ソフトウェア演習Ｖ 課題2(再々提出)

15122013 尾持涼介

提出日：2018年2月9日

1　作成したプログラムの設計情報

1.1　全体構成

各ファイルで記述した関数は以下の通りである。

- main.c
  - main 関数
  - void error(char *mes)
- scan.c
  - int keyword_search(char *string)
  - int init_scan(char *filename)
  - int keyword_search(char *string)
  - int scan()
  - int get_linenum()
  - void end_scan()
- prettyprinter.c
  - int parse_program()
  - int block()
  - int var_decl()
  - int var_names()
  - int type()
  - int ar_type()
  - int sub_decl()
  - int form_para()
  - int fukugou()
  - int statement()
  - int bunki()
  - int kurikaeshi()
  - int call_st()
  - int exp_narabi()
  - int dainyu()
  - int var()
  - int shiki()
  - int simple()
  - int kou()
  - int inshi()
  - int input_st()
  - int output_st()
  - int shitei()
  - void indent()

次に、各関数の呼び出し関係、データ参照関係について述べる。

- main.c
  - main 関数内
    - init_scan() 関数を呼び出し
    - scan() 関数を呼び出し
    - parse_program() 関数を呼び出し
    - end_scan() 関数を呼び出し
  - error 関数内
    - get_linenum() 関数を参照
- scan.c
  - scan() 関数内
    - keyword_search() 関数を呼び出し
- prettyprinter.c
  - parse_program() 関数内
    - scan() 関数を呼び出し
    - block() 関数を呼び出し
  - block() 関数内
    - var_decl() 関数を呼び出し
    - sub_decl() 関数を呼び出し
    - fukugou() 関数を呼び出し
  - var_decl() 関数内
    - indent() 関数を呼び出し
    - scan() 関数を呼び出し
    - var_names() 関数を呼び出し
    - type() 関数を呼び出し
  - var_names() 関数内
    - scan() 関数を呼び出し
    - ar_types() 関数を呼び出し
  - ar_types() 関数内
    - scan() 関数を呼び出し
  - sub_decl() 関数内
    - indent() 関数を呼び出し
    - scan() 関数を呼び出し
    - form_para() 関数を呼び出し
    - var_decl() 関数を呼び出し
    - fukugou() 関数を呼び出し
  - form_para() 関数内
    - scan() 関数を呼び出し
    - var_names() 関数を呼び出し
    - type() 関数を呼び出し

- fukugou() 関数内
  - scan() 関数を呼び出し
  - statement() 関数を呼び出し
  - indent() 関数を呼び出し
- statemen() 関数内
  - indent() 関数を呼び出し
  - dainyu() 関数を呼び出し
  - bunki() 関数を呼び出し
  - kurikaeshi() 関数を呼び出し
  - scan() 関数を呼び出し
  - call_st() 関数を呼び出し
  - input_st() 関数を呼び出し
  - output_st() 関数を呼び出し
  - fukugou() 関数を呼び出し
- bunki() 関数内
  - scan() 関数を呼び出し
  - shiki() 関数を呼び出し
  - statement() 関数を呼び出し
  - indent() 関数を呼び出し
- kurikaeshi() 関数内
  - scan() 関数を呼び出し
  - shiki() 関数を呼び出し
  - statement() 関数を呼び出し
- call_st() 関数内
  - scan() 関数を呼び出し
  - exp_narabi() 関数を呼び出し
- dainyu() 関数内
  - var() 関数を呼び出し
  - scan() 関数を呼び出し
  - shiki() 関数を呼び出し
- var() 関数内
  - scan() 関数を呼び出し
  - shiki() 関数を呼び出し
- shiki() 関数内
  - simple() 関数を呼び出し
  - scan() 関数を呼び出し
- simple() 関数内
  - kou() 関数を呼び出し
  - scan() 関数を呼び出し
- kou() 関数内
  - inshi() 関数を呼び出し
  - scan() 関数を呼び出し

