N-034

C 言語における関数の戻り値とブロック構造の確認が可能な自動採点ツールの提案

An Automated Evaluation Tool for Checking Function Return Values and Block Structures in C Program

白石 太陽† 水谷 泰治† 西口 敏司† 橋本 渉†
Taiyo Shiraishi Yasuharu Mizutani Satoshi Nishiguchi Wataru Hashimoto

1. はじめに

大学におけるプログラミング演習において多数の学習者のプログラムを目視で採点するには、大きな負担がかかる。そこで、自動採点手法を活用することによりその負担の軽減を図ることができる。そのような自動採点ツールとして、ダミー初期値挿入ツール[1]や自動テスト機能を備えたプログラム提出システム[2]などがある。これらの研究ではコンパイルの可否やテスト入力に対する出力の検査が可能であるが、問題の要件を満たしているかの検査ができない。

しかし自動採点を行う際、関数で行うべき処理が関数内に含まれていなかったり、関数の戻り値の仕様が課題で求められているものと異なっている場合がある。例えば、関数の呼び出し元で出力を調整している場合などが挙げられる。そのようなプログラムが、最終的な出力結果が正しいことにより誤って正解と判定されてしまう場合がある。これにより、学習者が要求された仕様を軽視してしまい、仕様の重要性を伝えることが十分にできない可能性がある。そこで本研究では、C 言語のプログラムを対象に、プログラム内のブロック構造を採点できる手法、および実行時に呼び出された関数の戻り値を採点できる手法を組み込んだ採点ツールを提案する。

2. プログラミング演習の自動採点における問題点

出力結果のみを評価基準とした自動採点では、プログラムが要求した仕様を満たしているのかどうかを十分に評価できない場合がある。例えば、関数を作成するという課題の要件に対して、関数を使用せず main 関数内にすべての処理を記述しても、出力さえ一致すれば正解とされてしまう。このような採点では、学習者が関数の使い方やプログラム構造といった重要な設計上の観点を軽視し、表面的な正解のみを目指してしまう可能性がある。これにより、プログラミング教育の本質である「考え方」の指導が困難となる。

3. ブロック構造と関数の戻り値の採点

この問題を解決するために、C 言語のプログラム内のブロック構造を精査する手法と、プログラム中で呼び出された関数の戻り値の正否を判定する手法を提案する。

3.1 ツールの全体像

ツールの全体像を図 1 に示す。まず、採点対象となるプログラムを本ツールに投入する。そこで構文解析を行い、プログラムの構造を木構造で表した構文木を作成する。その構文木を操作することにより、次節以降で説明するラベル付きプログラムと関数の戻り値を出力するように改変したプログラム（関数ラップソースと呼ぶ）を作成する。その後、関数ラップソースを gcc でコンパイルした実行ファイルから出力ファイルを生成する。その出力ファイルとラベル付きプログラムに採点者が grep などのパターンマッチングを行うことにより採点する。この一連の流れはシェルスクリプトによって行い、そこでプログラムのテスト入力とその結果の採点も行う。コンパイルに失敗する、もしくは出力が一致しなかった場合、そこでそのプログラムの採点を終了する。出力までの採点で正しいと判定したプログラムは、後述するブロック構造表示機能と関数の戻り値を表示する機能による採点を行う。

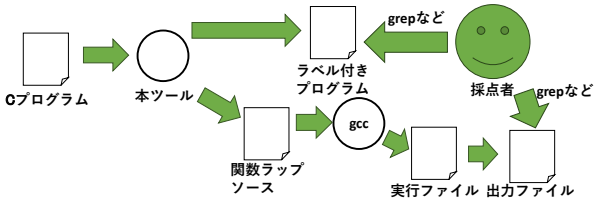

図 1. ツールの全体像

3.2 ブロック構造を表示する機能

図 1 のラベル付きプログラムを生成する機能であり、図 2 のようにブロック構造をプログラムの行頭に追加したファイルを作成する機能である。これに grep などのパターンマッチングを組み合わせることで採点する。例えば、図 2 の例にて「/func_sum:printf」や「/func_sum/for」というパターンをこのファイルから検索することにより、それぞれ「sum 関数内で printf を記述しているか」「sum 関数内で for 文を使用しているか」ということを検査できる。このようなパターンマッチングを採点者が事前に設定しておくことで、ブロック構造を調べることができる。

なお、構文木作成の際にブロック構造を検査することも可能であるが、その場合採点者は構文木の扱い方について熟知する必要がある。しかし、構文木を扱うための学習には大きな負担がかかるため、本ツールでは正規表現や Linux コマンドの知識があれば扱うことができるようにラベル付きプログラムを生成するようにした。

3.3 関数の戻り値を表示する機能

図 1 の関数ラップソースを生成する機能である。関数ラップソースとは図 3 のように、元々の sum 関数を sum_original 関数に名前を変更し、新たに sum 関数を挿入する。その sum 関数内で sum_original 関数を呼び出すことにより、sum 関数の戻り値の出力を行うソースコードとなっている。図 2 のようなプログラムの場合は「sum: 3」と「main: 0」を改行しつつ出力されたファイルを作成する。これにより最終的な出力が正しいが、関数内の処理が仕様と異なっている場合などにその誤りを検出しやすくなる。例えば、平均値を求める関数が仕様として求められている場合、最後の除算を関数の外（main 関数内など）で行って

