実引数が整数リテラルの関数呼出しの調査と考察

M2022SE011 横井秀哉
指導教員：蜂巣吉成

1　はじめに

プログラムには特別な意味を持った数値のリテラルが使われることがある．これはマジックナンバーと呼ばれ，数値の意味を理解することが難しい．Sharma ら [2] は，C #のコードにおいて調査を行ったところ，16 行に 1 つはマジックナンバーが含まれているという結果になった．関数呼出しにおいても実引数に整数リテラルが使われていると，関数呼出しを見ただけではどのような振る舞いがされているのか分からない．定義から理解したりコメントなどのドキュメントから理解しなければならず手間である．

本研究では実引数が整数リテラルである関数呼出しに注目し，開発支援を行う開発方法の考察をすることを目的とする．そのために次のリサーチクエスチョンを設定した．対象は C 言語とする．

RQ1. 整数リテラルがどれだけ使われているのか．

RQ2. 整数リテラルの使われ方にパターンはあるか．

実引数に整数リテラルが用いられている関数のソースコードを調査していき，パターン分けを行うために次の 2 つについて調査する．

調査 1 実引数が整数リテラルである関数呼出しについて調査する．

調査 2 実引数が整数リテラルである関数定義について調査する．

調査 1 では，RQ1 である関数呼出しにどれだけ整数リテラルが使われているのか調べる．調査 2 では，RQ2 である整数リテラルの使われ方にパターンがあるのか調査するために関数定義を見る．

以上の調査から整数リテラルの使われ方についてパターン分けを行い，プログラムを編集する人への開発支援を行う方法について考察する．

2　関連研究

Sharma ら [2] は，1998 のリポジトリから 4,900 万行以上の C #コードを調査したところ，実装においてマジックナンバーが平均して 16 行に 1 つあり，最も頻繁に発生するコードスメルの一つであると分析している．

江坂ら [3] は，C 言語では，関数呼出しと関数定義，関数定義と依存関係にある記述が複数のファイルに分散されており，関数の処理を理解するのに時間がかかる場合がある．このように理解する作業などの手間を減らすことでプログラムの利用が効率的に行えると考えた．そこで実引数に NULL が使用されている関数に着目した．NULL の使われ方をパターン定義し，それを元にパターン分類ツールを作成して開発支援ツールを提案した．

3　事例調査

3.1　調査概要

本研究では，ソースコードから実引数が整数リテラルであるものに注目し，どのような使われ方をしているのかを提示する．

調査対象は二つである．一つ目は Apache の httpd-2.4.57[4] を展開してできる第一階層のディレクトリにある C ソースファイルで，必要に応じて apr-1.7.4[5] を使用する．Apache の C ソースファイルは 58 個であり，48,500 行が対象となった．二つ目は postfix-3.7.2[6] を展開してできる第二階層ディレクトリにある C ソースファイルである．postfix の C ソースファイルは 661 個であり，33,640 行が対象となった．以上二つの調査対象から関数呼出しにおける実引数が整数リテラルであるものに注目し，調査を行なっていく．今回は Apache と postfix それぞれ分けて調査を行なっていく．

3.1.1　調査１：関数呼出しの調査

1 つ目の調査として実引数が整数リテラルになっている関数呼出しの調査を行う．調査の一連を説明する．

1. 実引数が整数リテラルとして用いられている関数呼出しを抽出
2. 呼出し個数が Apache httpd-2.4.57[4] では 5 個以上，postfix-3.7.2[6] では 7 個以上であった関数をまとめる
3. 全体の呼び出し個数から引数が整数リテラルである呼び出し個数との割合を調べる

RQ1 の整数リテラルがどれだけ使われているのか調べるために実引数が整数リテラルとして用いられている関数呼出しを抽出する．

次に，抽出した関数呼出しについて，何番目の引数に整数リテラルが用いられているのか，値とともに記録した．抽出した関数呼出しはファイルにまとめた．

3.1.2　調査 2: 関数定義の調査

2 つ目の調査として関数定義について調査を行う．次のように行った．

1. 関数定義を探す
2. 見つかった関数定義を調査
3. パターンを調査

関数定義でパターンが見つからなかった場合は，関数呼出しからパターンを調査する．

3.2　Apache の調査

3.2.1　Apache の関数呼出しの調査

抽出した関数呼出しをカウントしたところ全部で定義された関数は 745 個であった．よって今回調査した範囲では整数リテラルは 745 個使われていた．調査した関数

呼出しで関数毎に呼び出し個数をカウントし，呼出し個数が 5 回以上であった関数をまとめた．今回は定義された関数 26 種類をまとめた．続いて，ここで抽出された関数において第一階層の C ソースファイル全体の呼び出し個数をカウントし，全体の呼び出し個数から引数が整数リテラルである呼び出し個数との割合を調べた．

