実践報告・資料

スクリーン・エディタによるプログラム資料作成

北九州職業訓練短期大学校　関谷　順太

Making the Program's Documents by the Screen Editor

Jyunta Sekiya

要　約　ソフトウェア工学の概念であるトップダウン設計と段階的詳細化法に基づく構造化プログラミングのパラダイムを習得することを目標にプログラム資料作成でのエディタの使用を試みている。「外部仕様を決め、プログラムを設計し、コーディング、テスト、評価する—これらの各工程でドキュメントを作ってそれがレポート（成果物）である。」と指導してきたが、実際にはこの順序に従って文書を紙に書く学生が少なく、すぐに画面に向かってプログラムの入力を始める。流れ図はコーディングの前に書くのではなく、完成後のソースプログラムを見て書く者が多かった。この現実を見て、「直接、エディタを使って仕様をまとめ、処理手順を入力して、それらを見ながらコーディングを行う。」方法を提案した。その他のレポート項目も文章と略図をすべてコメントでの入力とした。キー入力が増えたので少し抵抗があるが、分かりやすいプログラムと読みやすいレポートができて好評である。日本語入力が増えるため、ブラインド・タッチができていない場合は、キー入力を別途練習するのが望ましい。

I　はじめに

ソフトウェア工学は、ソフトウェアの品質向上とソフトウェア技術者の生産性向上とを目標にして提唱され、現在のソフトウェア開発や保守の基礎となっている[1]。

ソフトウェア作成の演習に、この考え方を強調するため、著者は演習レポートの項目を次のように例示している。

① 要求定義
② システム設計
③ プログラム設計
④ 検証・テスト
⑤ 結果と考察
⑥ 性能評価・作成工数・反省

学生の提出するレポートは、上記項目の誤解を始め、日本語の文章になっていないものや、誤字などがある。開発のあらゆる段階にドキュメント（文書）が必須であることを説明しているが、なかなか、理解と実行が難しい。小さなプログラムでも、トップダウン設計と段階的詳細化法に基づく構造化プログラミングのパラダイムを習得することを目標にして2年間が過ぎた。

わかりやすい文書を作成するには、「紙に手書きするのではなく、スクリーン・エディタを使う。」との結論に達して、試行しているので、報告する。

II　従来の手書きのプログラミング演習と問題点

1　プログラミングの工程[2]

プログラミングには次の工程とドキュメントがある。

① プログラム設計：モジュール構成図,
　インタフェース仕様書など

論文受付日　1992.6.15

② モジュール設計：モジュール仕様書（データ構造と処理手順）
③ コーディング　：ソース・リスト
④ デバッグ　　：単体テスト計画書・報告書
⑤ 結合テスト　　：結合テスト計画書・報告書

プログラムの規模が小さいときは、①と②、④と⑤とは一つになるなどの違いはあっても、論理的な作業としては、このようになる。③はプログラム言語により異なるが、それ以外は言語に強くは依存しない。

2　従来の紙に手書きをする方法の理由

従来はプログラム設計作業とその結果を紙に書くことを求めていた。それから、コーディング・シートに手書きし、それを机上デバッグし、最後にコンピュータに入力して、デバッグする。まだコンピュータが高価でターンアラウンド・タイムが半日もかかるような時代で、誰もがコンピュータを自由に使えなかった状況では、早くプログラムを完成するために、必要であった。

また、パソコンやワープロが普及しておらず、日本語の文書がコンピュータでは扱かいにくかった。

3　処理手順や流れ図の作成

学生の演習で、最初は文法の学習が必要で短く簡単なプログラムが多く、処理手順や流れ図（資料）の必要性が少ない場合もあるが、続くアプリケーションプログラム[3]やアルゴリズムの学習では、資料が必要になる。しかし、学生は「始めにこれらを手書きしても、デバッグで変更が起こり書き換えになります。それが面倒だから、結局レポートのために後でまとめます。」としているものが、クラスのほとんどであった。

