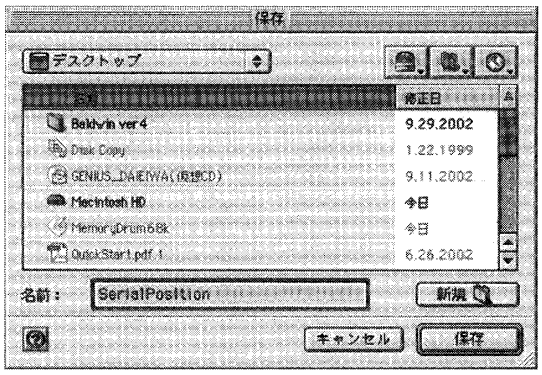
画面表示された結果は、同一の書式でプリントアウトすることが出来る。これには、他のアプリケーションソフト(例えば、マイクロソフト「ワード」など)と同じ方法で、結果表示画面にあるメニューバーの「ファイル」から「印刷」を選択すればよい。

「印刷」と同様に、必要に応じて結果をテキストファイル形式で「保存」することもできる。テキストデータとして保存しておけば、表集計ソフト(例えば、マイクロソフト「エクセル」)などによって更なるデータ処理が可能である(図7)。

なお、「印刷」「保存」を可能にするためには、あらかじめ以下の手続きをプログラムしておく必要がある。

まず、REALbasicの起動にともない〝ウィンドウエディタ〟とともに現れる〝プロジェクタウィンドウ〟の〝menu〟から新しい項目として「用紙設定」「印刷」「保存」を作成し

図7　メニュー―「ファイル」―「保存」の手続き(MacOS ver9. 2の例)



ておく。次に、"ウィンドウエディタ"をプログラムするためのエディタ("コードエディタ")を開くと現れる"コードエディタブラウザ"と呼ばれる一覧表示部の中から"EnableMenuItems"を選択し、さきに新設した項目（「用紙設定」「印刷」「保存」）を使用可能になるように登録する（"PageSetupMenuItem.Enabled＝True" "PrintMenuItem.Enabled＝True" "SaveMenu.Enabled＝true"）（ソースコード7）。

そして、"コードエディタ"の編集メニューから「新規メニューハンドラ」を選び、「用紙設定」「印刷」「保存」に対応した"メニューハンドラ"の名前（"PageSetupMenuItem" "PrintMenuItem" "SaveMenu"）を指定する（ソースコード8）。

あとは、"コードエディタブラウザ"の中から、"PageSetupMenuItem"、"PrintMenuItem"、"SaveMenu"を選択し、それぞれに「用紙設定」、「印刷」、「保存」に対応した作業内容をプログラムする。「印刷」、「保存」のプログラム内容は、画面上に表示するために先に作成したコードをそのままコピーし、一部を変更するだけでよい。「印刷」の場合は、「印刷」のための宣言をしたあと、画面表示のための指示を「印刷」の指示に書き換えればよい（ソースコード9）。「保存」の場合は、「保存」のための宣言をしたあと、画面表示のための指示を「保存」の指示に書き換えればよい（ソースコード10）。これらの点は、ソースコードを見比べることによって容易に理解されるであろう。

4．まとめ

記憶の系列位置効果は、一般心理学の中でもつねに言及される基本的な現象である。日常生活においてもあらゆる場面でよく経験される。しかし、日常場面ではさまざまな要因が交絡しているため、その過程を解明することは困難である。混乱要因を可能な限り排除し、純粋な関連要因を操作し、因果関係を見出すことが実験法の主旨である。本稿で取り上げた系列予言法は、系列位置効果に関わる要因の関係性を解明するというよりは、系列位置効果の存在を客観的な事実として具体的に捉えることに主眼がおかれた、伝統的で基本的な正統派の心理学教育の一手法といえよう。したがって、現在でも心理学コースあるいは一般心理学の基礎実験の一つとして重要な位置を占めている。

伝統的な実験では、当該の実験に特化された特別な器機が用いられることが多い。系列予言法でも、メモリードラムと呼ばれる特別な器機が製造されている。概してこのような特化された器機は機能の割に高価であり汎用性に欠け、実験者の実験構想が必ずしも十分に反映されないことが多い。

本研究では、系列予言法の実験場面をパーソナルコンピュータ上に実現することを試みた。

その目的は、実験者の実験構想が十分に反映されて小回りが利き、安価で、汎用性のある実験環境を構築することである。同時に、実験制御における人的ミスの発生を最小限に止めることも目指された。