3.2.2 Apache の関数定義の調査

全体の呼び出し個数から引数が整数リテラルである呼び出し個数との割合が 40 %以上であった関数について関数定義を調べていく．調査対象は定義された関数 22 種類である． Apache httpd-2.4.57[4] を展開してできる第一階層にある C ソースファイルで，必要に応じて apr-1.7.4[5] を使用する．関数定義が見つかった関数が 14 個，C 言語の標準関数であったものが 2 個，今回の調査では見つからなかったものが 6 個であった．見つからなかったものは HTTP_VERSION，YYCASE_などであった．これらは調べると関数ではなくマクロであることが分かった．

関数定義が見つかった関数を調査していく．今回の調査から関数呼出しの実引数において整数リテラルが用いられている際の使われ方について次の通りのパターンが分かった．

1. exit の実引数
2. if の条件式
3. コマンドの終了ステータス
4. 制御結合のフラグ
5. 別の実引数で使われている文字列の長さ

exit の実引数

1 つ目は，exit の実引数として使われているパターンである．次の 2 つの関数は exit の実引数として使用されていた．

- destroy_and_exit_process
- cleanup_tempfile_and_exit

まず関数 destroy_and_exit_process を調べる．実引数が整数リテラルの呼出し個数が 21 個であり，うち第 2 引数が 0 であるものが 5 個，1 であるものが 16 個であった．次に関数定義を Listing1 に示す．実引数が整数リテラルであるのは第 2 引数であったので，int 型の仮引数 process_exit_value に代入されていることが分かる．process_exit_value は exit 関数の引数であり，destroy_and_exit_process の関数定義の第 2 引数は exit の引数であることが分かった．関数 cleanup_tempfile_and_exit も同様であった．

Listing 1 destroy_and_exit_process の関数定義

```
1 static void destroy_and_exit_process
2 (process_rec *process,
3                  int process_exit_value) ···
4 {
5     exit(process_exit_value);
6 }
```

if の条件式

2 つ目は，if の条件式であるパターンである．次の関数がこのパターンであった．

- sock_nopush

実引数が整数リテラルの呼出し個数が 5 個であり，うち第 2 引数が 0 であるものが 2 個，1 であるものが 3 個であった．第 2 引数は，apr_socket_opt_set の第 3 引数に代入されている．関数 apr_socket_opt_set の関数定義のソースコードを Listing2 に示す．apr_socket_opt_set の第 3 引数である変数 on に代入されている．on は if 文の条件式になっているので sock_nopush の第 2 引数は if 文の条件文になっている．

Listing 2 apr_socket_opt_set の関数定義

```
1 APR_DECLARE(apr_status_t)
2   apr_socket_opt_set(apr_socket_t *sock,
3     apr_int32_t opt, apr_int32_t on)
4 { ···
5    if (on)
                  ···
6 }
```

log 関数の引数

3 つ目は，log 関数の引数である．このパターンの関数は 4 種類得られた．

いずれの関数も全て第 3 引数が 0 であった．これはログ出力のきっかけとなったコマンドの終了ステータスを表す．通常 0 は成功である．

制御結合のフラグ

4 つ目は，制御結合のフラグである．このパターンの関数は 1 種類得られた．flags の値によって制御が変わる．flags は処理を制御するために使われている．

文字列の長さ

5 つ目は，別の実引数で使われている文字列の長さである．関数 strncasecmp である．この関数は C 言語の標準関数であり，大文字，小文字の区別を無視して先頭から指定された長さの文字列を比較する関数である．このように第 3 引数を整数リテラルにしてしまうと自力で計算しなければいけないので長くなれば計算ミスが起こる可能性がある．

3.3 Postfix-3.7.2 の調査

3.3.1 Postfix-3.7.2 の関数呼出しの調査

続いて postfix-3.7.2 についても同様に調査を行う．Apache httpd-2.4.57 では呼び出し個数が 5 回以上の関数を抽出していたが，postfix-3.7.2 では呼び出し個数 7 回以上の関数を抽出した．7 個までで十分な数が抽出できたためである．

抽出した関数呼出しをカウントしたところ出現数は全部で 2508 個であった．また，引数に関数リテラルが用いられている関数呼出しとして定義された関数の種類は 339

種類であった．調査した関数呼出しで関数毎に呼び出し個数をカウントし，呼出し個数が 7 回以上であった関数をまとめた．続いて，ここで抽出された関数において第二階層の C ソースファイル全体の呼び出し個数をカウントし，全体の呼び出し個数から引数が整数リテラルである呼び出し個数との割合を調べた．

