N-021

多数の採点項目によるCプログラミング実技試験の自動採点
Automated Marking of C Programming Examination from Various Viewpoints

石原 俊 † **田口 浩** † **島川 博光** ‡
Shun Ishihara Hiroshi Taguchi Hiromitsu Shimakawa

1. はじめに

現在，ほとんどの大学の情報系学部において，C 言語プログラミング実技試験が実施されている．学生数の多い大学において，プログラミング実技試験の採点は，教員にとって多大な負荷となる．そのような場合，教員は自動採点システム [1] により，採点作業の負荷を軽減する．負荷が軽減される一方で，自動採点システムには評価が細やかでないという問題点がある．教員の負荷軽減に重点を置いた既存の自動採点手法として，GAME [2] がある．GAME が学生に開示する採点結果は，点数やエラーメッセージなどの定量的なもののみである．そのため，学生は採点結果から，自分のプログラムのどこに欠陥があるのかを確認することができない．また，評価の細やかさに重点を置いた手法として，動的解析 [3] がある．しかし，動的解析は，出題可能な問題形式への制限や，教員の作成物の多さなどから，プログラミング実技試験の採点には適さない．

そこで，本論文では，多様な採点項目による自動採点を容易に実現するため，Mark Up Marking 手法を提案する．本手法では，教員が特殊なマークアップ言語を用いて採点項目を設定する．これにより，低負荷で細やかな自動採点が実現される．提案手法を実現するシステムを実装し，性能評価実験を行った結果，時間的に低負荷で採点項目を追加できることが確認された．

2. プログラミング実技試験と自動採点

2.1 プログラミング実技試験

大学で実施されている C 言語プログラミング実技試験の問題の多くは，書式や機能が指定された関数を学生に作らせるような形式である．教員は問題の他に，動作確認用の main 関数を学生に与える．学生は試験中，動作確認用 main 関数を用いて，自分のプログラムの動作を確認しながら解答を作成する．

2.2 自動採点システム

学生数の多い大学において，プログラミング実技試験の採点は，教員にとって多大な負荷となる．負荷軽減のため，教員は採点をシステムにより自動化する．システムによる自動採点の流れを図 1 に示す．

システムは，教員と学生の解答を共通の入力データにより実行し，両者の出力を比較する．比較の結果，両者が一致していれば学生の解答は正解と，一致していなければ不正解と判定される．本論文では，このような流れで, 複数の学生の解答に対する採点を自動で行うようなシステムを，自動採点システムと呼ぶ．

自動採点システムの問題点として，採点時に用いる採点項目, すなわち入力データの少なさゆえに，評価が細やかではないという点が挙げられる．一方で，多数の採点項目による細やかな採点が可能なシステムは，作成や利用の負荷が高く，多くの学生が受験する試験には適さない．問題の解決には，評価の細やかさと，利用に要する教員の負荷のバランスを考慮した手法が必要である．

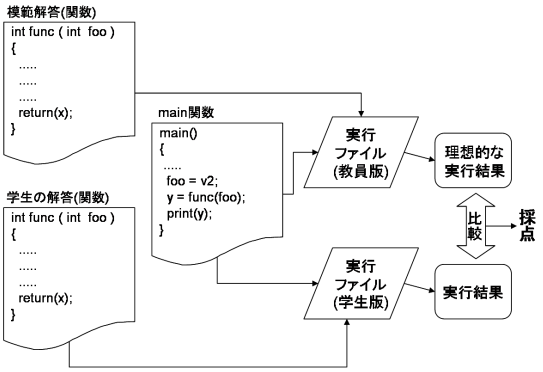

図 1: システムによる自動採点の流れ

3. Mark Up Marking 手法

本論文では，多様な入力による採点を容易に実現するため，Mark Up Marking 手法を提案する．提案手法をシステムとして実装したものを Mark Up Marker と呼ぶ．Mark Up Marker は，動作確認用 main 関数を大量に自動生成することで，複数の入力データを供給する．

3.1 Mark Up Marker

Mark Up Marker の概略図を図 2 に示す．まず，教員と学生の両者は，システムに対してプログラムファイルを入力する．学生が入力するファイルは，自分の作成した解答関数である．教員が入力するファイルは，模範解答関数と，採点用 main 関数という特殊な main 関数で

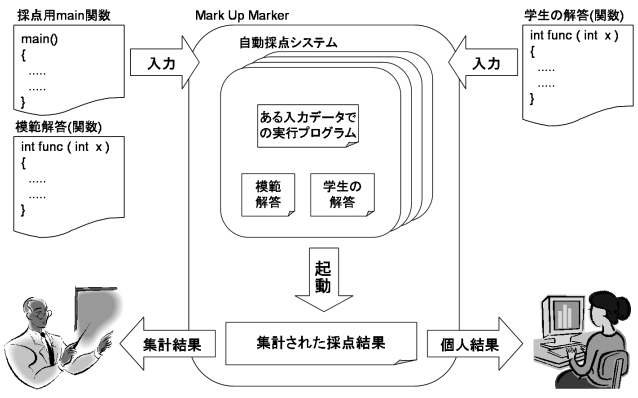

図 2: Mark Up Marker の概略図

†立命館大学大学院理工学研究科
‡立命館大学情報理工学部

ある．採点用 main 関数とは，動作確認用 main 関数に特殊なマークアップ言語を用いて複数の採点項目を追記したものである．採点項目は，学生の解答に適用する入力データ，採点項目に対する配点，採点項目の作成意図などのコメント，の 3 つの情報から構成される．

