ヒューマンインタフェース 60－1
（1995．5．17）

プログラミング操作記録による学生の行動分析

鈴木悦子・小川貴英
津田塾大学数学科

計算機教育にプログラミング演習は欠かせないものである．しかし，演習授業では教師1人に対する学生の人数が多いため学生全員に目が届かない．以前我々は，学生のエディタへのキー入力の種類や，コンパイラのメッセージから，演習中にどの学生が行き詰まっているか，あるいは教師の助言を必要としているのは誰かということを抽出し，モニターすることを試みた．本研究では，キー入力の種類ではなくキーコードを収集，解析することで，学生がどのようにエディタを使い，プログラミング演習をしているかが把握できることがわかった．

Analysis of student's behavior from operating records of programming

Etsuko Suzuki, Takahide Ogawa
Department of Mathematics
Tsuda College
Tsuda-machi Kodaira-shi, Tokyo 178, Japan

A programming writing class is indispensable to computer science education. However, it is hard to monitor all the students's progress on programming exercises in the class, when the number of students becomes large. We developed a system for monitoring the students's progress to assist a teacher. The system collects all inputs to a text editor, which students use for writing programs and invoking compiler. Our analysis of the collected data suggests that the data can indicate several behavior patterns, which suggests the type of the knowledge that students lack and the assistance that they need.

1 はじめに

計算機教育にプログラミング演習は欠かせないものである．しかし，演習中に学生全体が滞りなく問題をこなしているかどうかは，教師一人に対して複数の学生という構成になることが多く，全体を把握するのは難しい．このような状況に対処するために，我々はリアルタイムモニタシステムを作成する研究をしている．

我々は，プログラミング演習時に学生と計算機とのインタフェースのほとんどがエディタであることを利用して，既にEmacsの中に編集とコンパイルを実行できる環境を作り，コンパイルのメッセージとキー入力情報を収集し，モニターすることにより学生の状態把握を試みた.[1] この研究では，キー入力の種類(alpha-numeric(0x20 ～ 0x7e),contorol(0x01 ～ 0x19,0x7f) に分類) のみを収集したために，エディタの使い方の詳細は得られなかった．エディタEmacsをキー入力レベルで解析した研究には [2] があるが，エキスパートなユーザに対する解析であり，初心者に対して調べたものはない．本研究では，入力したキーコードを収集することによって，学生のエディタの使い方の詳細を調べた．その結果，Emacsに関しては，よく使うコマンドは何か，ウィンドウの概念，ファイル，プロセスといった概念を正しく理解しているかを推定できた．また，ファイル入力時のキーを調べることで，問題が難しいかどうかが推測できた．

以下2章では収集した操作記録について説明をし，3章では操作記録解析について述べる．

2 操作記録

Emacsは，正式名をGNU EmacsといいUNIXの上で広く使用されているエディタである．豊富な機能と拡張性を持つ高機能なエディタであるが，それゆえ，初心者にはわかりにくいという面を持つ．X Windowをサポートしていて，X Window上では，新しくウィンドウを作って起動することができる．研究対象となる学生は，ネットワークにつながれたワークステーションで，X Windowを立ちあげEmacsを使用して演習を行なう．

学生はプログラムの編集，コンパイルまでをEmacsの中で行なっている．プログラムの実行はEmacsの中からは行なわない．操作記録の収集はネットワークを介して行なわれる．学生がEmacsを起動すると自動的に操作記録が採り始められ，UDPパケットに生成され，クライアントからサーバへと送られる.(図1) サーバには，出力先の変更機能があり，操作記録を採るならファイル，モニターシステムなど他のプログラムと接続して使うならソケットを選択できる．

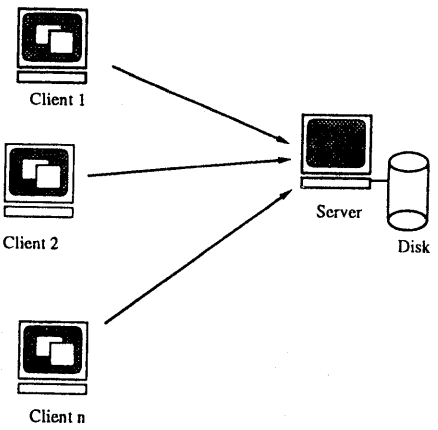

図 1: 操作記録収集

キー入力は，Emacs Lispが常に最後のキー入力を100文字を保持していることを利用し，1分に一度キー入力を調べて新しく入力された部分を採ることにした．1分に一度では，100文字以上の入力があった場合切り捨てられてしまうが，学生のタイプ速度は速いとはいえず，かつ，演習中は更に速度が落ちると考えて，1分に一度の採取とした．またファイル名はEmacsのバッファ名を，ファイルサイズはバッファサイズを利用している．

