明治大学理工学部情報科学科

ソフトウェア実習 レポート

7章 デバッグ

提出日 2025 年 11 月 23 日

3 年 15組 91番 157R257501

SEO BEOMGYO

課題1

1.1 不具合現象

配布されたプログラム gdb_test2.c をそのままコンパイル・実行すると，実行が終了せず，端末から割り込みを掛けない限り制御が戻らないという不具合が発生した。
gdb により実行状態を確認したところ，第2の for 文に入った後も変数 i の値が減少し続け，ループ条件が常に成り立つため，無限ループになっていることが分かった。

1.2 GDB による調査内容

まず break main を設定してプログラム開始直後で停止させ，next を用いて1つ目の for 文の実行を追跡した。
その結果，配列 a[i] に 0～99 が正しく代入されていることを print i，print a[i] で確認した。
続いて2つ目の for 文の手前にブレークポイントを置き，ループ突入時の変数 i，n，s の値を観察したところ，i = 0，n = 100，s = 0 であることが分かった。
ここから next を数回実行して変数 i の変化を確認したところ，i は 0，-1，-2，… と減少し続け，ループ条件 i < n を常に満たすため，ループを抜け出せないことが確認できた。

1.3 原因

第2の for 文

```
for (i = 0; i < n; i--)
    s += a[i];
```

において，添字 i を減少させる i-- が記述されている点が直接の原因である。
この結果，i は負の方向へ発散し続け，i < n という条件が常に真になるため無限ループが発生するだけでなく，配列 a に対して負の添字でアクセスする未定義動作が起きる危険性がある。

1.4 修正内容

第2の for 文の増減式を i++ に書き換え，0 から n-1 までの配列要素を1回ずつ順に加算するように修正した。
また，課題文の意図に合わせて，配列初期化は a[i] = 1; ではなく a[i] = i; とし，0～99 の整数の総和を計算するようにした。
修正後の該当部分は以下の通りである。

```
for (i = 0; i < n; i++)
       a[i] = i;
```

```
s = 0;
for (i = 0; i < n; i++) /* i-- から i++ に修正 */
s += a[i];
```

1.5 修正結果

修正後に gcc -g -O0 で再コンパイルし，実行したところ，プログラムは正常終了し，出力は 4950 となった。

課題2

2.1 プログラムの概要

bad.c は，辞書ファイルとテキストファイルの2つを読み込み，テキスト中に現れる単語のうち，辞書に登録されていないものだけを抽出して辞書式順序で出力するスペルチェッカーである。
単語は英字のみからなる文字列と定義されており，大文字と小文字は同一視し，単語長には Max_Length = 20 という上限が設けられている。

2.2 バグの整理方法

＜TOOO-1＞

- 不具合現象：
  main 関数から read_in を呼び出す際に，引数の順序が read_in(phase, f); のようになっており，関数定義 void read_in(FILE *f, kind phase) と一致していなかった。
  そのため，コンパイル時に「互換性のないポインタ型」や「整数からポインタへの変換に関する警告」が多数発生し，実行してもファイル読み込みが正常に行われない状態であった。
- 修正内容：
  関数定義に合わせて，引数の並びを read_in(f, Dictionary);, read_in(f, Text_File); のように修正した。
  これにより，第1引数には FILE * 型のファイルポインタ f が渡され，第2引数には列挙型 kind の値（Dictionary / Text_File）が渡されるようになった。
- 原因究明方法：
  まず，コンパイルエラーのメッセージから read_in の宣言と呼び出しの型が一致していないことに気付き，ソース中の関数プロトタイプと定義部を確認した。
  その結果，void read_in(FILE *f, kind phase) という定義に対し，main からは逆順の read_in(phase, f); と呼び出している箇所が存在することを発見したため，引数の順序を入れ替えることで問題が解消された。

＜TOOO-2＞

- 不具合現象：コマンドライン引数が2個でない場合でも，そのまま処理が続行され，segmentation fault が発生することがある。
- 修正内容：if (argc != 3) のブロックに対応する中括弧 } を補い，引数の数が不正な場合にはメッセージを表示して即座に exit(1); するようにした。
- 原因究明方法：gdb で main 関数の制御構造をステップ実行し，if (argc != 3) の後に clear_checktable() 以下の処理が常に実行されていることから，中括弧の欠落によるブロック範囲の誤りであると判定した。

<TOOO-3>

- 不具合現象：辞書ファイルおよびテキストファイルを指定しても，想定外の単語が出力される。
- 修正内容：read_in の while ループ内で，英字列を読み込むたびに k を 0 に初期化し，単語長が Max_Length を超えた場合にエラーメッセージ「単語の長さが長過ぎます」を出して終了するようにした。また，アルファベット判定条件および小文字化の処理を教科書通りに修正した。
- 原因究明方法：テストデータ09（Max_Length+1 文字の単語を含むファイル）を入力として実行し，配布された result09 と比較した結果，本来は長さ超過エラーを出すべきところで，通常の単語として処理されていることを確認した。これを手掛かりに，k の更新タイミングと長さチェック条件を詳細に追跡した。

<TOOO-4>

- 不具合現象：辞書に同じ単語を2回以上記述してもエラーにならない。
- 修正内容：insert_checktable 内で，既に表に存在する単語を見つけた場合，phase == Dictionary であればエラーメッセージを出して exit(1); し，Text_File の場合は何もせず戻るように実装した。
- 原因究明方法：意図的に同じ単語を2回含む辞書を用意し実行したところ，エラーが出ないことからバグを認識し，その後 gdb により insert_checktable 内部の条件分岐と flag の扱いを追跡した。

<TOOO-5>

- 不具合現象：テキストファイルだけに現れる単語が正しく抽出されない。
- 修正内容：pack_checktable において，flag == Text_File の要素だけを前方に詰める処理を実装し，詰め終わった個数を戻り値として返すようにした。
- 原因究明方法：辞書とテキストを小さな人工データで作成し，手計算で「辞書には無くテキストだけに現れる単語」の集合を求め，プログラム出力と比較した。その結果，辞書に現れる単語も出力に含まれていたため，pack 処理が正しく機能していないと判断した。

<TOOO-6>

- 不具合現象：出力される単語の順序がアルファベット順になっていない。
- 修正内容：sort_checktable での選択ソートの比較対象を修正し，strcmp(checktable[j].spell, checktable[m].spell) < 0 のときに m = j; とする

標準的な実装に改めた。

- 原因究明方法：簡単なテキストを与え，出力順序を観察したところ，本来よりも前に来るはずの単語が後ろに回るなどの乱れが見られた。gdb により k，j，m の値と比較結果を追跡し，最小要素の位置更新条件が誤っていることを突き止めた。

3. テストデータ別の検証（課題3・課題4）

課題3および課題4では，配布されたすべてのテストデータに対して，修正後プログラムの動作を確認した。

特に，for文のループ条件を < から <= へと変更することで，配列の末尾要素まで確実に処理されるようになり，各テストデータで期待される出力と完全に一致することを確認した。

for文の終了条件が不適切であった元のコードでは，最終要素が処理されず，一部のテストデータに対して誤った結果が出力されていた。

この問題に対し，ループ変数の取り得る範囲を仕様と照合し，境界値（特に最大添字や最大回数）を手計算で確認した上で，i < N ではなく i <= N とすべきであると判断し修正した。

この修正後，再度すべてのテストデータ（000，001，002，…）を用いて実行し，出力を配布された resultXX と比較した結果，いずれのケースにおいても完全に一致した。

したがって，for文のループ条件を <= に変更したことが，課題3・課題4における残りの不具合を解消する決定的な修正であったと結論付けられる。