ソフトウェア工学 98−10
（1994．5．25）

ソフトウェアの信頼性に影響を及ぼす人的要因の一実験

高橋宗雄　古宮誠一　三宅武司

桐蔭学園横浜大学工学部　ＩＰＡ技術センター　ＮＴＴソフトウェア研究所

ソフトウェアの信頼性に影響を及ぼす人的要因は多種多様であり，ソフトウェアの信頼性はこれらの要因の複雑な相互作用により作り込まれる。したがって，人的要因によるエラー混入のメカニズムが解明できれば，それに基づいてソフトウェアの信頼性向上技術の開発や改善を行うことができる。ソフトウェアの信頼性に影響を及ぼす人的要因を解明しようとする試みはこれまでにもいくつか行われている。しかし，それらのほとんどは仮説の提案にとどまっており，ソフトウェアの開発実験により実証されたものは少ない。仮説は実証されて始めて意味があり，実験を通して仮説を実証し，その結果をソフトウェアの信頼性向上技術の開発に結び付けることが重要である。そこで，本稿では品質工学的アプローチに基づいて，人的要因の相互作用を分析するための人的要因モデルを提案する。さらに，このモデルに基づいてコードレビューを対象とした実験モデルを設定し，コードレビューにおける人的エラーと人的要因の関係を統計的手法により分析した結果を示す。

AN EXPERIMENT ON HUMAN FACTORS WHICH AFFECT SOFTWARE RELIABILITY

Muneo Takahashi　Seiichi Komiya　Takeshi Miyake

Toin University of Yokohama　IPA Software Technology Center　NTT Software Laboratories

1614 Kurogane-cho, Midori-ku, Yokohama-shi 225 Japan　3-1-38 shibakoen, Minato-ku, Tokyo 105 Japan　3-9-11 Midori-cho, Musashino-shi, Tokyo 180 Japan

Software reliabirity results from the mutual actions among various kind of human factors. Therefore, if the mechanisms of the error occurrence caused by the human factors can be found, software reliability technologies can be developed or improved based on those findings. Some attempts to find the human factors have been made so far. However, the most of those have been left as hypotheses. It is important to verify the hypotheses through experiment and to bind the verification results to the development of the software reliability technologies. In this paper a human factor model for analyzing the mutual action among human factors is proposed based on the concept of quality engineering. An experimental model obtained from applying the human factor model to the code review is also proposed and the relationships between human errors and human factors are statistically analyzed by using it.

1．はじめに

ソフトウェアの信頼性に影響を及ぼす人的要因は多種多様であり，ソフトウェアの信頼性はそれらの要因の複雑な相互作用により作り込まれる。したがって，人的要因によるエラー混入のメカニズムが解明できれば，それに基づいてソフトウェアの信頼性向上技術の開発や改善を行うことができる。これまでにも，ソフトウェアの信頼性に影響を及ぼす人的要因を明らかにする試みはいくつか行われている[1)-7)]。しかし，これらの試みは必ずしも十分なものではなく，実用上解決しなければならない問題も多い。その第一は，従来の提案が個々の人的要因に着目したものであり，人的要因相互の関係はほとんど議論されていないことである。第二は，それらの提案は仮説としての提案であり，実際のソフトウェア開発データによって検証されたものが少ないことである。仮説は検証されて始めて実際的な意味をもち，有用なものとなる。

第一の問題に対しては，多種多様な人的要因を体系的に検証するための基本となる枠組みを与えて，ソフトウェア開発の種々の局面にその枠組みを統一的に適用しながら仮説の検証を行い，人的要因の相互関係や影響度を明らかにするというアプローチが有効である。そして，この結果をソフトウェア信頼性向上のための有効な技術開発に反映していくことである。第二の問題については，過去の開発データを用いた従来の経験論的アプローチに限界があると考えられるので，ソフトウェアの開発実験を行い，計画されたデータ収集と分析により検証を行うのが効果的であり，技術開発へ反映させ易い。

そこで本稿では，これらの問題の解決を目指して品質工学的アプローチを採用し，これに基づいて人的要因の相互作用を分析するための人的要因メタモデルを提案する。さらに，このメタモデルの適用例として，コードレビューとソフトウェア設計の二つの局面を対象とした実験モデルによる分析実験の結果について報告する。