得られるパケットの内容は，以下のようになっている．

記録の種類 日付 ユーザID ワークステーション名
編集しているファイル名 ファイルサイズ
パケットID プロセスID
キー入力数 (キー入力数＞0 なら) キーの並び

記録の種類はパケット転送システムを作成した時にパケットの種類を区別するためにつけるもので今回は1種類しか採取していない．

対象とした学生は2年生26名で，1年の時にPascalの講義と演習をそれぞれ週に1時間ずつ受講した．また現在は情報科学コースに在籍し，週にCの講義と演習を1時間ずつ受講している．操作記録を収集した時期は第11回目の演習授業であり，内容はC

```
typedef struct complex{
    double  real;
    double  img;
  } COMPLEX;

addcomplex(COMPLEX x,COMPLEX y,COMPLEX *z){
  z->real = x.real + y.real;
  z->img  = x.img  + y.img;
}
```

図 2: 問題 2 の配布リストの一部

によるプログラミングである. 学生の Emacs 歴は約 1 年半である.

今回の演習問題として出した問題は以下のような構成で 3 題である. なお, 問題 2,3 に関しては実行可能なプログラムリストを紙に印刷し, 配布した.

1. n! を計算する関数 fact を再帰呼び出しを使って作りなさい.
2. addcomplex を参考にして subcomplex, mulcomplex を作りなさい. (参照 図 2)
3. insertlist を参考に, データを与えるとそのデータの要素を削除する関数 deletelist を作りなさい. 削除したリストは free すること.

90 分の授業であるが,30 分はレポート返却, 問題解説などに費やすので, 実際に学生が集中して作業できるのは 60 分くらいである. 提出期限は 3 日後であり, すべての問題を時間内で終了した学生はいなかった. 問題をやる順番は特に指定していないが半数以上が 1 と 2 は手をつけている. 実際に授業をしていて学生にとっての問題の難易度は 2,1,3 の順であるように感じられた. 2 はプログラムが部分的に分かればプログラム全体の意味が分からなくてもできる. 1 は再帰呼び出しが直感的に分かりにくいのか, 途中まで書いて手が止まる学生が多かった. ポインタは前回導入した概念だが, 前回とは異なるリスト操作を与えた. 学生にはポインタの概念はなんとなく分かっているが, プログラムが簡単には読めなかったので手がつかなかったようである.

3 操作記録の解析

3.1 タイプ速度の解析

文字数 / パケット	人数	
40 ～ 44	1	*
45 ～ 49	5	*****
50 ～ 54	1	*
55 ～ 59	3	***
60 ～ 64	0	
65 ～ 69	2	**
70 ～ 74	7	*******
75 ～ 79	2	**
80 ～ 84	2	**
85 ～ 89	0	
90 ～ 94	1	*
95 ～ 99	1	*
100 ～	1	*

表 1: 1 パケットあたりの文字数最高値

演習問題にプログラムリストを与えたものがあったので操作記録を使って学生がどのくらいの速度でタイプを打てるかを調べることができる. ただし, タイプのスピードテストをしているわけではないので正確な値は得ることはできない. ここでは 1 パケットあたりの文字数を数えている. また, 連続して入力されている同じ制御文字はオートリピートと考えて 1 と数えている.(*ex.* "C-f C-f C-f" → 1, "a a a" → 3) 各学生のタイプ速度の最高値を表 1 に示す.

この表に載った値をもっているパケットを調べたところ, 問題 2,3 の配布されたプログラムリストを入力している時であった. タイプの試験をしているわけではないので, この値が学生達の最高値であるとはいえないが, 約半数はあまり早いとはいえない.

また, 1 パケットに入った文字数をオートリピートの分も含めて数えた場合は, 表 2のようになる. 約半数が 0 であり, 実際に手を動かしているより考えている時間の方が長いと思われる. また 100 文字入ったパケットは 20 しかなく, そのうちカーソル移動 (C-p,C-n,C-f,C-b) が 50% 以上含まれているパケットは 55%, 80% 以上がアスキー文字であるパケットは 20% であった.