- inshi() 関数内
  - var() 関数を呼び出し
  - scan() 関数を呼び出し
  - shiki() 関数を呼び出し
  - inshi() 関数を呼び出し
- input_st() 関数内
  - scan() 関数を呼び出し
  - var() 関数を呼び出し
- output_st() 関数内
  - scan() 関数を呼び出し
  - shitei() 関数を呼び出し
- shitei() 関数内
  - scan() 関数を呼び出し
  - shiki() 関数を呼び出し

1.2 各モジュールごとの構成

prettyprinter.c では、読み込んだトークンを出力するために次の図 1 のような配列を用意した。

```
char *token_str[NUMOFTOKEN+1] = {
"", "",
"program", "var", "array", "of", "begin", "end", "if", "then",
"else", "procedure", "return", "call", "while", "do", "not", "or",
"div", "and", "char", "integer", "boolean", "readln", "writeln", "true",
"false", "", "", "+", "-", "*", "=", "<>", "<", "<=", ">",
">=", "(", ")", "[", "]", ":=", ".", ",", ":", ";", "read","write", "break"
};
```

図 1: 読み込んだトークンを出力するための配列

なお、読み込んだトークンが「名前」、「符号なし整数」、「文字列」のときはその実体を出力しなければならないためこの配列は用いない。そのためそれらのトークンコード 1、27、28 にあたる要素には何も入れられていない。

また、与えられたマクロ構文に沿って prettyprinter.c 内で関数を定義して if 文や while 文、switch 文で判別し、判別したトークンを適宜出力している。なお、トークンの読み込みにはそれぞれの関数内で scan() 関数を呼び出すことにより行っている。また、読み込んだトークンの出力はそのトークンが「名前」、「文字列」の時には string_attr を、「符号なし整数」のときは num_attr を、それ以外のときは上記で述べた token_str を参照している。

次に使用した大域変数とその意味について述べる。

- main.c 内
  - int numtoken[NUMOFTOKEN+1] : 各トークンを数え上げるために使用。各トークンのトークンコードと同じ場所にその個数を格納する。すなわち、例えば、numtoken[1] には名前の個数が格納される。なお、NUMOFTOKEN とは定義したトークンの個数であり、49 と定義されている。
  - char *tokenstr[NUMOFTOKEN+1] ：各トークン名が格納されている。
  - int token：読み込んだトークンのトークン番号が格納されている。
- scan.c 内
  - int cbuf：ファイルから読み込んだ文字を格納する 1 文字分の文字バッファ
  - int line_cnt：行数を数え上げる変数

- int newline：次の文字から新しい行が始まる場合は 0、そうでない場合は 1 を格納しておく。
- FILE *fp：ファイルを操作するためのポインタ
- char string_attr[MAXSTRSIZE]：読み込んだトークンが「文字列」か「名前」のときにその実体が格納される。なお、MAXSTRSIZE は定義された文字列の最大長であり、1024 である。
- int num_attr：読み込んだトークンが「符号なし整数」の時にその実態が格納される。

- prettyprinter.c 内
  - int indentnum：段付け (インデント) をする回数を格納する。
  - char *token_str[NUMOFTOKEN+1]：字句を格納する配列。

次に各関数内で定義した変数とその意味について述べる

- scan.c
  - ikeyword_search() 関数内：課題 1 で記載済み
  - scan() 関数内
    - int i：課題 1 で記載済み

- prettyprinter.c
  - bunki() 関数内
    - int flag：分岐文内に複合文があるかどうかを示す変数。(1 ならある、0 ならない)
  - kurikaeshi() 関数内
    - int flag：繰り返し文内に複合文があるかどうかを示す変数。(1 ならある、0 ならない)
  - indent() 関数内
    - int i：for 文によるループで用いる変数。

1.3 各関数の外部 (入出力) 仕様

ここでは、各関数の機能、引数と返り値等について説明する。

1.3.1 main.c 内で記述されている関数

- main 関数

  引数 コマンドライン引数として int nc と char *np[] を指定する。nc は指定された引数の個数を表し、np はプログラムを起動するときに指定する引数であり、本プログラムでは読み込むファイル名を指定する。

