プロジェクト演習

前回の指摘事項が改善されているとは言えません。
とにかく、説明が少なすぎます。
ヘッダファイルを担当するなら、そのデータ構造がプログラムで
どのような役割を持っているかなどを、他の人が読んで理解できる
ように説明すべきです。

氏名：　古本　涼
学生証番号：　745031

担当したソースコード

(1)newnode.c

(2)stnode_imp.h

(3)stnode.h

(4)symset.c,　symset.h

(5)token.h

(1)newnode.c

newNode では enum 型で列挙された nodetype(数値)を取る。ここで、stnode のサイズをバイト数で返し、関数 malloc でそのバイト数のメモリを確保し、assign, return, while, for, break のいずれかにノードを割り当てていると考えられる。

また、newNodeExpand でも同様に、if, input, call, print, println のいずれかにノードを割り当てていると考えられる。offsetof マクロで指定した構造体の先頭から指定した構造体メンバまでのバイト単位を返している。

それは何に使うのですか

(2)stnode_imp.h

それぞれ構造体のプロトタイプ宣言を行なっている。構造体_argnode では union 宣言を行なっている。union 宣言は共用体という考え方で、同一のメモリ領域をそれぞれ異なる型の別の変数名で操作できるようにするものである。

構造体 assign では assign と return のことが記述されており、assign の時は global は変数が global 変数であるか、offset はメモリの割り当て位置を示していると考えられる。

構造体 ifclause では if 構文のことであるがポインタの使用を宣言している。

構造体 whilenode や fornode でもポインタの使用を宣言し、また global 変数と offset も示している。これは、構造体 varinfo でも行なっており変数リストの位置を示している。構造体 argnode では、offset は proc のインデックスを示している。

(3)stnode.h

enum 宣言で列挙し、node_assign から順に 0,1,2.....と int の値を持たせている。16 行目から 29 行目は構造体宣言を行い、typedef 宣言で同意義の別名を宣言している。

また構造体 stnode では*next で構造体を自己参照し、リスト処理を行なっていると考えられる。

(4)symset.c, symset.h

構文解析中に出現する可能性のある token の集合を表現するために使用される。

関数 symsetCreate で token_t 型の配列に入れられた要素をビット列に変換し、関数 symInitialize で集合を設定する。関数 symsetUnion では、引数 a と引数 b の表す集合の和集合を、新たに a の値にする。関数 symsetAdd は、第 1 引数の集合に、第 2 引数の要素を追加します。関数 symsetHas は、第 1 引数の集合に、第 2 引数の要素が含まれているかを調べている。

c のソースファイルはコンパイルした際に関数や変数は 1 つのモジュール単位でしか参照関係が機能しないため extern 記憶域指定子でプロトタイプ宣言を行なうことで参照可能にしている。

(5)token.h

enum token 型で tok_id から順に 0,1,2....と int の値を持たせている。combined_symbol_0, reserved_word_0, all_nomal_symbols はこれまでの値を表しており、token の一部ではないため次の token に代入している。35 行目の unsigned char 型は 8 ビットで表し、0~255 の範囲を表す。また 37 行目の unsigned short は unsigned short int のことだと考えられ、16 ビットで 0~65535(2^{16}-1)の範囲を表している。

token はコードを分解していき識別子、記号などに分解されたもので characters は",#,$などの記号に対応しており定義されているまた sym_xxx は使わない記号を当てはめている。reserve words は名前の通り、and, break, call などの予約語を対応させている。このグループでは repeat 文の導入があるので token に予約語として repeat 文を作成し sym_repeat などとして対応させる必要があると考える。

token	文字	token	文字	token	文字
sym_quo	"	sym_eq	=	sym_decl	declare
sym_shp	#	sym_gt	>	sym_do	do
sym_doll	$	sym_ques	?	sym_else	else

sym_pcnt	%	sym_at	@	sym_elsif	elsif
sym_amp	&	sym_lbk	[	sym_end	end
sym_squo	'	sym_bsls	¥	sym_for	for
sym_lpar	(	sym_rbk	]	sym_func	func
sym_rpar	)	sym_us	_	sym_if	if
sym_ast	*	sym_lbrace	{	sym_input	input
sym_plus	+	sym_vbar	|	sym_not	not
sym_comma	,	sym_rbrace	}	sym_or	or
sym_minus	-	sym_leq	<=	sym_print	print
sym_dot	.	sym_geq	>=	sym_println	println
sym_sls	/	sym_equal	==	sym_proc	proc
sym_col	:	sym_neq	<>	sym_return	return
sym_scol	;	sym_and	and	sym_step	step
sym_lt	<	sym_break	break	sym_then	then
sym_to	to	sym_var	var	sym_while	while

表の形にすると分かりやすいので、まあ意味はそれなりにあると思いますが、もっと他に頑張る所があるのではないですか。