3.3.2 Postfix-3.7.2 の関数定義の調査

2 つ目の調査として関数定義について調査を行う．

今回は引数が整数リテラルの関数呼出しが 7 回出現した関数を調査した．調査対象としては定義された関数の 47 種類である．関数定義が見つかった関数を調査していく．今回の調査から関数呼出しの実引数において整数リテラルが用いられている際の使われ方について次の 4 通りのパターンが分かった．

1. 文字列の長さ
2. NULL を表す
3. if の条件式
4. 秒数

文字列の長さ

1 つ目は，文字列の長さとして使われているパターンである．2 種類の関数は文字列の長さを表すものとして実引数において整数リテラルが使用されていた．

関数 vstring_alloc を調べる．実引数が整数リテラルの呼出し個数が 853 個であり，第 1 引数の値の内訳は，100 が一番多くて 366 個で割合としてはおよそ 41 %，10 が 292 個で割合としてはおよそ 34 %，1 が 72 個で割合がおよそ 8 %となった．

次に関数定義を見るとコメントより，最初に任意の長さの文字列を指定していることが分かる．以上より，vstring_alloc の実引数の意味は文字列長さを指定する値となっていることが分かる．

関数 argv_alloc も同様であった．

NULL

2 つ目は NULL を表すパターンである．関数の実引数において整数リテラルが NULL を表すために使用されていた．定義された関数は 12 種類であった．

NULL を表す表現方法として以下の 4 つが分かった．

1. (char*)0
2. (void*)0
3. (Vstring*)0
4. 0

1 つ目は (char*)0 である．関数 4 種類がこれに該当した．

argv_add について調査する．実引数が整数リテラルの呼出しの出現個数が 74 個であり，実引数において整数リテラルの値は全て 0 であった．0 の値の内訳は第 3 引数が一番多くて 58 個で割合としてはおよそ 78 %，第 4 引数が 12 個で割合としてはおよそ 16 %，第 5 引数が 3 個で割合がおよそ 4 %となった．

関数呼出しにおいて argv_add(list, item, (char *) 0); や argv_add(＊, ”-t”, transport, (char *) 0); となっている．この (char *) 0 は NULL と書かずに 0 と書いており，NULL と同じ意味を表している．この場合は関数定義より argv_add(ARGV *argvp,...) と仮引数の最後が... となっているので可変長引数の終わりを表している．よって，関数 argv_add は実引数で整数リテラルが使われている際は，NULL と同じ意味で使われていた．

2 つ目は (void*)0 である．関数 5 種類がこれに該当した．

3 つ目は (VSTRING*)0 である．関数 2 種類がこれに該当した．

4 つ目は 0 である．関数 1 種類がこれに該当した．

関数呼出しを見ると cfg_get_ str(parser, ”where field”, NULL, 1, 0)) == 0) と cfg_get_str(＊, ”result_format”, 0, 0, 0)) == 0) となっている．ここから第 3 引数は NULL と 0 となっている．これは，NULL と書く人と 0 と書く人がいることが分かる．よって第 3 引数の 0 は NULL を表していることが分かる．

if の条件式

if の条件式に実引数が整数リテラルになるパターンである．関数 3 種類がこれに該当した．

1 つは cleanup_milter_error である．実引数が整数リテラルの関数呼出しの出現個数は 12 個である．第 2 引数において 0 が 12 個であった．

関数 cleanup_milter_error の関数定義のソースコードを Listing3 に示す．

Listing 3 cleanup_milter_error の関数定義

```
1 static const char *cleanup_milter_error(
      CLEANUP_STATE *state, int err)
2 {
3   if (err) ･･･
4   }
```

関数定義を見ると第 2 引数の仮引数が err である．err は関数中で if の条件式として代入されている．これは真偽を表現する．0 以外の値が真で 0 が偽と表記される．よって，関数 cleanup_milter_error は実引数が整数リテラルの関数呼出しの出現個数は 12 個のうち第 2 引数において全て 0 であったので偽を表す．

秒数

整数リテラルが遅延の秒数を表すパターンである．関数 2 種類がこれに該当した．

event_request_timer を調査する．実引数が整数リテラルの呼出しの出現個数が 45 個であった．第 3 引数の値の内訳は 0 が一番多く 8 個となり割合としてはおよそ 47 %，1，2，3，4 が 2 個で割合としてはおよそ 12 %，10 が 1 個で割合がおよそ 6 %となった．

次に関数定義を見る．関数 event_request_timer の関数定義のソースコードを Listing4 に示す．

Listing 4 event_request_timer の関数定義

```
1 time_t event_request_timer(
      EVENT_NOTIFY_TIME_FN callback, void *
      context, int delay)
2 {  …
3  }
```