† 大阪工業大学 Osaka Institute of Technology

```
:int sum(int arr[], int size){
/func_sum: int s = 0;
/func_sum: for(int i=0; i<size; i++){
/func_sum/for: s += arr[i];
/func_sum/for:}
/func_sum: return s;
/func_sum:}
: int main(void){
/func_main: int nums[] = {1,2,3,4,5}, total = sum(nums,5);
/func_main: printf("合計:%d¥n",total);
/func_main: return 0;
/func_main:}
```

図 2. ラベル付きプログラム

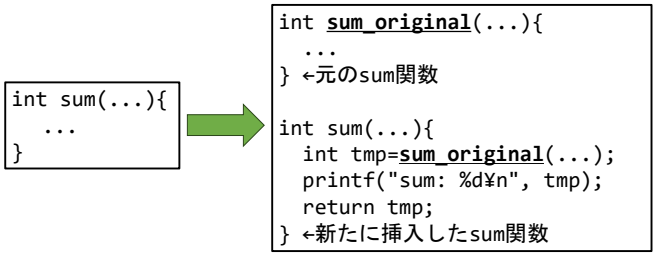

図 3. 関数ラップソースへの変換

も最終的な出力は一致する。しかし、それでは「平均値を求める関数」という仕様から外れたプログラムとなってしまう。それを本機能により容易に誤りと判定することができるようになる。また、関数の戻り値の誤りを main 関数内でつじつま合わせをすることで最終的な出力が正しくなるようにしているプログラムも誤りとして判定しやすくなる。

4. 実験

大阪工業大学情報科学部で行われている「C 演習 I」で提出されたプログラムを提案手法で採点し、(1)コンパイルの可否、(2)出力の正否、(3)ブロック構造の正否、(4)関数の戻り値の正否、それぞれにどれだけ誤りを発見できるかを実験する。実験環境は msys2[3]である。問題の内容としては、入力された 3 つの数値を順に上底、下底、高さとしたときの台形の面積を求める問題となっている。仕様としては台形の面積を求める際に traArea 関数を定義・利用する必要があり、関数に目的に反した処理（printf など）が記述されている場合は減点となる。

本実験では、msys2 環境上でコンパイルの可否とテスト入力に対する出力結果の正否という二項目の採点に加え、ブロック構造表示と関数の戻り値表示による二項目でも採点を行う。その差異により、本ツールを用いることで新しく発見することができた誤りのあるプログラムの数を計測する。総ファイル数は 431 であり、ブロック構造の採点の際の減点要素は、①traArea 関数が存在しないこと、②traArea 関数内で printf を用いていることとする。

採点した 431 個のプログラムのうち、コンパイルに失敗した、もしくは出力が誤っていたプログラムは 22 個見られた。出力まで正しいことが確認できた 409 個のプログラムのうち、ブロック構造上の減点要素を含んでいたプログラムは 27 個あり、同様に関数の戻り値に誤りがあるプログラムは 32 個検出できた。

また、実際に発見できたプログラムの一例を図 4 に示す。図 4 の例ではコンパイルは可能であり、プログラム全体の出力自体は正しい。しかし、traArea 関数の戻り値の型が double ではなく int になっており、さらに traArea 関数内の

```
....
int traArea(int a, int b, int c){
   double menseki;
   menseki = (a + b) * c;
   return menseki;
}
int main(void){
   int a,b,c;
   double e;
   ....
   e = traArea(a,b,c);
   printf("%.2f¥n",e/2);
   return 0;
}
```

図 4. 関数戻り値表示機能で新たに誤りを検出できた一例

計算でも「/2」を行っていない。代わりに、main 関数内で「/2」をすることにより、出力のつじつま合わせを行っていることがわかる。これは台形の面積を求める関数としては不適切であり、このような誤りのあるプログラムを新たに検出できるようになった。

このように出力のみの採点では発見しきれなかった誤りを、本ツールによって検出することができることが確認できた。よって C 言語の基礎的なプログラムに対して本ツールは有用であるといえる。

5. 関連研究

涌井らの研究[4]では、プログラムの構文木の終端ノードのペアとその間の非終端ノードの列として分解したものを示す Path context を使った評価手法を提案している。Path context の類似度検証によって部分点を算出し、プログラムの的確な採点が可能である。しかし、構文木の採点で終止してしまうことにより、学習者にとってなぜその採点結果になったのかというフィードバック性が低いということや、そのプログラムが正確な動作をするかということを確認しづらいことが考えられる。

6. おわりに

本稿では C 言語で記述されたプログラムの自動採点を行う際、プログラムのブロック構造と関数の戻り値の採点を行うツールを提案し、実験を行った。実験の結果、提案ツールによって従来のツールでは発見しきれなかった誤りを精度良く発見できることがわかった。

今後の課題は、グローバル変数のように本来直接参照すべきでない変数が、main 関数などの想定していないブロックで使用されている場合に、誤りを検出することが難しい点である。このようなケースにも対応可能な仕組みの導入が求められる。

謝辞

本研究は JSPS 科研費 JP23K02644 の助成を受けました。

参考文献

[1] 白石太陽,四良丸恵伍,水谷泰治,西口敏司,橋本渉, "C プログラムの未初期化変数へダミー初期値を挿入するツールの提案", FIT2024, N-016, 2024

[2] 望月将行,森田 直樹,北英彦,高瀬 治彦,林照峯, "自動テスト機能を備えたプログラム提出システム", 2003 PC カンファレンス, pp.343-344, 2003.

[3] https://www.msys2.org/

[4] 涌井慧,寺田実,中山泰一, "Path Context を用いたプログラム部分点の算出方式とその評価", 情報教育シンポジウム論文集 2024, pp.118-123, 2024