2．人的要因モデルの設定

ソフトウェアの信頼性に影響を及ぼす人的要因をＫＪ法により抽出し，要因連関図にまとめたものを図１に示す。これから，人的要因は多種多様であり，かつそれらの相互関係は多重構造になっていることが分かる。たとえば，エラーの生成は設計不足と関連が強く，設計不足の原因は個人の経験や知識が関係している。さらに，経験や知識はストレスとか思考様式により影響を受けるといった具合である。このように，人的要因は複雑に関連し合う多重構造をなしているが，我々は人的エラーの生起に対する影響が直接的か間接的かに着目して，次のような二つの要因からなる２重構造モデルを仮定した（図２）[8)]。

一つは，人の内部にあって人手による作業にエラーを起こさせる直接の原因となるものであり，これを素因と呼ぶ。もう一つは人の外にあって，素因の影響を強めたり弱めたりする働きをするものであり，これを誘因と呼ぶ。素因は直接制御することが不可能または困難な要因であり，誘因は

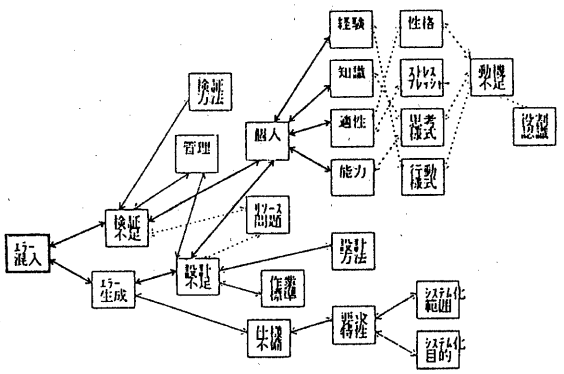
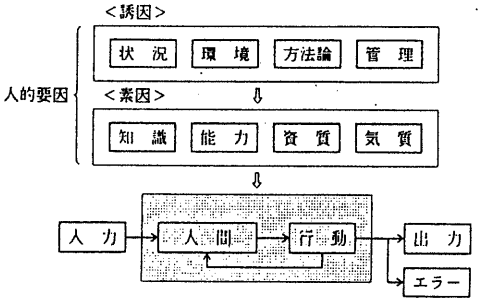


図１　人的要因の連関図（例）



人力　：仕様，プログラムなど
出力　：設計仕様，プログラム，テストケースなど
エラー：アルゴリズム誤り，テストケースもれなど
知識　：言語知識，業務知識，ＯＳの知識など
能力　：思考力，理解力，分析力，発見能力，表現力など
資質　：思考の柔軟性，リーダシップ，協調性など
気質　：外向的，内向的，慎重，粘着など
状況　：単調作業，きつい線表，コスト不足，プレッシャーなど
環境　：作業環境（騒音など），体調（疲労，寝不足など）など
方法論：技法，ツール，プロセスなど
管理　：チーム編成，動機付け，役割認識

図２　人的要因モデル

制御可能な要因である。素因と誘因を合わせたものを人的要因と呼ぶ。人的エラーと人的要因，すなわち素因・誘因との関係を表1に示す。表1における○印は人的エラーの諸現象と素因・誘因の対応関係を表している。そして，これにはそれぞれの対応関係を説明する重みをもった関数のようなものが存在すると仮定する[9]。

表1　人的エラーと人的要因との関係

要因 / エラー	素因1	素因2	… 素因n	誘因1	誘因2	… 誘因n
エラー 1	○	○		○		
エラー 2		○				
エラー 3			○		○	○
・ ・ ・						
エラー p		○			○	

3．実験モデルの設定

提案した人的要因モデルは，ソフトウェア開発のあらゆる局面に適用できるメタモデルである。このメタモデルを用いて，人的要因を検証するためには具体的なインスタンスを設定して実験を行う必要がある。インスタンスはそれがどのように利用されるかを想定して設定されなければならない。このインスタンスを実験モデルと呼ぶ。たとえば，ソフトウェアの設計，設計レビュー，テストケース設計などを対象とした実験モデルが考えられる。

ソフトウェア開発における人的エラーは，開発工程よってエラーの現象が異なる（表2）。したがって，実験モデルは，ソフトウェア開発のどの工程あるいはどの現象を対象とするかによって，対応する素因・誘因が異なることが予想される。このため，ソフトウェア開発で重要かつ人的要因の影響が大きいと考えられる工程や現象から実験モデルを設定し分析を進めることが効果的であり技術開発のインパクトも大きい。

