プロジェクト演習2週目レポート

学籍番号：744177
氏名：上田晃

・担当したソースコード

exp_imp.h
expnode.h
expression.c
expression.h
exptree.c
functions.c

全体としてうまく説明できており、良いと思います。
他の箇所との関係も把握して、機能の拡張方法についても
検討を始めましょう。

・exp_imp.h
struct _expnode、struct _oprExpnode、struct _argExpnodeでは、token.hで宣言されているtoken型のtok_id、tok_num、tok_str、などのkindとprefixはそれぞれ共通したものとなっている。 kindには式や数字などが、prefixには0や'+', '-','not'が入る。

構造体_expnodeにはunionという複数の型を同一のメモリ上で共用することができる構造が含まれている。kindが数字または文字列として代入されている時にlong型として、kindが変数名として代入されている時に構造体varinfoとして扱うことができる。この共用体をvとして扱うことができる。 うまく説明できています。

構造体oprExpnodeにはkindとprefixの他に*operand[2]、長さ2の配列が用意されている。operand、被演算子が入るため2つ分の長さである。被演算子とは'1 + 2'とある場合に演算子ではない部分である'1'と'2'を指す。

構造体_argExpnodeには*args[1]、長さ1の配列が用意されているのは、exptree.cの関数newArgnode()で必要になるからである。これはexptree.cで説明する。

・expnode.h
expnode.hではexptree.cで扱う関数のプロトタイプ宣言を行なっている。

この整数値は、文字列リテラルに順番に割り当てられる通し番号です

・expression.c
用意された文書に乗っていること以外の部分について説明する。
関数strExpression()では、関数fgetItem()で得たトークンが文字列の場合にexptree.cの関数newExpnode()を呼び出し新しく式に相当する構文木をlong型のintvalueに当てはめた後に関数expression()に戻り値を返す。
文字列ではなかった場合は？
関数expressionList()では、連続している演算子を関数expression()でまとめつつ配列に加えて式の優先順位を()で括った構文木を配列として納める。
例を挙げると、
もちろん実際に ( ) を使っているわけではありません
１＋４×３の式の場合に
（１＋（４×３））という風に計算順位を括弧で括った構文木をを配列として納める。

・expression.h
expression.hではexpression.cで扱う関数のプロトタイプ宣言を行なっている。

・exptree.c
関数newExpnode()では、プログラム実行時に診断機能を付け加えることができるassertが宣言されている。assertには真となる評価式を渡すことが前提となる。評価式の結果が真の場合にはそのままプログラムが実行していく。評価式の結果が偽の場合に診断メッセージを出力してプログラムを終了する。exptree.cでは、kindの値がトークン型の数字、文字列、変数名である場合にはプログラムをそのまま実行されるが、それ以外の場合にabort.cの関数_assert()が呼び出され診断メッセージを出力して終了する。
assert機能をCで扱う場合にはassert.hをインクルードしないといけないのだが、今回のプログラムの中でインクルードしている場所が見つからなかったため、どのようにしてインクルードをしているのかがわからなかった。
今回は自前のマクロ定義があります
assertの評価式が真の場合にmalloc(size)によりsize分のメモリ領域を確保した後、構造体xpのkindとprefixに元のkindとprefixを代入して構造体xpに戻り値を返している。このsizeが構造体で宣言されている場合はその構造体が占有しているバイト数分を確保する。

関数newOprnode()では、malloc()でメモリ領域を確保した後に関数名の通りに被演算子をxp->operand[0]とxp->operand[1]にそれぞれ代入している。

関数newArgnode()では、まず、変数szに構造体argExpnodeが占有しているバイト数を代入したあとargnumが1より大きい、つまり関数名の量が1より大きい場合にその量分のメモリ領域を確保するためにszを増やしている。その後関数の名前やその情報を構造体agpに代入している。
関数名ではない
下の段落の記述が正しいです
exp_imp.hの構造体_argmExpnodeで*args[1]だけ用意されているのは、引数の個数に応じて配列に入れるためである。34行目の計算では、メモリ領域を引数の個数分だけ変数szに追加することで解決している。

関数term()はプログラムの動きが複雑であまりよくわからなかったが、termの英単語の意味を調べると期間、用語など色々な意味があった。プログラムのコメントを見ると、二項演算子や単項演算子など数学的な内容が多かったため、項が適切な意味であると思われる。
また、この関数の説明で用意された文書には項に相当する構文木（expnodeによる木構造）を返す関数 term()とあった。構文木を返すということが詳しく理解できていないので今後学ぶ必要がある。

プログラムの構文を、節である構造体をポインタで参照し合う木構造で表現しますが、その木構造の根に当たる構造体を参照するポインタを渡せば、木全体の情報を渡すことになります。

・functions.c
functionとある通り、関数に対して情報を格納するプログラムである。

functions.cはstatements.hでプロトタイプ宣言されている。まず関数parseProgram()から呼び出され、fgetItem()で得たs.tokenがサブルーチン定義（関数の予約語'func'または手続きの予約語'proc'）または変数宣言（局所(ローカル)変数の予約語'var'）またはサブルーチンの前方参照（予約語'declare'）かによって関数funcDefine()またはstatements.cの関数varsDeclareList()または関数funcDeclare()が呼び出される。

変数宣言以外の時に関数func_header()が呼び出され、処理をした後に関数parameter_list()が呼び出される。

関数parameter_list()では、func '関数名'(引数1、引数2)などの引数を読み込む。引数がない場合（）は0を返す。引数がある場合には（左括弧から）右括弧まで読み込む。
変数宣言の場合はstatements.cの関数varsDeclareList()が呼び出され、varに続く変数名を読み込む。変数宣言が複数存在している場合は、コンマがなくなるまで読み込まれる。

func_header()の頭の処理からparameter_List()が呼び出されるまでの処理がよくわからなかった。

サブルーチンは、前方宣言がある場合とない場合、前方宣言と本体定義が一致する場合と一致しない場合、同じ関数名で定義が違う場合など、いろいろなケースを考えて対応する必要があります。
また、仮引数の列をローカル変数に準じるものとして登録する必要があります。ここに同じ名前が含まれていると変数と同様にエラーにしなければなりません。
さらに、仮引数も含め、ローカル変数のスコープの管理をする必要がありますが、
サブルーチン本体の定義部分ではその最初の処理も行う必要があります。