文字数	バケット数
0 ～ 4	932
5 ～ 9	120
10 ～ 14	123
15 ～ 19	113
20 ～ 24	85
25 ～ 29	102
30 ～ 34	92
35 ～ 39	99
40 ～ 44	104
45 ～ 49	84
50 ～ 54	59
55 ～ 59	74
60 ～ 64	55
65 ～ 69	41
70 ～ 74	24
75 ～ 79	18
80 ～ 84	11
85 ～ 89	16
90 ～ 94	14
95 ～ 99	3
100 ～	20
total	2189

表 2: バケットあたりのキー入力数

3.2 エディタに関する知識の解析

対象学生は Emacs を 1 年の演習から使用している. 入門時に基本的なコマンド約 40 種について, 授業時に説明とコマンドのインデックスを配布されている. Emacs ではコマンド関数に結びつけられ (binding) ているキーを" キー" と呼ぶが, ここでは収集したキーと区別するためにコマンドと呼ぶことにする. なお, 2 年になっても Emacs の使い方をあまり覚えていないので, この操作記録を採る前に数回に渡ってファイル操作 (オープン, 挿入, 書き出し, カット&ペースト) に関するコマンドの復習をしている.

3.2.1 ウィンドウに関する知識

この操作記録では, プロセス ID を記録しているので Emacs を二つ以上起動しているかどうかを調べることができる. 一度に起動している数は表 3のようになり, 一度に複数の Emacs を使用している学生が相当数いることがわかる.

同時に起動している数 (max)	1	2
人数	20	6

表 3: 起動数

2 つ起動している場合は,

1. 一時的にファイルを参照する
2. Emacs の中から辞書を引き参照する
3. プログラム作成を中断して他のプログラムを作成する

という 3 つの場合があった. これらを行なう理由としては,

1. バッファの切り替えができない, あるいは知らない. 一つの Emacs から複数のファイルが参照可能であるにもか変わらずその機能が使えない. (Emacs にはウィンドウという概念があり 1 つのエディタ内で複数のファイルを同じ画面に表示できる.)
2. Emacs を起動時にデフォルトの大きさ (80 × 24) にしてしまう. この大きさでは二つのファイルを見るには小さいのでもう一つエディタを起動する.

が考えられる.

1 に関しては, 1 年の時に演習で説明を受けているが, コマンドの存在を忘れてしまったか, 機能としては知っているがコマンドを覚えていないので, もう一つエディタを起動してしまうと考えられる. 2 に関しては, ウィンドウのリサイズは知っているが, デフォルトの大きさを使い慣れているのか, あまりリサイズをして使おうとしない. 筆者は演習時に質問されるたびに, 学生のウィンドウのサイズを大きくするようにしていたため, 幾人かはリサイズして使用するようになったが, 多くの学生はデフォルトの大きさで使用している.

1 年の時のプログラミングはあまり大きいファイルを作成しないので, デフォルトの大きさでも十分であったが, 作業量が増えても, そのまま使い続けるのは問題があるといえる.

3.2.2 ファイルに関する知識

表 4は学生全体で, 演習時間中の Emacs のプロセス起動した回数とその時に参照したファイルとの関係を表している. ただし, Emacs で一時的にできるバッファ (* で囲まれたバッファ名 *ex.* *scratch*)

Emcasの	ファイル数		
プロセス数	1	2	3
1	58	2	3
2	10	0	0

表 4: Emacs のプロセスと対応するファイル数

と一分以内に入力もなく参照を辞めたものは 0 と数えている．表 4は 1 回の編集で 2 つの異なるファイルを参照したものは学生全体で 2 例あり，1 つのファイルを 2 回に分けて編集した場合が 10 例あったことを示している．

1 つのファイルを 2 回に分けて編集している場合が 10 例あるが，2 例を除いては，1 番の演習問題に行き詰まって 2 番の問題を始め，後から 1 番の問題に取り組む場合であった．1 回の編集で複数のファイルを参照している場合が 5 例あるが，1 例を除いては，複数の問題に渡って Emacs を継続して使用していない．また，1 回の編集で 1 つのファイルしか参照しなかった場合を調べると，問題 2 の関数を一つ一つ別のファイルに作成している 10 例を除いていは，"1 つの問題 =1 つのファイル" となっている．つまり，ほとんどの学生が長くても一つの問題に対するプログラムを書き終えるとエディタを終了している．次のコマンドの項を見ると明らかであるが，ファイルをオープンする (C-x C-f) というコマンドを知っているにも関わらず，問題ごとにエディタを終了してしまうのは，学生にそうしなければいけないという思い込みがあると推測される．