分析データは実験計画法の直交表に基づいて，ソフトウェア開発実験により収集し，分析は統計的手法を用いて行う（表3）。

表2　人的エラー（例）

工　程	エラーの例
要求分析[10]	・曖昧な要求 ・不完全な要求 ・追跡不可能な要求 ・対象外の要求 ・矛盾のある要求
設　計[11] （プログラミング）	・要求機能の未定義 ・初期設定誤り ・名前の参照誤り ・データの定義不正 ・論理の誤り ・パラメータ の受渡し誤り ・エラー条件の不備
試　験	・条件組合せの抜け ・値域や境界の考慮不足 ・例外処理の抜け ・試験項目の不備 ・結果の確認誤り
運　用[12] （保守）	・バージョン誤り ・ディグレード

表3　分析方法

データ収集 / 手法	統計的手法	解析的手法
過去のプロジェクトから収集 (empirical study)	収集工数　：小 必要情報量：少 作業工数　：小 分析精度　：低	収集工数　：小 必要情報量：多 作業工数　：大 分析精度　：中
実験から収集 (experimental study)	収集工数　：大 必要情報量：少 作業工数　：小 分析精度　：中	収集工数　：大 必要情報量：多 作業工数　：大 分析精度　：高

4．メタモデルを用いた実験例

ソフトウェア開発において機械化が難しく人的要因の影響が大きいアクティビティはレビューと設計である。そこでまず，レビューを対象に実験モデルを設定した。

レビューに関与する人的要因を解明するため，コードレビューにおける人的エラー（埋め込みエラーの見逃し，指摘間違い）と人的要因（査読者の知識，能力の個人差）との関係に着目した実験モデルを設定した（図3）。そして，このモデルを用いて，プログラムに既知の論理エラーと構文エラーを埋め込み，それをテスト形式で検出する実験を通して，人的エラーに有意な影響を及ぼす素因・誘因を明らかにする実験を試みた。

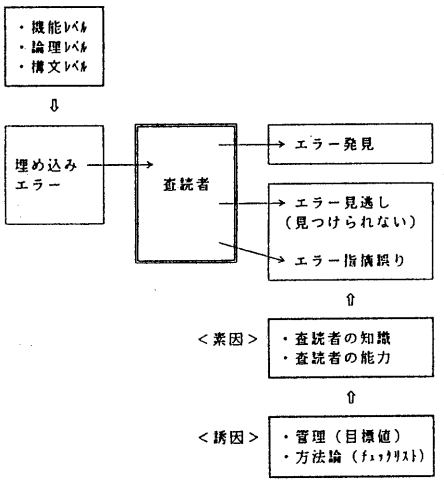


図3　実験モデル1

(1)　問題の選定

問題分野の知識に強く依存するような問題や複雑な言語機能を用いて記述しなければならない問題は実験の目的から不適切である。そこで，日本語で記述されたプログラム仕様から一般的な知識で問題の意味が理解できる「数値の大小比較」の問題を選定し，C言語で記述した（行数26行）。

(2)エラーの埋め込み

エラーについては，局所的な理解で分かるもの（大小比較エラー）とプログラム全体を理解しなければ分からないもの（初期設定エラー）の2種類の論理エラーと言語の構文エラーを合わせて10個（論理エラー6個）埋め込んだ。

4．1　第1ステップの実験

第1ステップの実験は，人的エラーに影響を及ぼす素因を明らかにすることを目的として実施したものである。この実験では，誘因は定数と見なして，素因の影響を調べた。

(1)　素因の仮定と測定

素因としては，知識（C言語，数学）および能力（思考力，理解力，問題発見力，表現力）を選定し，被験者の自己評価に基づいて，“ある”，“普通”，“ない”の3段階で数値化した。

(2)　被験者の選定と実験時間

C言語の教育を1年間受けた工学系の学生64名を対象にして，30分の時間内に何個埋め込んだエラーを発見できるかを調べる実験を行った。実験では，C言語で記述したプログラムと日本語で記述した簡単なプログラムの仕様を与えた。