  返り値 プログラムが終了した際に 0 を返す。

  参照・変更する大域変数 なし

- void error(char *mes)

  機能 エラーが発生したときに、その部分までのプリティプリントをして、エラーメッセージとエラー発生箇所を表示する。

  引数 char *mes：表示するエラーメッセージ

  返り値 なし

  参照・変更する大域変数 なし

1.3.2 scan.c 内で記述されている関数

- int init_scan(char *filename)：課題 1 で記載済み
- int keyword_search(char *string)：課題 1 で記載済み
- int scan()

  機能 トークンを 1 つスキャンし、そのトークンを識別する。

  引数 なし

  返り値 トークンのコードを返す。End-of-File が現れたときやエラーが発生したときは-1 を返す。

  参照する大域変数 cbuf、newline、fp

  変更する大域変数 cbuf、newline、line_cnt、num_attr、string_attr
- int get_linenum()：課題 1 で記載済み
- void end_scan()：課題 1 で記載済み

1.3.3 prettyprinter.c 内で記述した関数

なお、以下で NORMAL と ERROR はそれぞれ prettyprinter.c 内で 0、1 と定義されている。

- int parse_program()

  機能 プログラムを解析する関数

  引数 なし

  返り値 エラーがあれば ERROR を、なければ NORMAL を返す。

  変更する大域変数 token

  参照する大域変数 token_str、string_attr、token
- int block()

  機能 ブロックを解析する関数

  引数 なし

  返り値 エラーがあれば ERROR を、なければ NORMAL を返す。

  変更する大域変数 indentnum

  参照する大域変数 token
- int var_decl

  機能 変数宣言部を解析する関数

  引数 なし

  返り値 エラーがあれば ERROR を、なければ NORMAL を返す。

  変更する大域変数 token、indentnum

  参照する大域変数 token_str、token
- int var_names()

  機能 変数名の並びを解析する関数

  引数 なし

  返り値 エラーがあれば ERROR を、なければ NORMAL を返す。

  変更する大域変数 token

参照する大域変数 token_str、string_attr、token

- int type()

  機能 型を解析する関数

  引数 なし

  返り値 エラーがあれば ERROR を、なければ NORMAL を返す。

  変更する大域変数 token

  参照する大域変数 token_str、token

- int ar_type()

  機能 配列型を解析する関数

  引数 なし

  返り値 エラーがあれば ERROR を、なければ NORMAL を返す。

  変更する大域変数 token

  参照する大域変数 token_str、token、string_attr

- int sub_decl()

  機能 副プログラム宣言を解析する関数

  引数 なし

  返り値 エラーがあれば ERROR を、なければ NORMAL を返す。

  変更する大域変数 token、indentnum

  参照する大域変数 token_str、token、string_attr

- int form_para()

  機能 仮引数部を解析する関数

  引数 なし

  返り値 エラーがあれば ERROR を、なければ NORMAL を返す。

  変更する大域変数 token

  参照する大域変数 token_str、token

- int fukugou()

  機能 複合文を解析する関数

  引数 なし

  返り値 エラーがあれば ERROR を、なければ NORMAL を返す。

  変更する大域変数 token、indentnum

  参照する大域変数 token_str、token

- int statement()

  機能 文を解析する関数

  引数 なし

  返り値 エラーがあれば ERROR を、なければ NORMAL を返す。

  変更する大域変数 token

  参照する大域変数 token_str、token

- int bunki()

  機能 分岐文を解析する関数

  引数 なし

  返り値 エラーがあれば ERROR を、なければ NORMAL を返す。

  変更する大域変数 token、indentnum

  参照する大域変数 token_str、token、string_attr

- int kurikaeshi()

  機能 繰り返し文を解析する関数

  引数 なし

  返り値 エラーがあれば ERROR を、なければ NORMAL を返す。

  変更する大域変数 token、indentnum

  参照する大域変数 token_str、token、string_attr

- int call_st()

  機能 手続き呼び出し文を解析する関数

  引数 なし

  返り値 エラーがあれば ERROR を、なければ NORMAL を返す。

  変更する大域変数 token

  参照する大域変数 token_str、token、string_attr

- int exp_narabi()