流れ図についていうと、時間をかけて書いているにも係わらず、単なるソースリストの書き写しでしかなく、見る気がしないものがあった。「概略流れ図がよい。」と注意していた。また、課題がそれぞれの教科から出されてレポート作成が毎日続くような状況があり、内容の少ない、書きなぐりのものもあった。

4　コーディングの仕方

コーディングでは、「紙（コーディング・シート）に書いた方が良い。」と言っていた。しかし、この紙はほとんど使われずに、材料庫に相当の量が残っている。学生が直接スクリーン・エディタで入力しているからである。

エディタの編集機能が、既存の手持ちプログラムの流用を含めてプログラミング作業を助けている。挿入・移動・書換えが自由に早く出来るのだから、わざわざ、紙に書く必要が無い。ブラインド・タッチではなくキーを見ながらでも、画面を見て考えることができるようになっているからである。

5　デバッグと結合テスト

デバッグと結合テストはコンピュータを使って行う。文法エラーについてはコンパイル・リストを見ながら、ソース・プログラムを修正・追加していく。マルチ・ウインドウが普及したので、一旦、リストをプリントしてチェックする必要性が少なくなった。それでも、廃棄シートの山を作っている。

今年度から、Ｃ言語導入教育では、コンパイル・リストのエラーメッセージをソースのファイルに複写することや、プログラム単位の考察を最後にコメントで入れるように指導している。「エラーの状況を把握して、何度も同じ間違いを繰り返さないようにしなさい。」とはいうが、自分用ノートにまとめている学生は、ほとんど見掛けない。

6　現状の問題点

1）「コンピュータを使いながらも、その能力を活かしていない。」

エディタの編集・検索などの機能は、プログラミングの思考を助ける道具として素晴らしい。それらを、ほとんど使っていない。（エディタを単なる入力・出力手段としか考えていない。）

この理由は、『プログラムの資料を紙に書くのではなく，直接エディタで作成する方が良い。』とは考えていなかったからである。勿論、これはプログラムの規模や内容によって異なる面があるであろうが。

2）「著者が現状を正しくとらえていない。」

これは自戒であるが、自分が体験した手書きを学生にも要求していて、現状の認識がずれていた。学生が、実際の課題をこなすための工夫を認めていなかった。

原因の一つは、キーボードアレルギーとは言わないにしても、コンピュータを楽に自由に使えるように、キー入力を練習していなかったことにある。

Ⅲ　スクリーン・エディタを使ったプログラム資料作成

1　考え方

プログラミング演習において、エディタをより広範囲に使うことにより、トップダウン設計と構造化プログラミングを学習すること目的とする。具体的には従来の紙に手で書く方法　レポート用紙に（後で）まとめる―でなく，エディタを使ってプログラム設計を行い，そのテキストを参照して（後戻りも自由にして）コーディングを直接エディタで行う。デバッグでの状況は、コンパイルリストからソース・プログラムにエラーメッセージを複写する。最後の考察もコメントとして入力する。

この方法は、キー入力が増えるため、ＶＤＴ作業時間が長くなる。疲れないで楽にエディタを使うためには、ブラインド・タッチが望ましい。

2　ＢＡＳＩＣでの例

図１．１は、ＢＡＳＩＣでの散布図作成プログラムで作成の途中段階の画面のハードコピーである。始めに処理概要を箇条書きで入力し、その後データの説明と、コーディングとを内蔵のエディタで行っている。（パソコンはＦＭ　Ｒ－６０でＦＢＨＧインタプリタを使用。）図１．２は完成したプログラムの一部を示す。