(3)　実験の結果

分散分析の結果を表4に示す。表4から以下のことが推察される。

a）C言語の理解度は，論理エラーの見逃しに有意な影響を及ぼす。すなわち，C言語を理解していると答えた学生は論理エラーの見逃しと指摘間違いが少ない。

b）数学の得意／不得意は構文エラーの見逃しに影響を及ぼす。すなわち，数学が得意であると答えた学生は構文エラーの見逃しが多い。数学が得意な学生は論理的な矛盾に興味あり，構文エラーの発見がおろそかになったとも考えられるが，偶然の結果とみたほうがよいかもしれない。このことは，実験後のインタビューで，学生は埋め込まれたエラーは論理エラーだけだと思っていたといいうことからも裏付けられる。

c）矛盾／欠陥発見能力は構文エラーの見逃しに影響を及ぼす。すなわち，この結果は発見能力があると構文エラーの見逃しが少ないということを意味していることになるが，構文エラーの発見には言語の知識が前提となるはずである。

d）論理エラーの見逃しを，プログラムの局所的な理解で分かるエラーの見逃しと全体を理解しなければ分からないエラーの見逃しに分けた場合の分散分析結果を表5に示す。表5から，C言語の理解度は両者に影響を及ぼすことが分かる。矛盾／欠陥の発見能力はプログラム全体の理解を必要とするエラーの見逃しに関係する。

表4，表5から統計的には，C言語の理解度，数学の得意度，発見能力がエラーの見逃しに影響を及ぼすという結果が得られた。この中で，言語の理解度は危険率1％で有意であり，コードレビューの重要な素因と考えられる。

表4　分散分析結果（1）

素因 ＼ ヒューマンエラー		論理エラーの見逃し	構文エラーの見逃し
知識	C言語	8.49 **	0.25
	数学	0.36	3.17 *
能力	思考力	0.71	2.06
	理解力	0.71	0.91
	分析力	0.91	1.56
	発見能力	1.11	3.91 *
	表現力	0.63	0.53

*　危険率5％有意
**　危険率1％有意

表5　分散分析結果（2）

素因 ＼ ヒューマンエラー		大局エラーの見逃し	局所エラーの見逃し
知識	C言語	7.14 **	6.11 **
	数学	0.35	0.14
能力	思考力	2.09	0.50
	理解力	0.24	0.04
	分析力	0.35	0.03
	発見能力	4.18 *	0.03
	表現力	1.32	0.26

*　危険率5％有意
**　危険率1％有意

4．2　第2ステップの実験

第1ステップの実験では図3の実験モデルで誘因を固定して実験を行った。これによると，論理エラーの見逃しに影響を及ぼす人的要因（素因）としては，言語の理解度が危険率1％有意で重要な要因であることが分かった。そこで，素因として言語の理解度を選定し，誘因を含めて分析する第2ステップの実験を行った。

(1)　誘因の仮定と測定

誘因は，固有知識に基づいて論理エラーの見逃しへの響度が大きいと想定されるものを選んだ。

実験では，図3に示したように「論理エラー摘出目標値の有無」と「チェックリストの有無」を仮定した。すなわち，この実験では埋め込んだエラーの種別，言語理解度の高低，摘出目標値の有無およびチェックリストの有無という四つの要因を選定し，いずれも2水準として実験を行った。この場合，すべての要因水準の組合せは$2^4=16$であり，実験回数は16回となる（表6）。しかし，直交表により実験を割り付けると，表6で＊印の付いた8回の実験を行えばよいことになる。

表6　実験の割り付け

実験	入力		言語		目標値		ｶﾞｲﾄﾞﾗｲﾝ		観測値
	論	構	高	低	有	無	有	無	
1*	○		○		○		○		
2	○		○		○			○	
3	○		○			○	○		
4*	○		○			○		○	
5	○			○	○		○		
6*	○			○	○			○	
7*	○			○		○	○		
8	○			○		○		○	
9		○	○		○		○		
10*		○	○		○			○	
11*		○	○			○	○		
12		○	○			○		○	
13*		○		○	○		○		
14		○		○	○			○	
15		○		○		○	○		
16*		○		○		○		○	