  機能 式の並びを解析する関数

  引数 なし

  返り値 エラーがあれば ERROR を、なければ NORMAL を返す。

  変更する大域変数 token

  参照する大域変数 token_str、token

- int dainyu()

  機能 代入文を解析する関数

  引数 なし

  返り値 エラーがあれば ERROR を、なければ NORMAL を返す。

  変更する大域変数 token

  参照する大域変数 token_str、token

- int var()

  機能 変数を解析する関数

  引数 なし

  返り値 エラーがあれば ERROR を、なければ NORMAL を返す。

  変更する大域変数 token

  参照する大域変数 token_str、token、string_attr

- int shiki()

機能 式を解析する関数

引数 なし

返り値 エラーがあれば ERROR を、なければ NORMAL を返す。

変更する大域変数 token

参照する大域変数 token_str、token

- int simple()

  機能 単純式を解析する関数

  引数 なし

  返り値 エラーがあれば ERROR を、なければ NORMAL を返す。

  変更する大域変数 token

  参照する大域変数 token_str、token

- int kou()

  機能 項を解析する関数

  引数 なし

  返り値 エラーがあれば ERROR を、なければ NORMAL を返す。

  変更する大域変数 token

  参照する大域変数 token_str、token

- int inshi()

  機能 因子を解析する関数

  引数 なし

  返り値 エラーがあれば ERROR を、なければ NORMAL を返す。

  変更する大域変数 token

  参照する大域変数 token_str、token

- int input_st()

  機能 入力文を解析する関数

  引数 なし

  返り値 エラーがあれば ERROR を、なければ NORMAL を返す。

  変更する大域変数 token

  参照する大域変数 token_str、token

- int output_st()

  機能 出力文を解析する関数

  引数 なし

  返り値 エラーがあれば ERROR を、なければ NORMAL を返す。

  変更する大域変数 token

  参照する大域変数 token_str、token

- int shitei()

  機能 出力指定を解析する関数

引数 なし

返り値 エラーがあれば ERROR を、なければ NORMAL を返す。

変更する大域変数 token

参照する大域変数 token_str、token、string_attr、num_attr

- void indent()

  機能 段付け (インデント) を付ける。

  引数・返り値 なし

  参照する大域変数 indentnum

2 テスト情報

2.1 テストデータ・テスト結果

私はまず、ブラックボックステストとして配布されたテストデータである、sample014.mpl、sample021.mpl、sample022.mpl、sample023.mpl、sample024.mpl、sample025.mpl、sample026.mpl、、sample02a.mpl、sample21.mpl、sample22.mpl、sample23.mpl、sample24.mpl、sample25.mpl、sample25t.mpl、sample26.mpl、sample27.mpl、sample28p.mpl、sample29p.mpl、sample2a.mpl についてテストを行った。さらに、ホワイトボックステストとして mysample021.mpl、mysample022.mpl、mysample023.mpl、mysample024.mpl、mysample025.mpl、mysample026.mpl、mysample027.mpl、mysample028.mpl、mysample029.mpl、mysample0210.mpl、mysample0211.mpl、mysample0212.mpl、mysample0213.mpl、mysample0214.mpl、mysample0215.mpl、mysample0216.mpl、mysample0217.mpl、mysample0218.mpl、mysample0219.mpl、mysample0220.mpl、mysample0221.mpl、mysample0222.mpl、mysample0223.mpl、mysample0224.mpl、mysample0225.mpl、mysample0226.mpl、mysample0227.mpl、mysample0228.mpl、mysample0229.mpl、mysample0230.mpl、mysample0231.mpl というテストデータを用意してテストを行った。

配布されたテストデータによるテスト結果と、自作のテストデータとその結果についてはメールにより提出する。なお、テスト結果を格納するファイル名は「(テストプログラム名)_test.txt」としている (テストプログラム名の.mpl は省略)。それらをまとめて「kadai2-test.zip」というファイルにまとめて圧縮して提出する。テスト結果を格納しているファイルには想定される出力結果と、実際に行ったテスト結果が書き込まれており、「想定」以下が想定される出力結果で、「結果」以下が実際の出力結果である。テスト情報を添付したメールの送信日時は 2 月 9 日 16 時 34 分である。