関数定義の仮引数を見ると第 3 引数は delay となっており，コールバック関数を呼び出すときの遅延秒数を表すことが分かる．

3.4 調査のまとめ

今回は Apache httpd-2.4.57 と postfix-3.7.2 を調査した．本研究のリサーチクエッションは以下の通りである．

RQ1. 整数リテラルがどれだけ使われているのか．

RQ2. 整数リテラルの使われ方にパターンはあるか．

RQ1 と RQ2 について回答する．RQ1 は Apache httpd-2.4.57 は実引数が整数リテラルになっている関数呼出しをカウントしたところ 745 個あった．第 1 階層にある C ソースファイルにおいても整数リテラルが 700 個以上使用されていることが分かった．
postfix-3.7.2 では実引数が整数リテラルになっている関数呼出しをカウントしたところ定義された関数が 339 種類あり，出現数は 2508 個であった．第 2 階層にある C ソースファイルにおいても整数リテラルが 2500 個以上使用されていることが分かった．
RQ2 として整数リテラルの使われ方を Apache httpd-2.4.57 は 5 種類のパターン，postfix-3.7.2 では 4 種類のパターンにそれぞれ分類することができた．

4 考察

4.1 パターン検出の自動化

if の条件式のパターンは Apache httpd-2.4.57 と postfix-3.7.2 どちらも if(引数) になっている．if の条件式に該当する場合は実引数の 0 を FALSE，1 を TRUE のマクロにできる可能性がある．Apache httpd-2.4.57 の方で見つかった文字列の長さは関数呼出しを見てみると，strncasecmp(＊,”Host:”,5); のように文字列と数字が並んでいる．以上よりこの関数で使われている整数リテラルは文字列の文字数が値であると推測できる．また，仮引数 p に対して，exit(p) になっている．このように仮引数を exit の実引数に使用している場合も自動で検出できる．今回は grep を用いて手動で探していたが構文解析を行なって自動化できれば効率良く探すことができる．

4.2 開発支援

今回は開発支援としてコーディング支援を想定する．プログラムを書く際の関数呼出しの引き数に整数リテラルを用いる場合の支援を考える．この場合，値を決定するために必要な情報は以下の 4 つである．

1. 仮引数名
2. 説明
3. 整数リテラルの個数
4. 整数リテラルの割合

引き数にどのような値を入れればよいか分からないとまずどのような値を入れればいいのか知る必要がある．以上より仮引数名と説明によって引き数が何を表すのか知ることができる．また，プログラムを書く際に適当な値が浮かばないときもある．このときに役立つのが割合と個数である．使用されている割合を見て参考にすることができる．

開発支援ツールの検討

これを基に開発支援ツールを検討する．調査結果から分かった情報をコーディングを行う際の支援に使えるようなツールを検討する．調査結果から分かった情報をデータベースに書き込み，溜め込む．調べたい関数を入力すれば，出力として関数名，何番目の引数なのか，仮引数名，説明，整数リテラルの個数，割合，関数定義が表示されるようにする．また，関数定義も表示することでこれらの情報では不十分だった時に詳しく調べることができるようにする．よって，今回の調査で分かった情報を用いてコーディング支援に役立つことが分かった．

5 おわりに

本研究では，プログラムを編集する人への支援を行う開発方法の考察をすることを目的とした．そのためにオープンソースである Apache httpd-2.4.57 と postfix-3.7.2 の関数呼出しや関数定義の調査を行った．そして，実引数が整数リテラルである関数呼出しに着目し，それぞれのオープンソースごとにパターンを見つけた．見つけたパターンにおいて自動で検出できるか考察も行った．また，調査結果からソースコードを利用する人を支援するために役立つ情報を考え，支援ツールを考察した．今後の課題としては，実際に支援ツールを作成し，有効性を検証することである．

参考文献

[1] Martin Fowler, Refactoring:lmproving the Design of Existing Code, Pearson Education, NJ，1999.(児玉公信, 友野晶夫, 平澤章, 梅澤真史訳, リファクタリングピアソン・エデュケーション, 東京, 2000).

[2] Tushar Sharma, Marios Fragkoulis, and Diomidis Spinellis: House of Cards: Code Smells in Open-source C # Repositories, ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, pp.424-429, 2017.

[3] 江坂亜純, 原野礼衣:C 言語における実引数が NULL の関数呼出のパターンに関する調査と考察, 南山大学理工学部卒業研究要旨集,2021.

[4] Apache httpd-2.4.57，https://httpd.apache.org/

[5] apr-1.7.4，https://apr.apache.org/download.cgi

[6] postfix-3.7.2：https://www.postfix.org