実験場面をコンピュータ上に構築するために、プログラム開発言語「REALbasic」(ver 3.5) を用いた。REALbasicの利点は、プログラマーがコンピュータOSの諸機能とユーザーとのインターフェイス部分に関与することなく、直感的にプログラミングを進められることにある。プログラムを構想するためのウィンドウ上に、次々と必要な作業アイテムを貼り付けていき、個々のアイテムに対して必要な作業内容を命令すればよい。

このようにして作成されたプログラムは、同時並行的に実行プログラムに変換される。実行プログラムを動作させるためのアプリケーションソフトが不要であることも利点である。本研究で使用したREALbasicはprofessional版であるので、実行プログラムはウィンドウズマシン上でも動作する。

実行プログラムをマッキントッシュ上で試用してみた結果、当初の実験構想どおりに動作することが確認された。しかし、時間制御の基本クロック信号をコンピュータの内蔵時計から取っていることにより、処理の内容によっては問題点が現われるかも知れない。幸い、本実験では刺激提示の厳密な時間制御はそれほど厳しく要求されておらず、むしろ刺激提示の順序と正誤反応の記録が重視されているので問題は表面化しないであろう。著者自身がREALbasicの特性を十分に熟知することも踏まえて、今後更なる研究が必要である。

実験制御の観点からもう一つの問題点が見られた。本実験プログラムでは、マウスクリックによって反応の正誤を記録しているが、このとき記録が被験者本人に委ねられている点である。このことは一般的な実験の本意からすれば、大きな逸脱である。敢えてこのような手続きを取り入れた理由は、この系列予言法の実験が仮説検証のためではなく、現象を確認し理解するための教材としての実験と捉えたからである。被験者自身が単独で系列位置効果を確認することの便宜を第一に考慮した結果である。しかも、被験者以外の者が実験者としてマウスをクリックすれば、歪みのない記録が可能となり本格的な実験事態にも対応できるのである。もちろん、本プログラムからは質的なデータを得ることができないので、本格的な実験事態に利用する場合は実験者による観察記録も必要である。そのような場合でも、本プログラムによる正誤の記録を利用すれば、誤反応の内容により深く注意を集中することが可能となるであろう。昨今の音声認識機能を取り入れることも構想は可能であるが、現実問題として実現までには時間を要するようである。

これらの問題点が存在はするものの、本プログラムの一部を改変することによって多くは

解消されることもわかっている。本研究は、コンピュータ上での心理学実験環境の構築に向けて今後の課題を提起し、現況の発展への可能性を示すことができたといえよう。さまざまな構想を積極的に取り入れ、今後さらにコンピュータによる心理学実験の制御を発展させていきたい。

5．あとがき

本実験プログラムの実行プログラムは、OS8.1以降を備え5.5MB以上のメモリーを有するマッキントッシュなら、機種を問わず使用できる。また、ウィンドウズマシンでも実行可能である。

本稿で紹介した実験の実行プログラムは必要に応じてフロッピーディスクファイルにて評価版として頒布の用意がある。それらの評価を受けてさらに改訂をはかりたい。

本研究は、和洋学園平成13年度研究奨励費（プロジェクト研究：個人研究）の補助を受けることによって進めることができた。ここに深く感謝申し上げます。

【付表】REALbasicによる系列予言法実験プログラムソースコード一覧

ソースコード1　実験開始インストラクション画面

〈イベント-Open〉

```
//プログラムの初期設定
timer1.mode = 0                    //タイマー割り込み禁止
amari = 0                          //提示刺激番号クリア
inp = 0                            //3試行連続正答フラグクリア
tasu = 0                           //
kazu = 0                           //マウスクリックカウンタクリア
rsp = 0                            //マウスクリック検知変数クリア
rsp2 = 0                           //マウスクリック検知変数2クリア
tim = 0                            //試行内提示刺激枚数カウンタクリア
ok = 0                             //3試行連続正答カウンタクリア
seikai = 0                         //正答数カウンタクリア
retry = 0                          //
cumul = 0                          //試行数カウンタクリア
staticText17.visible = false       //実験終了メッセージを見えなくする
canvas1.visible = false            //「？」マークを見えなくする
```

```
canvas2.visible = false          //刺激「よわ」を見えなくする
canvas3.visible = false          //刺激「しう」を見えなくする
canvas4.visible = false          //刺激「しよ」を見えなくする
canvas5.visible = false          //刺激「いた」を見えなくする
canvas6.visible = false          //刺激「くか」を見えなくする
canvas7.visible = false          //
canvas8.visible = false          //実験終了時の画面を見えなくする
refresh
staticText1.visible = true       //実験開始インストラクションを表示する
```