2.2 テストデータの十分性

sample014.mpl、sample021.mpl、sample022.mpl、sample023.mpl、sample024.mpl、sample025.mpl 以外の配布されたサンプルプログラムで、エラーメッセージが表示されない場合に通るすべての命令が網羅されている。

残りのテストデータにおいて通常実行されない、コマンドラインが与えられていないときのエラーメッセージ、ファイルが開けないときのエラーメッセージが表示される場合を除く、エラーメッセージを表示するすべての場合に通る命令を網羅している。なお、ここでは prettyprinter.c 内において scan() 関数からの返却値が-1 であった場合に ERROR を返すという処理については、sampe014.mpl によって一か所正常に実行されることが確認できたので、全ての処理において正常に実行されると判断し、命令が網羅されているとした。

すなわち、C0 カバレッジで 100%であると言える。

3 本課題を行うための事前計画 (スケジュール) と実際の進捗状況

3.1 事前計画 (スケジュール)

事前計画は以下の表 1 のように立てた。

表 1: 課題 2 における事前計画

開始予定日	終了予定日	見積もり時間	作業内容
10月30日	10月30日	1	スケジュールを立てる
10月31日	11月2日	3	配布資料・サンプルプログラムを熟読する
11月3日	11月3日	2	コンパイラのテキストを熟読する
11月4日	11月6日	5	構文解析系(プリティプリンタ)の概略設計
11月7日	11月15日	9	プリティプリンタの作成
11月16日	11月16日	1	バグがない場合の想定テスト結果の準備(配布されたテストプログラムについて)
11月17日	11月17日	2	ホワイトボックステスト用プログラムの作成
11月18日	11月18日	1	バグがない場合の想定テスト結果の準備(自分で作成したテストプログラムについて)
11月19日	11月24日	8	テストとデバッグを行う
11月25日	11月25日	1	作成したプログラムの設計情報を書く
11月25日	11月26日	1	テスト情報を書く
11月26日	11月26日	1	事前計画と実際の進捗状況を書く
11月26日	11月26日		プログラムとレポートの提出

しかし、演習中に計画を以下の表 2 のように修正した。

表 2: 課題 2 における修正後のスケジュール

開始予定日	終了予定日	見積もり時間	作業内容
10月30日	10月30日	1	スケジュールを立てる
10月31日	11月2日	3	配布資料・サンプルプログラムを熟読する
11月3日	11月3日	2	コンパイラのテキストを熟読する
11月4日	11月6日	5	構文解析系(プリティプリンタ)の概略設計
11月7日	11月15日	9	プリティプリンタの作成
11月16日	11月16日	1	バグがない場合の想定テスト結果の準備(配布されたテストプログラムについて)
11月17日	11月17日	4	ホワイトボックステスト用プログラムの作成
11月18日	11月18日	4	バグがない場合の想定テスト結果の準備(自分で作成したテストプログラムについて)
11月19日	11月25日	10	テストとデバッグを行う
11月26日	11月26日	1	作成したプログラムの設計情報を書く
11月26日	11月26日	1	テスト情報を書く
11月26日	11月26日	1	事前計画と実際の進捗状況を書く
11月27日	11月27日		プログラムとレポートの提出
12月25日	12月25日		再提出

3.2 事前計画の立て方についての前課題からの改善点

プリティプリンタのコーディングの時間を長めに確保した。

3.3 実際の進捗状況

表 2 のように変更したように、テストとデバッグ、特にホワイトボックステストが予想以上に時間がかかってしまった。その他の作業についてはだいたい計画通りに進んだ。ただ、プログラムの最後の「.」を出力していなかったため再提出を行った。。

3.4 当初の事前計画と実際の進捗との差の原因

命令を網羅するために必要なテストプログラムの数の見積もりが少なかったことが原因であると考えられる。また、再提出となったのは、見直しが不十分であったからであると考えられる。