前回のレポートで指摘した箇所が修正されていないままです。
拡張機能をどのように実装できるかについて、考察をしてほしかったのですが。
今回の「問い」は機能拡張をする際に把握しておくべき事項ですので、
よく理解しておいてください。

プロジェクト演習課題３ ソースコードの理解

学籍番号 745086
氏名 松浦　康大

○自分が担当したファイル

①abort.c　②abort.h　③commontype.h　④errmessages.c
⑤evaluate.c　⑥evaluate.h　⑦execute.c

①abort.c

本プログラムは不正な入力が行われた際にエラーメッセージを表示するプログラムである。

abortMessage_string 関数でメッセージの出力を行なっており、以下で記述する関数の引数によって条件分岐し出力内容が変わる。また、標準入力があれば printTextBuffer 関数を呼び出すのだが、この関数の内容がまだ読み取れていない為、どういった出力が行われるか不明である。　**前回コメント済み**

abortMessage 関数では、エラーメッセージのみを渡す。よって、エラーメッセージに加えて別に出力する必要がない場合、例えば ”~が多すぎる”、 ”~が指定されていない” などのメッセージを出力したい場合に呼び出す関数であると考えられる。

abortMessageWithToken 関数では、エラーメッセージに加えて文字列・変数名・数字などを渡す。よって、実際にエラーが起きている箇所を指定して出力したい場合、例えば ”~が間違っている”、 ”これは~ではない” などのメッセージを出力したい場合に呼び出す関数であると考えられる。

abortMessageWithString 関数も、エラーメッセージに加えて文字を渡しているが、shorten 関数の実体がまだ理解できていない為、何を出力しようとしているのか不明である。

ここで、shorten 関数の実体から部分的に読み取れるところを調べわかったことをまとめる。

assert() は Duskul処理系そのものの不具合を検出する目的で置かれており、エラーが発生したC言語のファイル名と行数が表示される。一方、abortMessageWithToken() などのエラー表示関数は、Duskulのプログラムの間違いなどを表示するために使われる。表示内容も、Duskulの間違いの箇所である。

・(cc & 0xc0)で bit 積をとっている。これを行うことで上位 2 ビットのみを見た論理積を抽出できる。つまり、cc が 11 から始まっているかを見ている。よって、cc は UTF-8 でエンコードした際の文字コードである。

・UTF-8 の符号方式では、上位 2 ビットが 11 で始まるということは複数バイト文字の先頭バイトであることを意味する。

・UTF-8 の符号方式では、上位 2 ビットが 10 で始まるということは複数バイト文字の第 2 バイト目以降であることを意味する。

assert()関数は本来成立しているはずの前提条件がなかった場合に、プログラム中のどの文でそのエラーが起こっているのかを示すものである。値が期待した通りになっているのかを診断するのに対し、abortMessageWithSToken()関数はユーザーの入力が文法的に間違っていたり、引数が間違っていた場合にその箇所を指摘するものである。abortMessageWithSToken()関数は、ユーザーの入力次第では起こり得るエラーを警告するのに対して、assert()関数は本来起こりえない異常を検知しプログラムを強制終了させる。よって、起こり得る間違いに対していちいちプログラムを強制終了させるのは間違いである。

②abort.h

関数のプロトタイプ宣言を行うことでコンパイラに関数の情報を渡している。また、#if defined(DEBUG)~#endif は、デバッグ処理のために作られており、動作の検証を行うためにプログラムを追加したりする場合は、DEBUG を定義してプログラムの宣言を行う。

③commontype.h

どのプログラムでも同様に定義される普遍な変数(true, false, bool)を定義している。 6 行目のコメントには、プログラム内で bool や BOOL を定義している

が、C 言語に実装されている_bool が将来的に bool や BOOL に変更され競合する恐れがあることを示している。15 行目のコメントには、C 言語における true は必ず 1 とは限らないため、if(式==true)と書くのは間違いであり、if(式)と書けば良いことを示している。

④errmessages.c

本プログラムは複数のエラーメッセージを構造体で格納するプログラムである。

5 行目でエラーメッセージを格納する構造体の宣言を行なっており、*tag にはエラー内容、*localized にはエラーメッセージが格納されている。

getErrorMessage 関数は、引数受け取ったエラーがどのエラー内容にあたるかを*tag から探し、一致したときに対応するエラーメッセージを返す。エラーメッセージは abort.c プログラムで出力される。