ソースコード2　STARTボタンの制御

〈コントロール-PushButton-Action〉

```
//「START」ボタンが押されたら以下のことをして実験開始する
timer1.mode = 2                  //タイマー作動
me.visible = false               //ボタンinvisible
staticText1.visible = false      //実験インストラクションを消す
```

ソースコード3　タイマー割り込みと実験セッションの処理

〈コントロール-Timer-Action〉

```
dim fin,finn As integer
if inp = 9999 then               //timフラグが立っていたら
   canvas8.visible = true
   staticText17.visible = true   //実験終了メッセージ表示する
   refresh

   fin = ticks
   finn = fin + 180              //180 * 1/60sec = 3sec
   while finn > = ticks          // 3 秒経ったら
   wend
   staticText17.visible = false  //実験終了メッセージを見えなくする
   refresh

   stop                          //実験終了、結果の処理へ
   timer1.mode = 0               //タイマー割り込み禁止
   return
```

```
end if
//timフラグが立っていなければ
//2秒ごとに以下の割り込み処理をする

//系列予言セッション開始
tim＝tim＋1                    //タイマー割り込みの回数を加算
select case tim
  amari＝tim ￥ 2              //提示刺激の番号をamariに入れる
case 1
  canvas1.visible＝true        //系列開始の合図「？」マーク
  refresh
  amari＝tim ￥ 2
case 2
  canvas1.visible＝false       //「？」はinvisible
  refresh                      //「？」提示中の反応はチェック、カウントしない
case 3
  canvas2.visible＝true        //第1刺激「よわ」visible
  refresh
  amari＝tim ￥ 2
case 4
  canvas2.visible＝false       //第1刺激invisible
  refresh
  cumul＝cumul＋1              //刺激提示試行数をカウント
  if rsp＜＞0 then
    kiroku(amari,cumul)＝1
    kaunta                     //第1刺激でマウスがクリックされたらkauntaへ
  end if
case 5
  canvas3.visible＝true        //第2刺激「しう」visible
  refresh
  amari＝tim ￥ 2
case 6
  canvas3.visible＝false       //第2刺激invisible
  refresh
  if rsp＜＞0 then
```

```
    kiroku(amari,cumul)=1
    kaunta                          //第2刺激でマウスがクリックされたらkauntaへ
  end if
case 7
  canvas4.visible=true              //第3刺激「しよ」visible
  refresh
  amari=tim ¥ 2
case 8
  canvas4.visible=false             //第3刺激invisible
  refresh
  if rsp<>0 then
    kiroku(amari,cumul)=1
    kaunta                          //第3刺激でマウスがクリックされたらkauntaへ
  end if
case 9
  canvas5.visible=true              //第4刺激「いた」visible
  refresh
  amari=tim ¥ 2
case 10
  canvas5.visible=false             //第4刺激invisible
  refresh
  if rsp<>0 then
    kiroku(amari,cumul)=1
    kaunta                          //第4刺激でマウスがクリックされたらkauntaへ
  end if
case 11
  canvas6.visible=true              //第5刺激「くか」visible
  refresh
  amari=tim ¥ 2
case 12
  canvas6.visible=false             //第5刺激invisible
  refresh
  if rsp<>0 then
    kiroku(amari,cumul)=1
    kaunta                          //第5刺激でマウスがクリックされたらkauntaへ
```

```
    end if
  else
    stop
  end select

  rsp＝0
  if tim＝12 then                    //刺激提示サイクルが一巡したら、次のことを確認する。
    tim＝0                           //刺激提示サイクルカウンタをクリア
    if seikai＝5 then                //全て（５つ）の刺激に正答できたか？
      ok＝ok＋1
      if ok＝3 then                  //3試行連続して正答できたか？
        inp＝9999                    //３試行連続正答できたら、終了フラグをオン。
      end if
    else
      ok＝0                          //３試行連続しなければokカウンタはクリア。
    end if
    seikai＝0                        //正答数カウンタをクリア。
  end if
```

ソースコード４　マウスクリックの検知

〈イベント-MouseDown〉

```
  //マウスがクリックされたら以下の処理をする
  rsp＝x
  rsp2＝y
  kazu＝kazu＋1                      //クリックカウンタを加算
  return true
```

ソースコード５　正答数のカウント

〈メソッド-kaunta〉

```
  if rsp＜＞0 then                   //マウスクリックを検知していれば
    seikai＝seikai＋1                //正答数カウンタを加算する
  end if
```

