N-016

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

Inserting Dummy Initialization to Uninitialization Variables in C Program

白石 太陽† 四良丸 恵伍‡ 水谷 泰治† 西口 敏司† 橋本 渉†
Taiyo Shiraishi Keigo Shiramaru Yasuharu Mizutani Satoshi Nishiguchi Wataru Hashimoto

1. はじめに

プログラミング言語の中には、変数を使用する際に宣言が必要となるものが多い。その中でも C 言語では、初期化をせずに宣言した局所変数には不定な値が格納される。しかし、C 言語の開発環境によっては宣言された変数に不定な値ではなく、偶然 0 が格納される場合も多い。例えば、msys2 [1]という環境が該当する。これにより、変数の初期化忘れによって本来正しく動作しないプログラムであっても、特定の環境上では正しく動作した時と同じ出力を偶然してしまう場合がある。msys2 は大学等のプログラミング演習においても用いられることがあり[2]、受講生がコンパイラの警告を無視することが多いことも相まって、前述のような挙動は受講生自身がバグに気付きにくくなるため好ましくない。そこで本研究では、未初期化の変数に対してダミーの初期値を自動的に挿入することにより、初期化忘れによるバグを表面化させやすくするツールを提案する。

2. プログラミング演習における問題点

C 言語では初期化をせずに宣言した局所変数には不定な値が格納されている。例えば、図 1 に示すプログラムでは、変数 sum は初期化がされていないため、合計値を正しく計算できない。しかし、sum に偶然 0 が格納されている場合には正しく動作した時と同じ出力をしてしまう。これはプログラミング演習において、以下の 2 つの問題がある。

① 演習の指導者の立場として、受講生から提出されたプログラムを実行して採点を行う際、偶然正解と同様の出力がされることで間違った採点をしてしまう。

② 演習の受講生の立場として、プログラムに誤りがあることに気が付くことができない。

これらの未初期化変数はコンパイル時に警告が出るが、指導者の立場としては動作に影響しない他の変数に対する警告である場合があるため、警告の有無だけではバグの存在を確信できない。逆に受講生の立場としてはプログラミング初学者はたとえ指導者が注意しても警告を無視しがちであり、偶然正しい出力が出た場合にバグに気が付けない。

3. ダミー初期値の自動挿入ツール

この問題を解決するために、C 言語のプログラムの未初期化変数にダミーの初期値を自動的に挿入するツールを提案する。

3.1 ダミー初期値

ダミー初期値とは、未初期化変数の宣言に対して本ツールが付与する、エラーを誘発するための初期値である。ダミー初期値の挿入を行うことで、正しく動作する時（図 1 の例では sum=55）とは異なる出力がされるため、受講生と指導者のどちらの立場からでもバグを認識しやすくなる。例えば、図 1 の sum の値に 344（ランダムに選んだ値）を代入しておけば、for 文後の sum の値の出力時に 399 という正しい動作の時とは異なる出力がされるため、バグに気づきやすい。

```
int i, sum;
for(i=1; i<=10; i++) sum+=i;
printf(”%d¥n”, sum);
```

図 1. 1 から 10 の和を出力するプログラム

3.2 構文解析の必要性

単純な方法として、grep や sed などを用いた行単位のパターンマッチングを行うことが考えられる。例えば「int x;」のような単一の変数宣言であれば「int x=344;」のように、変数名とセミコロンの並びを文字列置換することでダミー初期値を挿入できるが、「int a[N];」や「struct test a;」のような宣言は、N の値や構造体の定義を参照する必要があり、単純には対応できない。また、C 言語では大域変数の宣言が未初期化である場合には、0 に初期化されることが言語仕様で決まっているため、大域変数にはダミー初期値を挿入してはならない。このように、ダミー初期値を挿入するためには、変数宣言をするうえで必要とする定義の情報やその変数宣言の種類を正確に把握・判別することが重要であるため、構文解析が必要となる。

3.3 初期化コードの挿入手順

全体の動作の流れとしては、まずプログラムに対して構文解析を行い、同時に対象のプログラムの構造を表す構文木を作成する。その後、構文木を探索し、局所変数の宣言を表すノードを見つける。検索によって発見したノードが、初期化をしていない変数宣言であるならば、そのノードの情報からその変数が単一の変数なのか配列なのか、構造体なのかといった変数の型の判定を行う。変数の宣言であるならば「=xxx」の形式の初期化（初期化子）を挿入する。配列や構造体など複雑な形の変数ならば、あらかじめ用意した初期化用の関数に渡す文を構文木に挿入する。例えば、setArrayDummy((int *)array, 5)のような関数呼び出しを構文木に追加する。

3.4 ツールの全体像

ツールの全体像を図 2 に示す。まず、検査したいプログラムをダミー初期値自動挿入ツールに入力し、ダミー初期値を挿入した C プログラムを出力する。この時、C プログラムは 3 つ出力され、それぞれ異なる初期値を変数に挿入している。これは、ダミー初期値を挿入することによって、かえって出力結果が偶然正しくなってしまうという挙動を

† 大阪工業大学 Osaka Institute of Technology
‡ パーソル AVC テクノロジー株式会社 PERSOL AVC TECHNOLOGY CO., LTD.