コーディングとデバッグの作業で、処理手順の変更などがあるので、その面では少しせわしいところがある。しかし、それを補っても余りがあるのは「始めに全体としての構造を考え、段階的に詳細化していくことから、途中での大幅な変更を避けられること」が大きい。

```
1110 '   SX0,SY0: Screen Coordinate Origin
1111 '   SX2,SY2: Screen Coordinate Maximum
1120 '   X0,Y0:   Data Value Coordinate Origin
1140 '   RX,RY:   Data Value Range
1180 '
1200 '       1. 初期作業
1300 '           1.1 制御変数の設定
1400 '           1.2 パラメータの計算
1500 '           1.3 タイトルを書く。
1600 '           1.4 枠,軸,目盛りを書く。
1700 '           1.5 データ数を読む。
1800 '       2. データのPLOT，統計諸量の計算
1900 '           2.1 データの読み込み
2000 '           2.2 座標計算
2100 '           2.3 PLOT
2200 '           2.4 統計諸量の計算
2300 '       3. 統計諸量の表示
2400 '
1310  SX0=100: SY0=600: SX2=600:SY2=100
1320  X0=125:X2=155: Y0=25: Y2=55
1410  SRX=SX2-SX0:SRY=SY2-SY0: RX=X2-X0: RY=Y2-Y0
1510  TITLE$=" 身長と体重の散布図（ｎ＝２０）"
1520  symbol@ (200,10),TITLE$,2,2,3
1610  line (SX0,SY0)-(SX2,SY2),pset,5,b
```

図１・１　作成途中の画面

```
1000 ' 統計データ処理 No.4 散布図の作成 1992.5.18 関谷
1010 ' SCAT2.BAS
1020 '   SX0,SY0: Screen Coordinate Origin
1030 '   SX2,SY2: Screen Coordinate Maximum
1040 '   X0,Y0:   Data Value Coordinate Origin
1050 '   X2,Y2:   Data Value Coordinate Maximum
1060 '   RX,RY:   Data Value Range
1070 '   SRX,SRY: Screen Coordinate Range
1080 '   TITLE$:  Title of  Diagram
1090 '   N:       Number of Data Set
1100 '   XMIN,XMAX:Range of Data X
1110 '   YMIN,YMAX:Range of Data Y
1120 '
1130 '       1. 初期作業
1140 '           1.1 制御変数の設定
1150 '
1160  SX0=100: SY0=600: SX2=600:SY2=100
1170  X0=125:X2=155: Y0=25: Y2=55
1180 '
1190 '           1.2 パラメータの計算
1200 '
1210  SRX=SX2-SX0:SRY=SY2-SY0: RX=X2-X0: RY=Y2-Y0
1220 '
1230 '           1.3 タイトルを書く。
1240 '
1250  cls
1260  TITLE$="身長と体重の散布図（ｎ＝２０）"
1270  symbol@ (200,10),TITLE$,2,2,3
1280 '
1290 '           1.4 枠,軸,目盛りを書く。
1300 '
1310  line (SX0,SY0)-(SX2,SY2),pset,5,b
1320  for I=0 to 10: SX=SX0+I*SRX/10
```

図１・２　完成したプログラム

図１　ＢＡＳＩＣの例

3　Ｃ言語での例

図２は素因数分解のプログラム作成演習（Ｃ言語）のレポートの例を示す。先ず、プログラム仕様や処理手順をコメントにしてエディタＲＥＤ２を使ってソース・プログラムの先頭に書いた。その後に、Ｃ言語でコーディングを行い、デバッグ完了後に考察を追加している。

4　エディタの利用と効果

エディタの良いところは，テキスト(日本語を含む)

の編集機能が便利で、追加・移動・複写・削除・検索・置換などの高速処理である。複数のファイルをマルチウィンドウで表示し、参照と更新ができる。この便利な道具を使わずに、紙と鉛筆・消しゴムで学習者の時間をかけるプログラミング演習はもう勧められない。

ＶＤＴ作業は、日本語の入力が増えた分だけ多くなるが、学生の反応はおおむね好評である。「分かりやすいプログラムができることが、一番効果があります。」と言う。作成者の意図が段階的に表現されているからであろう。指導と添削する著者にとっても格段に、分かりやすくなった。