ソースコード6　結果の処理

〈メソッド-stop〉

```
dim gx,gy As integer
dim gr(3,4) As integer
dim tx,ty,i,ii As integer
//3試行連続無失策の基準がクリアされたので以下のことを行う

for tx=0 to 5                          //提示刺激ごとの正答数を合計する。
  for ty=0 to cumul
    tot(tx)=tot(tx)+kiroku(tx,ty)
  next
next

//結果を画面上に棒グラフで表示する
for i=0 to amari                       //グラフ表示のための座標を確保する
  select case i
  case 1  //第1項目
    canvas8.graphics.drawString"よわ",120+i*25,166+cumul*16
    canvas8.graphics.drawString"よわ",350+i*25,165
    canvas8.graphics.drawString str(tot(i)),129+i*25,182+cumul*16
    canvas8.graphics.drawString"刺激項目ごとの合計正答数",120+i*25,198+cumul*16
    canvas8.graphics.drawString"刺激項目ごとの正誤一覧表（黒=正答)",350+i*25,149
    canvas8.graphics.drawString"1",122,150+cumul*16  //正答数1の目盛り
                                                              (合計正答数用)
    canvas8.graphics.drawString"1",352,182  //正答数1の目盛り（正誤一覧表用)
  case 2  //第2項目
    canvas8.graphics.drawString"しう",120+i*25,166+cumul*16
    canvas8.graphics.drawString"しう",350+i*25,165
    canvas8.graphics.drawString str(tot(i)),129+i*25,182+cumul*16
  case 3  //第3項目
    canvas8.graphics.drawString"しよ",120+i*25,166+cumul*16
    canvas8.graphics.drawString"しよ",350+i*25,165
    canvas8.graphics.drawString str(tot (i)),129+i*25,182+cumul*16
  case 4  //第4項目
    canvas8.graphics.drawString"いた",120+i*25,166+cumul*16
```

```
      canvas8. graphics.drawString"いた", 350＋i＊25, 165
      canvas8. graphics.drawString str(tot (i)), 129＋i＊25, 182＋cumul＊16
    case 5　//第5項目
      canvas8. graphics.drawString"くか", 120＋i＊25, 166＋cumul＊16
      canvas8. graphics.drawString"くか", 350＋i＊25, 165
      canvas8. graphics.drawString str(tot(i)), 129＋i＊25, 182＋cumul＊16
    end select
  next
  for i＝5 to cumul step 5                    //正答数5の目盛りを付ける
    ii＝i
    canvas8. graphics.drawString str(ii), 122, (150＋cumul＊16)−ii＊14　//（合計正答数用）
    canvas8. graphics.drawString str(ii), 352, 166＋ii＊15　//（正誤一覧表用）
  next
  for gx＝1 to 5                              //提示刺激ごとの正答数合計を棒グラフで表示する
    for gy＝tot(gx) downto 1
      canvas8. graphics.FillRect 120＋gx＊25, (150＋cumul＊16)−gy＊16, 23, 15
    next
  next
  for gx＝1 to 5                              //全試行全項目の正誤状態を一覧表示する
    for gy＝1 to cumul
      if kiroku(gx,gy)＜＞0 then
        canvas8. graphics.FillRect 350＋gx＊25, 150＋gy＊16, 23, 15
      else
        canvas8. graphics.DrawRect 350＋gx＊25, 150＋gy＊16, 23, 15
      end if
    next
  next
```

ソースコード7　メニュー項目の登録

〈イベント−EnableMenuItems〉

```
//メニューの「ファイル」項目に「用紙設定」「印刷」「保存」を追加する

PageSetupMenuItem.Enabled＝True
PrintMenuItem.Enabled＝True
SaveMenu.Enabled＝true
```

ソースコード8　用紙設定

〈メニューハンドラ-pageSetupMenuItem〉

```
//メニュー―「ファイル」―「用紙設定」の作業内容
//プリンターを設定する

  Dim PS As PrinterSetup
  PS=New PrinterSetup
  If PS.PageSetupDialog Then
    PageSetupStr=PS.SetupString
  End If
```