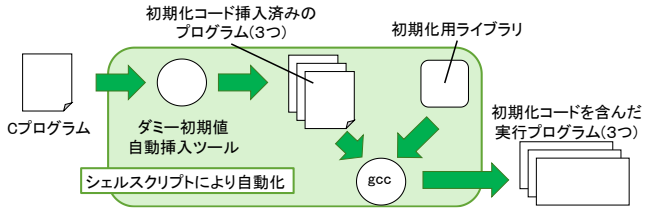

図 2. ツールの全体像

表 1. 挿入される初期値

種別	int	double	float	char	ポインタ	構造体
1	1～999	0.9～999.9		'a'～'z'	NULL	メモリ全てに 97～122
2	0	0		'a'～'z'	NULL	メモリ全てに 0
3	−999～−1	−999.9～−0.9		'a'～'z'	NULL	メモリ全てに 97～122

しても、初期化忘れを発見できるようにするためである。その後、出力された 3 つの C プログラムと、初期化のための関数が定義されているライブラリを C コンパイラに渡して、それぞれの実行プログラムを作成する。これらの一連の処理はシェルスクリプトによって自動化している。

3.5 ダミー初期値の選定方法

表 1 に各変数に挿入する初期値を示す。この表の縦軸は、ダミー初期値自動挿入ツールの実行時に指定する初期値の種別である。出力される 3 つの C プログラムには、それぞれ表 1 中の各行のダミー初期値が挿入される。また、ポインタ型変数以外は、乱数を用いて決定しているため、プログラムの実行ごとに異なる値がダミー初期値として挿入される。ポインタ型変数に NULL を挿入するのは、そのまま初期化をせずにポインタ先を参照した際に、プログラムにセグメンテーションエラーなどの実行時エラーを発生させやすくするためである。ポインタに関しては、誤った結果の出力ではなく、セグメンテーションエラーというプログラムが正しい動作をしないという形で、初期化忘れの表面化を行う。

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」で実施された実技テスト 2 問について、提出されたプログラムに提案手法を適用し、初期化によるバグをどれだけ表面化させることができるのかを実験した。実験環境は msys2 である。まず問 1 は、入力されている数値が正の整数である間その数値を足し続け、最終的に入力された正の整数の平均値を出力するプログラムである。問 2 は、入力された 8 つの正の実数を配列に格納し、それらの値の合計を繰り返し処理によって計算する。その後、その結果を出力するプログラムである。各問は、いずれも入力された値を for 文や while 文を用いて足し合わせていく処理を含むプログラムを記述することが、解答として期待される C 言語の基礎的な問題である。

本実験では、ツールを使わずに msys2 環境上でコンパイルしたプログラムの正誤と、本ツール使用後のプログラムの正誤から、ツールの使用によってどれだけ誤答が増えたかを測定する。この実験におけるツール使用後のプログラムの正誤判定は、3 種類の実行プログラム全てが正しい結果を出力した場合に正解、どれか 1 つでも間違った結果が出力された場合は誤答という条件で行うこととする。

表 2. ツールの実験結果

	問1	問2
総ファイル数	407	401
ツール未使用時に正しい出力をするプログラム	225	336
初期化忘れによるバグがあるプログラム(目視)(①)	14	31
ツールによってバグが発見できたプログラム(②)	14	31
発見率(②/①*100)	100%	100%

その実験結果は表 2 に示す。表 2 は上からテストした総ファイル数、ツール未使用時に正しい出力をするプログラム数、その中で初期化忘れによるバグが目視で確認できたプログラム数（①）、ツールによってバグが発見できたプログラム数（②）、発見率（②/①*100）をそれぞれの問ごとにまとめたものである。

表 2 より、問 1 では、ツールを使用せずに正しい出力をしたものが 225 個あり、そのうちの 14 個が初期化忘れで偶然正しい出力と同じ出力をしたプログラムである。これは問 1 で正しい出力をしたプログラムの内の 6%にあたる。その 14 個は、本ツールで初期化忘れによるバグを全て発見することができた。問 2 も同様に、正しい出力をした 336 個のプログラムのうち、31 個、すなわち問 2 で正しい出力をしたプログラムの内の 9%に初期化忘れによるバグを確認できた。そしてそれらのバグも、本ツールで全て発見することができた。

以上のことより、ツールの使用によってバグを表面化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

Jana らの研究[3]では、初期化されていない変数によって実行時エラーに繋がる可能性のある障害を特定するために、動的解析を用いた未初期化変数の検出ツールを作成している。

6. おわりに

本稿では C 言語で記述されたプログラムに対してダミーの初期値を自動的に挿入して、変数の初期化忘れによるバグを表面化させるツールを提案し、実験を行った。実験の結果、提案ツールによって初期値忘れによるバグを精度良く発見できることがわかった。

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] https://www.msys2.org/
[2] 水谷泰治, 井垣宏, 尾花将輝, 西口敏司, 橋本渉. "大阪工業大学情報科学部の初年次 C プログラミング演習における BYOD のためのプログラミング環境". 情報処理学会第 81 回全国大会, 5J-05, (2019-03).
[3] A. Jana, R. Naik, "Precise Detection of Uninitialized Variables Using Dynamic Analysis - Extending to Aggregate and Vector Types", 19th Working Conference on Reverse Engineering, pp. 197-201, 2012