初回のレポートとしては大変よく内容を把握できていると思いました。
この調子で頑張りましょう。

プロジェクト演習

学生番号：744221
氏名：　榎本 啓希

- 自分が担当したソースコード
  - getitem.c, getitem.h
  - getsymbol.c, getsymbol.h
  - identifires.c, identifires.h
  - idtable.c, idtable.h
  - main.c

EOFのことですね。
「終端文字」とはあまり聞かない古い(?)言葉ですなあ。
UNIXには終端文字と呼ばれる「文字」はありません。

- getitem.c, getitem.h

getitem.c には item を返す get_identifire と fgetItem, bool を返す fgetEOF, void 型の ungetItem という関数があった。ここでの item はヘッダーファイル(getitem.h)の方に構造体として定義されている。

関数 fgetItem はソースコードの 1 文字を解析する関数である。渡された空白ではない文字を評価して得られた文字を分類し、ca_sym ならシンボル、ca_quot なら文字列を取得する。この関数は、シンボル()や文字列、数値、変数名を受け取ることから、ファイルの終端文字は予期していないため、abortmessage として表示している。

関数 get_identifier は fgetItem が alpha を取得したときに渡される。この関数の詳しい処理は理解できなかったが、おそらく current_only が bool として渡されており、説明文にも新しい identifier ならテーブルに追加するという旨が書かれているため、これは current_only が true の場合はローカル変数を、それ以外はグローバル変数を探すメソッドであると思う。新しい変数が呼ばれた場合は記憶する機能も持っていると考えられる。

だいたいその通りです。

- getsymbol.c, getsymbol.h

getsymbol.h は getitem でも読み込まれており、fgetItem での文字の判別に使った ca_nul など、キャラクターの属性の定義を列挙型で行なっている。

文頭で define されている [SY, DG, AL]は(* | ca_instr)で定義されているのはビット単位の OR であるため、1 バイト目でそれぞれの判別をつけると同時にこれらは string であることを 2 バイト目に示しているものだと解釈した。

だいたいその通りです。バイトではなくてビットの操作になります。

getsymbol.c で定義されている chTable は ASCII コードの属性がマッピングされており、ch2symTable では chTable のシンボル部分が詳細に定義されている。対応表を次ページに示しておく。

ch2symTable 対応表

10 進数	16 進数	ASCII	ch2symTable
33	0x21	!	sym_xxx
34	0x22	"	sym_quo
35	0x23	#	sym_shp
36	0x24	$	sym_doll
37	0x25	%	sym_pcnt
38	0x26	&	sym_amp
39	0x27	'	sym_squo
40	0x28	(	sym_lpar
41	0x29	)	sym_rpar
42	0x2a	*	sym_ast
43	0x2b	+	sym_plus
44	0x2c	,	sym_comma
45	0x2d	-	sym_minus
46	0x2e	.	sym_dot
47	0x2f	/	sym_sls
48~57		0~9	0

これはすごいですね！

58	0x3a	:	sym_col
59	0x3b	;	sym_scol
60	0x3c	<	sym_lt
61	0x3d	=	sym_eq
62	0x3e	>	sym_gt
63	0x3f	?	sym_ques
64	0x40	@	sym_at
65~90		A~Z	0
91	0x5b	[	sym_lbk
92	0x5c	¥	sym_bsls
93	0x5d	]	sym_rbk
94	0x5e	^	sym_xxx
95	0x5f	_	sym_us
96	0x60	`	sym_xxx
97~122		a~z	0
123	0x7b	{	sym_lbrace
124	0x7c	|	sym_vbar
125	0x7d	}	sym_rbrace
126	0x7e	~	sym_xxx
127	0x7f	DEL	0

関数 chAtrribute は整数値 x を引数とし、chattr_t 型を返す。この時 x が EOF(終端文字)を表す場合 null を表す ca_nul を返し、そうでない場合は chTable の 1 バイト目のみ参照して chattr_t として返している。

これは構造体の「不完全型宣言」と呼ばれるものです。
構造体を扱わせたいが、内容の詳細までは明らかにする必要がないとき、「こういう名前が出てきたらとりあえず構造体だよ」とコンパイラに教えます。構造体のポインタを扱うだけなら、これでコンパイルが可能です。
構造体の大きさ、構造体のメンバが必要な場合には対応できません。

関数 getnumber はファイルテキストの TIP と ASCII コードの数値で渡される ch を引数として long の数値を返している。渡される数値が ASCII コードであるため、'0'を引くことで制御文字を引いた人間が扱っている数値を返すことができている。1 文字ずつ読み込むが while 文の処理によって複数桁にも対応できるようになっている。

つまり10進数ということです

関数 getsymbol では、ch と TIP を引数としてシンボルを解析して適切な token を探し、item として返している。それぞれのシンボルは token.h で割り振られており、それに応じて解析がなされている。

get_string では、TIP を引数として、クォートが出てくるまで buffer に文字列を溜め込み、最終的には addLiteralString 関数にかけて文字列を返している。クォートからクォートまでを想定しているため、EOF が来ることは想定されていない。そのため return -1 はエラーの場合の返り値である。

- identifires.c, identifires.h

identifires.h には string_info と_idrecord が構造体として定義されている。また_idtable も定義されているが、1 行で定義された構造体は初めて見たので調べてみるとおそらく次のことを示していると解釈した。struct _idtable で_idtable という型をつくり、次の行でその型を持った idtable という変数を定義している。

identifires.c は最初に予約語が定義されていることと getItem から呼ばれていることからアルファベットから定義されている文字を探すことや、見つからなかった場合、新しく定義することができる機能を持っている。

- idtable.c, idtable.h

idtable では、identifire と連携しており、定義された変数の箱として取り扱われる。idtable では関数名にポインタ(*)が使われていることが多くこれの使い方がよくわからなかった。ポインタ関数と呼ばれているものは少し表記が違ったが、詳しく調べれば情報が得られそうであった。

関数名を見ると、Add や Pop, Search があることから、table はオブジェクトのように扱われていると推測できる。

構造体へのポインタを返す関数です。
duskulでは、「関数ポインタ」は使っていません。

- main.c

  main 関数では、最初にオプションのチェックを行いそれの有無に応じて出力を変化させている。ソースファイルが指定されていない場合や、ファイルを開けて NULL だった場合にはエラーメッセージを表示させる。

  それらのチェックが通れば、ファイルからテキストを読み、様々な初期化をし、ファイルを実行する。