次に，システムは採点用 main 関数を解析し，採点項目の数だけ，項目内の入力データに対応した main 関数を作成する．さらに，生成されたすべての main 関数と，双方の解答を用い，採点項目の数だけ図 1 と同様の自動採点システムを構成する．

最後に，システムは構成されたすべての自動採点システムを逐次起動し，結果の集計を行う．集計された結果は，学生には個人結果として，教員には全体結果としてそれぞれ通知される．そのさい，採点項目に対する配点や，作成意図などコメントも併せて通知される．これにより，学生は正解か否かだけでなく，どういう点が至らなかったのかを確認することができる．

3.2 マークアップ言語

マークアップ言語は，以下の 3 種類のタグから成る．

args タグ 関数の引数に値を設定している箇所に目印をつける．

replace タグ 目印をつけた箇所に対する差し替えコードを記述する．

case タグ 差し替えコードの組み合わせなどから構成される，採点項目を記述する．

これらのタグは，C 言語コンパイラからはプログラムコメントに見えるように，/*と*/で囲んで記述する．図 3 にタグ同士の関連を示す．まず，教員は args タグを用いて，採点対象関数の引数に値を設定している箇所に目印をつける．次に，replace タグを用いて，目印をつけた箇所に対する差し替えコードを記述する．最後に，case タグを用いて，採点項目を記述する．採点項目は，複数存在する差し替えコードのうち，どれとどれを用いるかという組合せ情報などから成る．システムは，目印のついた箇所を任意の差し替えコードで置き換えることで，多数の main 関数を生成する．

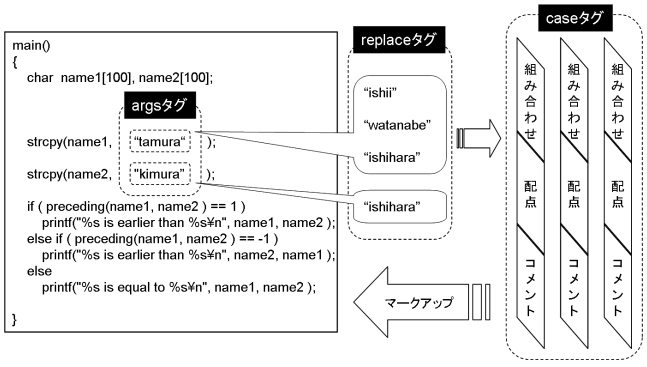

図 3: タグ同士の関連

表 1: 採点結果

	A	B	C	D	E	F	G	H	I	J	K	L
初回	50	50	80	0	0	0	100	100	80	80	50	0
最終	25	50	80	0	0	0	75	75	65	80	65	0

4. 評価

本論文では，提案手法が採点項目をどの程度容易に追加できるかを検証するため，実験を行った．

まず，解答データ収集のため，2.1 で説明したような形式のプログラミング実技試験を，学生 12 人を対象に実施した．次に，被験者は学生 12 人分の解答を，Mark Up Marker を用いて 3 つの採点項目で採点した．採点項目は，教員の裁量に任せて作成させた．最後に，全採点結果のうち，点数に差がついていない学生群に着目し，採点項目の見直しを行った．見直しは，採点用 main 関数にマークアップされた採点項目を追加，変更することにより行われた．学生群の点数に差がついた時点で実験終了とし，それまでに要した時間を計測した．

初回と最終の採点結果を表 1 に示す．初回の結果では，学生 A，B，K が 50 点で，C，I，J が 80 点で，G，H が 100 点でそれぞれ同じ結果となった．一方，最終の結果では，新たに 3 つの採点項目が追加され，それにより学生 A，B，K 間，C，I，J 間において点数が差別化された．初回の採点が終了してから最終の採点が終了するまでに教員が要した時間は 9 分 3 秒であった．このことから，提案手法は時間的に低負荷で採点項目を追加できることが確認された．

5. おわりに

本論文では，C 言語プログラミング実技試験を対象として，多様な採点項目による自動採点を従来より低負荷で実現するため，Mark Up Marking 手法を提案した．また，Mark Up Marking 手法を実現する Mark Up Marker を実装し，性能評価を行った．その結果，提案手法は時間的に低負荷で採点項目を追加できることが確認された．

今後の課題として，受験者の多い実際の試験への適用や，マークアップ言語の記述支援などを考えている．

参考文献

[1] Lauri Malmi, Ari Korhonen, Riku Saikkonen: ” Experiences in Automatic Assessment on Mass Courses and Issues for Designing Virtual Courses ”, Proc. of ITiCSE’02, pp.55-59 (2002)

[2] Michael Blumenstein, Steven Green, Ann Nguyen and Vallipuram Muthukkumarasamy: ”GAME: A Generic Automated Marking Environment for Programming Assessment ”, Proc. of ITCC’04 (2004), pp.212-216

[3] Nghi Truong, Paul Roe, Peter Bancroft: “ Automated Feedback for “ Fill in the Gap ” Programming Exercises ”, Proc. of Australasian Computing Education Conference 2005, pp.117-126 (2005)