*　直交表に基づく実験の割り付け

(2)　被験者の選定と実験時間

言語コンパイラ，ＯＳ，交換ソフトウェアなどの実用プログラムの開発経験が3～10年の人を20名選定した。実験時間は15分で行った。

(3)　実験データの収集

被験者20名を彼らのC言語能力の高低に応じて8名をＧ1～Ｇ4（高い能力），12名をＧ5 Ｇ5～Ｇ8（低い能力）に割当て，データを収集した（表7）。

(3)　分析の結果

分散分析の結果を表8，表9および表10に示す。表8は論理エラーの実験データを，表9は構文エラーのそれを，表10は全体のそれをそれぞれ分析したものである。

表7　実験データ

実験	人的要因			人的エラー		
	言語	目標	CL	論理	構文	全体
G1	高い	あり	あり	1.0	0.0	1.0
G2	高い	あり	なし	2.0	1.5	3.5
G3	高い	なし	あり	0.5	0.5	1.0
G4	高い	なし	なし	3.0	2.5	5.5
G5	低い	あり	あり	3.0	3.3	6.3
G6	低い	あり	なし	3.7	3.0	6.7
G7	低い	なし	あり	3.0	2.7	5.7
G8	低い	なし	なし	3.5	4.0	7.5

CL:ﾁｪｯｸﾘｽﾄ

表8　分散分析結果（論理エラー）

	不偏分散	F値	寄与率
言語能力	5.617	17.390*	58.0
数値目標	0.011	0.034	0.1
ﾁｪｯｸﾘｽﾄ	2.765	8.560	28.6

* 危険率5％有意

表9　分散分析結果（構文エラー）

	不偏分散	F値	寄与率
言語能力	9.040	22.157**	66.2
数値目標	0.453	1.110	3.3
ﾁｪｯｸﾘｽﾄ	2.527	6.194	18.5

** 危険率1％有意

表10　分散分析結果（全体）

	不偏分散	F値	寄与率
言語能力	28.880	24.207**	64.4
数値目標	0.627	0.526	1.4
ﾁｪｯｸﾘｽﾄ	10.580	8.868*	23.6

* 危険率5％有意
** 危険率1％有意

分析の結果を要約すると以下のようである。

a）言語能力は論理エラーおよび構文エラーの検出に有意な影響を与える。すなわち，言語の能力が低い人ほどエラーを指摘きないかあるいは指摘誤りを起こしやいといえる。

b）目標値は論理エラーおよび構文エラーの検出に有意な影響を与えるとはいえない。レビューで目標値があると，エラーの指摘に効果が期待できそうであるが，実験の結果はそうならなかった。このような結果になった一つの原因は，レビュー時間が関係しているのではないかと推察される。レビュー時間が15分と短かったので，時間内に見つけられるエラーだけが指摘され，目標値を達成させるための努力を十分行えなかったのではないかと考えられる。

c）チェックリストは，論理エラーの検出には有意な影響を与えるが，構文エラーの検出に対しては有意でないという結果となった。これは，言語能力が高い人にとっては，チェックリストがなくてもエラーは見つけられるし，逆に言語能力が低い人にとっては，チェックリストがあっても言語の知識がないと意味がよく分からないため，その効果が期待できなかったということであろう。

今後さらに実験を継続して，レビューのやり方やチェックリストの内容とエラー検出との関係を明らかにしていく必要がある。

d）言語能力の人的エラーへの影響度は，論理エラーの場合約60％，構文エラーの場合約70％である。言語能力の影響度は，構文エラーに対するほうが論理エラーに対する場合に比べて10％ほど高い。これは，構文エラーの検出が論理エラーのそれに比べて，より高い言語能力を必要とするからであり，妥当な結果といえる。

e）チェックリストの人的エラーへの影響度は，論理エラーの場合約30％，構文エラーの場合約20％である。

6．おわりに

古くて新しいテーマ「ヒューマンファクターの分析」について，一つの考え方（モデル）を提案し，それを用いた実験例を示した。今後さらに，実験を継続し実証データを蓄積して，制御可能な人的要因相互の関係を明らかにしていく予定である。実験により有用な結果を得るためには，以下の観点に留意して研究を進めることが必要であると考える。

(1) 人的要因の分析は対象が人間であるから，ノイズと要因による影響を詳細に見極めることは困難である。したがって，特定の個人を対象として詳細に調査するよりは，多数の人間を対象として実験によりデータを収集して統計的に処理し，その結果をふまえて詳細に分析する方が良い結果が得られる。

(2) 人的要因は人間の個性に強く関係すると予想される。人間の個性は心理学的なものあるいは生理学的なものに左右される。したがってこの研究には，ソフトウェア工学の知識の他に心理学的知識や生物学的知識も必要になる。心理学や生物学の長い歴史を考えれば，これらの知識は簡単には得られないので，専門家の協力を得ることが研究を効率的に進める上で重要である。