3.2.3 コマンドに関する知識

学生が入力したコマンドのうち入力された回数が多い順に並べたものが表 5である．ここから，学生がどういうコマンドを覚えているのかがわかる．

使用しているコマンドから推測されることは以下のことである．

1. 絶対に必要なコマンドは知っている．(カーソルの上下左右，一文字削除など)
2. 最近何度か復習したコマンドはキー定義は覚えていなくても使えるようになった学生もいる
3. バインドされたキーによってカーソルを一文字ずつ移動するより，矢印キーやマウスによる移動の方が使い慣れている学生もいる

表を見れば明らかなように，移動と DEL による削除は全員が使用していて，入力されたコマンドの 8 割以上を占めている．

数は多くないが，機能を覚えはじめたカット&ペーストやファイルの挿入なども使用している．

カーソル移動はマウスでクリックすることにより移動することも可能で，マウスによる移動をすると C-x C-@ が記録される．約 $\frac{1}{3}$ の学生が使用していると考えられる．また，カーソル移動は矢印キーによる移動も可能で，矢印キーによる移動もバインドされたキーも同じ記録が残るので区別はつけられない．しかし，SUN のキーボードのファンクションキーのうち，Emacs で未定義なものを入力すると前のパラグラフにジャンプするコマンド (ESC }) が入ってしまう．このコマンドがカーソル移動のコマンドの中で行頭，行末コマンドの次に多いのは，矢印キーを入力し損なっている可能性が高い．

エディタの終了 (C-x C-c) が少ないのは終了時に Emacs から記録を取り出せないまま終了してしまうというキー入力を取り込む Emacs Lisp で記述したプログラム部分のバグであるが，終了直前にはキー入力はほとんどないため，今回の解析ではあまり問題がないと考えた．

ここには現れていないが，必要であったはずのメールを読むコマンドなどは忘れてしまっている．(実際に，これとは別の演習時間中にメールでプログラムリストを配布した時にほとんどの学生はメールを読むことができなかった．) メールの読み書きは一年の授業では，最初の導入で使わせる以外は，あまり使用していないらしい．

3.3 プログラム編集行動の分析

プログラムを入力している時の削除コマンドと移動コマンドの入力量を調べた．「プログラムを入力している時」とは，ファイルをオープンして入力をし，一回目のコンパイルを起動するまでをいうことにする．対象は，問題 1,2 の両方を演習授業中に終了している学生 11 名分に絞った．問題 2 の場合，複数のプログラムを作成する可能性があるが，一つめに入れたプログラムに対するキー入力のみを調べている．問題 1 は自分でプログラムを作成する問題であり，問題 2 はプログラムリストそのものか，数文字の変更ですむ問題である．

削除コマンドは一文字削除 (C-d, DEL) の入力量の和を d とし，移動コマンドはカーソル移動の上下

命令	バインドされているキー	全体で使用された回数	使用している人数 (%)
カーソル右	C-f	4669	100
カーソル左	C-b	3505	100
カーソル上	C-p	2558	100
カーソル下	C-n	2539	100
一文字削除 (後)	DEL	3609	100
一文字削除	C-d	670	69.2
カット	C-k	445	76.9
マウスによるカーソル移動	— (C-x C-@)	422	34.6
ペースト	C-y	98	30.8
日本語切り替え	C-\	91	34.6
行頭	C-a	70	50.0
行末	C-e	70	34.6
コマンドの中止	C-g	47	46.2
コマンド起動	ESC x	34	61.5
ファイルのオープン	C-x C-f	27	23.0
前のパラグラフにジャンプ	ESC [	22	46.2
ヘルプ	C-h	11	26.9
マーク	C-@	10	23.0
コンパイルエラー行にジャンプ	C-x ‘	8	7.7
ファイルの挿入	C-x i	8	15.4
エディタの終了	C-x C-c	6	15.4