ソースコード9　印刷

〈メニューハンドラ-printMenuItem〉

```
  Dim PS As PrinterSetup
  Dim G As graphics
  dim i,ii,gx,gy As integer
//メニュー―「ファイル」―「印刷」の作業内容
//結果表示画面をプリントアウトする

  PS=New PrinterSetup
  If PageSetupStr <> "" Then
    PS.SetupString=PageSetupStr
    G=OpenPrinter(PS)
  Else
    G=OpenPrinterDialog(PS)
  End If
  If G <> nil Then
    for i=0 to amari
      select case i
      case 1
        g.drawString"よわ",120+i*25,166+cumul*16
        g.drawString"よわ",350+i*25,165
        g.drawString str(tot(i)),129+i*25,182+cumul*16
        g.drawString"刺激項目ごとの合計正答数",120+i*25,198+cumul*16
        g.drawString"刺激項目ごとの正誤一覧表",350+i*25,149
```

```
      g.drawString"1", 122, 150 + cumul * 16
      g.drawString"1", 352, 182
    case 2
      g.drawString"しう", 120 + i * 25, 166 + cumul * 16
      g.drawString"しう", 350 + i * 25, 165
      g.drawString str(tot(i)), 129 + i * 25, 182 + cumul * 16
    case 3
      g.drawString"しよ", 120 + i * 25, 166 + cumul * 16
      g.drawString"しよ", 350 + i * 25, 165
      g.drawString str(tot(i)), 129 + i * 25, 182 + cumul * 16
    case 4
      g.drawString"いた", 120 + i * 25, 166 + cumul * 16
      g.drawString"いた", 350 + i * 25, 165
      g.drawString str(tot(i)), 129 + i * 25, 182 + cumul * 16
    case 5
      g.drawString"くか", 120 + i * 25, 166 + cumul * 16
      g.drawString"くか", 350 + i * 25, 165
      g.drawString str(tot(i)), 129 + i * 25, 182 + cumul * 16
    end select
  next
  for i = 5 to cumul step 5
    ii = i
    g.drawString str(ii), 122, (150 + cumul * 16) - ii * 14
    g.drawString str(ii), 352, 166 + ii * 15
  next

  for gx = 1 to 5                          //項目ごとの正答数合計グラフ表示。
    for gy = tot(gx) downto 1
      g.FillRect 120 + gx * 25, (150 + cumul * 16) - gy * 16, 23, 15
    next
  next
  for gx = 1 to 5                          //全試行全項目の正誤一覧表示。
    for gy = 1 to cumul
      if kiroku(gx,gy) <> 0 then
        g.FillRect 350 + gx * 25, 150 + gy * 16, 23, 15
```

```
        else
          g.DrawRect 350 + gx * 25, 150 + gy * 16, 23, 15
        end if
      next
    next
  End If
```

ソースコード10　保存

〈メニューハンドラ-SaveMenu〉

```
  fSave　//ファイルメニュー「保存」が選ばれたら、fSaveへ
  return true
```

〈メソッド-fSave〉

```
  dim hozon As folderItem
  dim dasi As textOutputStream
  dim i,ii As integer
  //メニュー―「ファイル」―「保存」の作業内容
  //結果をテキストファイル形式で保存する
  //全試行の全刺激項目について正答には1誤答には0で一覧表示される
  hozon = getSaveFolderItem("text/plain", "SerialPosition")
  if hozon <> nil then
    dasi = hozon.createTextFile
    dasi.writeLine"全試行全項目の正誤一覧（ヨコ項目、タテ試行)"
    for i = 1 to cumul
      for ii = 1 to amari
        dasi.write str(kiroku (ii,i))
        if ii = amari then
          dasi.write chr(13)
        end if
      next
    next
    dasi.close
  end if
```

参考文献

1）内田雅人，「事象間連合学習におよぼす要因の研究—コンピュータ制御プログラミングの検討—」，和洋女子大学紀要第36集（文系編），1996，pp
2）内田雅人，「ハイパーカードによる心理学実験制御プログラミング—迷路学習実験—」，和洋女子大学紀要第37集（文系編），1997，pp 1-20
3）掌田津耶乃著，「初歩から始めるREALbasic」，アスキー，1999
4）心理学実験指導研究会編，「実験とテスト＝心理学の基礎—実習編—」，培風館，1985，pp. 160-162
5）真紀俊男著，「進め！REALbasic」，毎日コミュニケーションズ，2000
6）真紀俊男著，「もっと進め！REALbasic」，毎日コミュニケーションズ，2001
7）わたなべけんいち，「REALbasicで目指せ！本物指向開発者」，MacPower，6月号，アスキー，1999，pp. 256-259
8）渡辺・角山・三星・小西編著，「心理学入門」，ブレーン出版，1987
9）「REALbasic3日本語版チュートリアル＆開発者ガイド」，アスキー，2001
10）「REALbasic3日本語版言語リファレンス」，アスキー，2001
11）「REALbasic3.5日本語版ユーザーライセンスパック」，アスキー，2001

（短期大学部日本文学科助教授）