エディタで、流れ図・処理手順・データ構造を示すための図形が同時に扱えると、更に便利になると思う。現在は、モジュール構造図を90度回転して、メイン(木の根)を左端に書き、サブルーチン・関数(枝)をその右に書いている。

```
/* insu.c
正の整数を読み、その素因数分解を出力するプログラム
情報システム系 92633　神田一成。1992.7.24

プログラム仕様
 入力 :正の整数t
 処理 :素因数分解(20回反復)
 出力 :素因数iを表示

処理手順
 1. プログラムと入力の説明文を表示する。
 2. 整数tを入力する。
 3. sが1から20の間,次の事を繰り返す。
 3.1　初期値のセット(素数i=2;商k=t++;flag=0;)
 3.2　因数分解する数値(商)kを改行して表示する。
 3.3　k が1より大きい間,次の事を繰り返す。
 3.3.1　kを素数iで割った余りが0ならば、
 3.3.1.1　flagが1ならば、"×"を表示する。
 3.3.1.2　素因数であるiを表示する。
 3.3.1.3　新しい商k=k/iを求め,flag=1とする。
 3.3.2　そうでなければ次の素数i=i+1とする。

考察 素因数の計算はうまくできた。
間に'×'を入れるのは,教えてもらいながらした。*/

#include <stdio.h>
void main()
 {
    int i,k,s,t,flag;
    printf("insu.c 素因数分解 92633 神田一成¥n");
    printf("何か整数を入力してください。");
    scanf("%d",&t);
    for (s=1;s<=20;s++) {
        i=2;k=t++;flag=0;
        printf("¥n%d=",k);
        while (k>1) {
            if (k%i==0) {
                if (flag==1)
                    printf("×");
                printf("%d",i);
                k=k/i;flag=1;
            }
            else
                i=i+1;
        }
    }
}
```

図２　Ｃ言語での例

５　キー入力練習[4][5]の必要性

ＶＤＴ作業の大きな問題点である、目や首・肩・腕・指等の障害の起きる可能性を減らすために、キー入力練習を必要とする。（作業時間の制限を守ると共に疲労の発生を少なくするため。）

タイピングは、プログラム作成に限らず、これからの事務作業での必須の技能の一つである。短大でも初心者にはきちんと指導すべきである[6]。

このタイピング練習については、プログラミングの演習とは分けて考える必要がある。ブラインド・タッチの基本練習をしたうえで、毎日の30分の練習を３か月程続ければ、楽にキー入力ができるようになり、少なくとも手書きと同じくらいの速さになるはずである。個人差があろうが、ブラインド・タッチを続けて行けば、手書きの２倍（６０字／分）にはなれるという。

Ⅳ　おわりに

ソフトウェア開発にコンピュータを使って高生産性と高品質の製品を作るためのＣＡＳＥの道具が、いくつか実用化されている。当短大では、富士通のＹＰＳの部分的な利用などを検討している。このほかにＰＡＤＥＴもある。日本語でのプログラミングとして、パブリックＭｉｎｄなどがフリーソフトで流通しつつある。

しかし、短大での基礎的なプログラミング演習での資料作成は、エディタを使うと効率化できる。現在、３ヵ月の試行であるが、好評であるので報告した。ＶＤＴ作業に伴う疲労は、キー入力の訓練をすることで改善されると思う。

最後にこのテーマを進めるにあたり、助言を頂いた情報システム系の平澤・庫本先生と、実際に取り組んでくれている学生の皆さんに感謝します。

[参考文献]

（１）野口正一・中森真理雄：大学等における情報処理教育の諸問題，情報処理，1990年10月，pp. 1385-1386

（2）国友義久：効果的プログラム開発技法〔第3版〕，近代科学社，1988年，pp.57-62
（3）国友義久：同上，pp. 162-184
（4）増田忠：キーボードを3時間でマスターする法，日本経済新聞社，1987年，pp.1-76
（5）「技能と技術」編集部：タイピングの訓練をどう進めていますか？，技能と技術，1992年5月，pp.40-42
（6）大岩元：一般情報教育，情報処理，1991年11月,pp.1184-1188