表 5: 入力されたコマンド

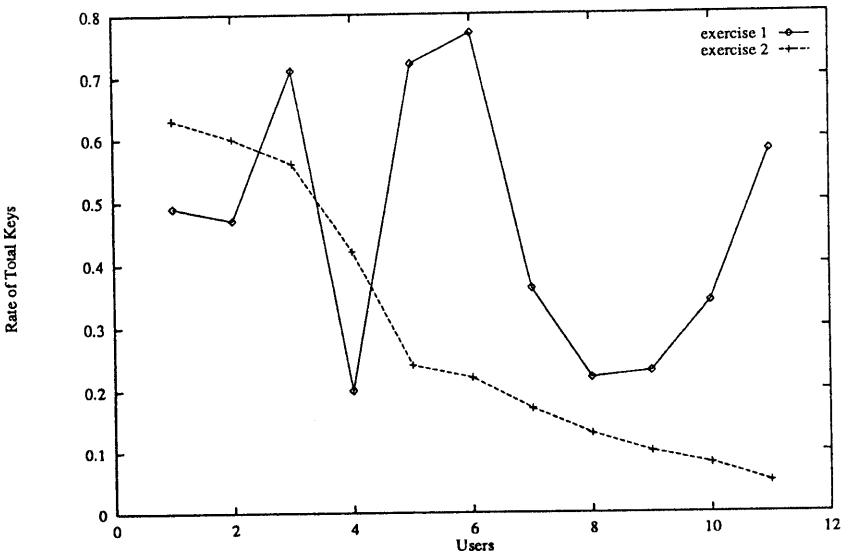

図 3: 問題 1,2 の $(d+m)/k$

左右 (C-p,C-n,C-f,C-b) の入力量の和を m とした. 入力時のプログラムに対する alpha-numeric キー入力総数を k として, $(d+m)/k$, をユーザごとに計算したものが図 3である. x 軸は各ユーザの番号を表し, 問題 2 における $(d+m)/k$ の値でソートある.

ユーザ 1,2,4 は問題 2 で, 関数定義を 2 つとも同一のファイルに作成していてキー入力全体が多いということもあるが, 1 ～ 4 のユーザはエディタの使い方が上手ではない. カット&ペーストをするコマンドを知っていても, 余分な文字のペーストをして一文字削除コマンドを連続して使用するなど無駄が多い.

また, 問題 1 に関しては, k の 4 割を越えて $d+m$ が入力されている学生達は, ほとんどプログラムをどう作っていいのかわからないで入力している. 逆に $(d+m)/k$ が 4 割以下の学生は, プログラムはある程度, 組み立てて入力している.

なお, この図とタイプ速度, 成績, ファイル作成時間などを調べたが強い相関は見られなかった.

4 まとめと今後の展望

Emacs へのキー入力を収集することで, 演習時間中に学生がどのような操作をしているかを解析し, 次のことが分かった.

1. プログラムリストを与えた場合には, 学生のタイピング速度のどれくらいかということがある程度分かる.
2. 入門教育における Emacs の利用に関しては以下のことが確認できた.
   (a) 学生は最初に習ったものを覚えている範囲でしか使わず, 新しいコマンドを自分で調べるということはない.
   (b) 直接操作 (Direct Manipulation) に近い操作, 例えばマウス, 矢印キーによる移動は忘れにくい.
   (c) ファイルやプロセスの概念を正しく理解しているかが推定できる. エディタを何回も起動し直す, あるいは, 複数起動するのは間違った思い込みがある可能性がある.
3. 演習問題の難易度がわかる. 学生が問題を理解し, プログラムを組み立てて入力しているかは, 削除と移動コマンドの入力量の割合から推測できる.

Emacs 固有の機能はともかく, ほとんどの人がどのエディタにも共通するような機能を使えないのは教育方法にも問題があるといえる. 学生に対してど

れだけサポートするかは議論の分かれるところだが, 熟練者向けのエディタを教育用に選択した場合は, 入門時にある程度時間をかけて, エディタの使用方法の教育をするべきである.

今後の課題としては, 以下のものが挙げられる.

1. Emacs のウィンドウのサイズと移動コマンドの入力量の関係の調査する. 移動コマンドの入力量は Emacs のウィンドウのサイズやファイルのサイズにも依存していると考えられる.
2. 今回の解析方法はいずれもプログラミング演習中にリアルタイムに処理が可能である. 今後は解析結果を表示するモニターシステムの構築をし, 演習に有益な情報を演習時に教師に提供する.

参考文献

1 野島久雄・阪谷徹: コンピュータネットワーク利用場面における他者の役割, 認知科学の進歩,5 特集 ヒューマンインタフェース, 1992

2 奥野 博: 画面エディタ Emacs のユーザ特性について, 第 25 回プログラミング・シンポジウム, 情報処理学会,pp164-173, 1984

3 大岩 元: 情報専門教育について「一般情報教育」, 情報処理,Vol.32 No.11,pp1184-1188, 1991

4 Richard Stallman:GNU Emacs Manual, 1985 (邦訳 竹内郁雄, 天海良治:GNU Emacs マニュアル, 共立出版, 1988)

5 鈴木 悦子, 小川 貴英: 効果的なプログラミング実習のための試み, 第 34 回プログラミング・シンポジウム, 情報処理学会,pp53-60, 1993

6 矢吹 道郎, 宮城 史朗: 初めて使う GNU Emacs, 啓学出版, 1992