この研究の目的は，ソフトウェアの信頼性に影響を及ぼす人的要因を明らかにして，それらの要因を制御することによりソフトウェアプロセスの改善をはかることにある。この意味で，将来的には実験により得られた知見を信頼性向上ガイドラインや教育・訓練カリキュラムの作成などに反映する予定である。

最近，ＣＡＳＥが注目されているが，ソフトウェア開発をいかに自動化しても，情報のソースを認識し，最後に知覚するのは人間であり，ソフトウェア開発においては，人間がもっとも根源的な対象であることは否定できない。人的要因の問題は，人間が何を知り，何を考え，どう思うのかという，人間の思考過程の解明にまで及ぶ長期的な研究課題であり，ＣＡＳＥとともにソフトウェア工学の重要な研究分野として，従来にも増してその発展が望まれる。

謝辞

本研究はＩＰＡ技術センターの「ソフトウェア信頼性に及ぼす人的要因の調査研究」プロジェクトのワーキング委員会メンバの協力を得て進めたものであり，関係各位に感謝いたします。

参考文献

(1) B.Curtis, Eds. "Tutorial: Human Factors in Software Development," IEEE Computer Society Press, 1985.

(2) T.Nakajo and H.Kume, "A Case History Analysis of Software Error Case-Effect Relationships," IEEE Trans. on Software Eng. Vol.17, No.8, 1991.

(3) E.D.Young, "Human Errors in Programming," International Journal of Man-Machine Studies, Vol.6, 1974.

(4) G.M.Weinberg and E.L.Schulman," Goals and Performance in Computer Programming", Human Factors, Vol.16, No.1, 1974.

(5) V.R.Basili and R.W.Reiter, Jr, "An Investigation of Human Factors in Software Development," Computer, Vol.12, No.12, 1979.

(6) G.Rzevski, "Identification of Factors which Cause Software Failure," Proc. Annual Reliability and Maintainability Symposium, 1982.

(7) 吉田他，"ソフトウェアの品質に影響を与える人的要因"，第７年度ソフトウェア品質管理研究会報告書，日科技連，1992.

(8) 高橋，古宮他，"人的要因メタモデルの提案とその適用実験例"，第47回情報処理学会全国大会講演論文集，1993.

(9) S.Komiya, M.Takahashi et al. "A Model of Human Error Generation Processes in Software Development and its Evaluation," Proc. Second IASTED International Conference on Reliability, Quality Control and Risk Management, 1993.

(10) T.E.Bell and T.A.Thayer, "Software Requirement : Are They Really Probiem?," Proc. Second ICSE, 1976.

(11) A.Endres, "An Analysis of Errors and Their Causes in System Programs," Proc. ICRS, 1975.

(12) 花田，高橋他，"ソフトウェアの品質向上を指向したバグ分析の一方法"，第１回ＳＰＣシンポジウム報文集，日科技連，1981.

付録［実験に用いた問題］

以下のプログラムに，もしエラーがあれば修正し，その行番号，エラーの内容，エラーとした理由，それに到った思考過程（このように調べていくうちに，あるいは考えているときに，おかしいと思ったなど）を解答用紙に記入して下さい。なお，間違ったときには消しゴムは使わず ——— 線で消して，下の行に修正した内容を記述して下さい。

［プログラムの説明］
ブランクで区切られた値が9999以下の正の整数を5個づつキーボードから読み込み，それらの整数の中から最大値と最小値を求め，それぞれ出力する。9999が入力されたときに処理を終了する。

［プログラム］

```
01   main( )
02    {
03        int  min, max, num[5];
04        int  i;
05        printf("data ?");
06        for(i=0; i<5; i++) ;
07        {
08           scanf("%d", &num[i]);
09           if (num[i]== 9999)  goto end;
10           if (num[i] < 0 &  num[i]>10000)
11              {
12                printf("error data") ;
13                i = i - 1 ;
14                continue ;
15               }
16         }
17        max = 9999;
18        min = 0;
19        for(i=0; i<5; i++);
20        {
21          if (num[i] < max)  max = num[i];
22          if (num[i] > min)  min = num[i];
23         }
24         printf("max= %d          min= %d n" ,max, min);
25  end:   printf("end");
26     }
```