ソフトウェア演習5　レポート3

提出日　2015/12/21

13122029　高瀬正典

(1)作成したプログラムの設計情報

(1-1)全体構成

ファイルの初期化、リストの初期化を行った後に、関数 prase()を実行する。成功した場合、リストに保存された内容を関数 PrintIdList(Id_c *id)で出力する。

関数 prase()の中で変数や関数の定義毎に、関数 AddIdName(char *name,int ispara, int top_next, Id_c **id)や AddIdType(int token, Id_c **id)を実行し、リストに変数や関数の情報を入れる。

(1-2)各モジュールごとの構成

```
typedef struct ID {
    char *name;             変数、関数名を保存する。
    char *procname;         ローカル変数であれば呼び出し関数名を保存する。
    struct TYPE *itp;       TYPE へのポインタ。
    int ispara;             仮引数か否か。今回は使用していない。
    int deflinenum;         定義された行番号。
    struct LINE *irefp;     LINE へのポインタ。
    struct ID *nextp;       次の ID へのポインタ。
} Id_c;

typedef struct TYPE {
    int ttype;              型の種類を保存する。
    int arraysize;          array 型であればサイズを保存する。
    struct TYPE *etp;       今回は使用していない。
    struct TYPE *paratp;    関数型であれば引数の型を保存するための次のポインタ。
} Type_c;

typedef struct LINE {
    int reflinenum;         使用された行番号を保存する。
    struct LINE *nextlinep; 次のポインタ。
} Line_c;

大域変数    Head_Name_Num:  定義時に変数がいくつも連なる場合の先頭の番号。
大域変数    Last_Name_Num:  定義時に変数がいくつも連なる場合の末端の番号。
extern      PlocFlag:       現在関数に所属しているのかどうかの判別用変数。
```

(1-3)各関数の外部仕様

関数 init_cross(): globalidroot を初期化する。

関数 c_error(char *mes): 関数 RemoveGrobalId を呼び出し、関数 end_scan を呼び出す。エラーメッセージを表示し、プログラムを終了する。

関数 RemoveGlobalId(): globalidroot を開放する。

関数 PrintIdList(Id_c *id): ポインタ id が null になるまで関数 PrintId を呼び出す。

関数 PrintId:(Id_c *id) 関数 PrintIdName、関数 PrintIdType、関数 PrintIdRef を呼び出し、id の情報を表示する。

関数 PrintIdName(Id_c *id): 変数 char str[]に名前を格納し、関数名もあれば追加で格納する。その後 str を表示する。

関数 PrintIdType(Id_c *id): 型名を表示する。

関数 sprintParatp(char *str, Id_c *id): 関数_PrintIdType を呼び出し、変数 str に引数名を連結する。

関数_PrintIdType(char *str, Type_c *itp): 変数 str に引数名を連結する。

関数 PrintIdRef(Id_c *id): refinenum を nextlinep が null になるまで表示する。

関数 AddIdName(char *name,int ispara, int top_next, Id_c **id): Id_c *newnode に各情報を追加し、id の末尾に追加する。二重定義のエラーも検査する。top_next で Head_Name_Num を変化させ、連続した定義に対応している。ローカル関数内であれば newnode->procname にローカル関数名 procname_t を代入している。

関数 CompareID(Id_c **id): 現在の語をリスト id で探し、あれば使用された行番号を保存する。global で使用された場合は global を探索し、local で使用された場合は local を探索する。グローバル変数を local で使用された場合に対応して、最後に global を探索する。もし一致しなかった場合は宣言されていない変数としてエラーメッセージを表示する。

関数_CompareID(Id_c **id): 現在の token がすでに登録されていないかを確認する。すでに登録されていれば 1 を、そうでなければ 0 を返す。

関数__CompareID(Id_c **id): 現在の関数で再帰的に同じ関数が使用されているかを確認する。再帰的に使用されていれば 0 を、そうでなければ 1 を返す。

関数*CreateLine(int reflinenum): Line_c *l に情報を付加し、l を返す。

関数 AddIdType(int token, Id_c **id): Type_c *newtype に情報を付加し、id を Head_Name_Num まで進め、Last_Name_Num まで id に newtype を連結する。array 型の場合は token に 100 を加えて判別している。

関数 SetProcName(): procname_t に現在の関数名を保存し、ProcFlag を 1 にする。

関数 FreeProcName(): procname_t を開放し、ProcFlag を 0 にする。。

関数 SetFormalNum_t(): SetFormalNum に Last_Name_Num を代入する。

関数 AddFormal(int token, Id_c **id): 引数として「int token, Id_c **id」を取る。id を SetFormalNum まで進め、仮引数の情報を付加した Type_c *T を連結する。

その他、以前作成した関数

関数 ValiableSequence(int top): 内部に関数 AddIdName を追加。

関数 Type(): 関数実行時に num_attr=0,Numal_Flag=1 を追加。

関数 NomalType(): Array 型と区別するために Numal_Flag が 1 の場合に関数 AddIdType を実行する。

関数 FormalArgument(): 関数 SetFormalNum_t()、AddFormal を追加。

関数 SubProgram(): num_attr=0、関数 AddIdName、関数 AddIdType、関数 SetProcName、関数 FreeProcName の追加。

関数 CallSentence(): 条件文___CompareID、関数 CompareID の追加。

関数 Value(int top): 関数 CompareID の追加。

その他の関数については以前と同じである。

(2)テスト情報
(2-1)テストデータ　　送信日時　　2015/12/21/15:24
sample021.mpl　sample11p.mpl　sample16.mpl　sample24.mpl　sample33p.mpl
sample022.mpl　sample11pp.mpl　sample17.mpl　sample25.mpl　sample34.mpl
sample023.mpl　sample12.mpl　sample18.mpl　sample26.mpl　sample35.mpl
sample024.mpl　sample13.mpl　sample19p.mpl　sample27.mpl
sample025.mpl　sample14.mpl　sample21.mpl　sample28p.mpl
sample026.mpl　sample14p.mpl　sample22.mpl　sample29p.mpl
sample032p.mpl　sample15.mpl　sample23.mpl　sample31p.mpl

Undeclared_value_error.mpl　　Double_decralation_error.mpl

(2-2)テスト結果　　送信日時　　2015/12/21/15:24
sample021.mpl.txt　sample11p.mpl.txt　sample16.mpl.txt　sample24.mpl.txt　sample33p.mpl.txt
sample022.mpl.txt　sample11pp.mpl.txt　sample17.mpl.txt　sample25.mpl.txt　sample34.mpl.txt
sample023.mpl.txt　sample12.mpl.txt　sample18.mpl.txt　sample26.mpl.txt　sample35.mpl.txt
sample024.mpl.txt　sample13.mpl.txt　sample19p.mpl.txt　sample27.mpl.txt
sample025.mpl.txt　sample14.mpl.txt　sample21.mpl.txt　sample28p.mpl.txt
sample026.mpl.txt　sample14p.mpl.txt　sample22.mpl.txt　sample29p.mpl.txt
sample032p.mpl.txt　sample15.mpl.txt　sample23.mpl.txt　sample31p.mpl.txt

Undeclared_value_error.mpl.txt　　Double_decralation_error.mpl.txt

(2-3)テストデータの十分性
　頭文字に「sample」のつくテストデータ 31 種類で正常な動作、エラーが発生する動作の確認を行った。これらでブラックボックステストは達成された。
　「Undeclared_value_error.mpl」、「Double_decralation_error.mpl」でブラックボックステストでは表示されなかったエラーをテストした。。これでホワイトボックステストは達成された。

(3)進捗状況

(3-1)事前計画

開始予定日	終了予定日	見積もり時間	作業内容
12/1	12/5	10	課題達成のための方針を立てる
12/5	12/10	20	リストに各値を代入し、表示する
12/10	12/15	20	課題達成のために他の機能を追加する
12/15	12/20	5	レポートの作成

(3-2)事前計画の立て方についての前課題からの改善点

課題 2 を行った際に、実験で忙しくて手を付けるのが遅れ、結果としてしわ寄せが提出前に集まってしまった。今回は重い実験もないため、早めに手を付けるように計画を立てた。

(3-3)実際の進捗状況

開始予定日	終了予定日	見積もり時間	作業内容
12/1	12/3	10	課題達成のための方針を立てる
12/5	12/15	20	リストに各値を代入し、表示する
12/15	12/17	10	id を比較するための関数の作成
12/17	12/20	10	サンプルを実行してのデバッグ
12/20	12/21	5	レポートの作成

(3-4)当初の事前計画と実際の進捗との差の原因

時間的余裕はあったが、だらだらとプログラムを書いてしまった。そのために友人が 2 日でできたことに 1 週間もかかってしまった。また kit スタンダードの存在を忘れており、課題の提出と試験が重なってしまった。よって結果的に早めに課題を達成した。