⑤evaluate.c

本プログラムは与えられた文字が Duskul の何にあたるかを評価し、種類に応じた処理を実行するプログラムである。

applyOperator 関数では、受け取った演算子が何かを判定し、それに応じた処理をスタックの上から 2 つとった数字に行う。

evaluate 関数では、受け取った文字が演算子だった場合に二項演算子か単項演算子かを評価し、二項演算子だった場合 applyOperator にて判定し、単項演算子(Duskul は+, -, not のみ)の場合それに応じた処理を行っている。次に 56 行目からのスイッチケース文では、与えられた文字が変数なのか数字なのか関数名なのかを評価している。関数と判定されるとそれがメインなのかサブルーチンなのか判定される。コールスタックの領域を超えて使おうとするとエラーメッセージが出る。

execPrint 関数、execCall 関数、execInput 関数は、それぞれの関数名から Duskul での Print 文、Call 文、Input 文の役割を担っていると考えられる。

execPrint 関数では isstring が成り立つかどうかでフォーマット指定子を char 型か int 型か振り分け出力していることがわかる。

冒頭に付けたコメント「--sp by this call」は、関数呼び出しによって push 操作を行うと訳せるため、applyOperator()関数を呼び出すことで、スタックの上から 2 つとって演算子を適用した結果をスタックに追加できるということを意味している。

二項演算子の評価の場合はそうですが、他の場合はどうですか。

容量ではなくて、そろそろ溢れるよ、というくらいの「危険水域」。厳密なものではなく、いきなりシステムがクラッシュする可能性は排除できません。

⑥evaluate.h

STACK_LOW は evaluate.c でスタックオーバーフローが起こるかの判定に使われていたため、実行中のサブルーチンの情報を格納しているコールスタックの容量であると考えられる。

extern 宣言とは宣言だけを行い定義は行わない宣言方法である。これがあることで異なるソースファイル間で変数を共有することができる。sp はスタックポインタ、localbase はフレームポインタである。

⑦execute.c

本プログラムは与えられた文が Duskul の何にあたるかを評価し、種類に応じた処理を実行するプログラムである。

execAssign 関数は、代入文の役割を担っている。

execReturn 関数は、return 文の役割を担っている。コメント文から、スタックポインタの値が戻り値になると考えられる。

前回指摘箇所

execIf 関数は、IF 文の役割を担っている。If 文を使って式が真かを判定し、真でなければ break させ、文を実行しないようにしている。

execWhile 関数は、WHILE 文の役割を担っている。C 言語の do-while 文を使って同じ動きを再現している。

execFor 関数は、FOR 文の役割を担っている。変数 step の値を条件にとっていることから、step の値にループの回数が入っていると考えられる。

ex_condition型は、ある文を実行し終わって次の文の実行に移る際、普通に終わったのか、breakや returnを実行して終わったのかを示します。
ループ構文、サブルーチンでは、この値を見て、文列の実行が終わった後の動作を決めています。

execStatements 関数では、受け取った文が代入文なのか、実行文の何文なのかを分類しそれぞれの関数に値を渡している。

subroutine 関数は関数名からサブルーチンを取り扱う関数であると考えられる。stack overflow のエラーメッセージを出力していることから、ここはコールスタックの領域を超えた場合の処理であるとわかる。

assert はプログラムの診断に使われ(デバッグ処理のようなもの)、引数が true ならそのまま実行し、false なら強制終了する。マジックナンバーとして定義されている 0xDEADBEEF は解放したメモリ領域に当てることでそこがすでに解放されていることを表している。

execWhile()関数は C 言語の while 文の実装である。do-while 文の継続条件は r が ex_normal であるかどうかとされているが、r には ex_normal の他に ex_break、ex_return がある。ex_return はそのままで return 文の時に返したものであるが、ex_break は ex_condition()関数の switch-case 文に何も当てはまらなかった時に代入されている。よって、なにも実行文がなかった時にループは終わることがわかる。

executeProgram 関数では malloc, free 関数を使ってメモリの確保、解放を行っている。

この関数の内容を理解するため、関数の型名が int 型であることに着目し、どこで呼び出されているかを調べると、main 関数の変数 code の値に代入されていることがわかった。また、この code は main 関数の戻り値つまりリターンコードである。また、呼び出されている部分がエラーメッセージを出力したすぐ後であることから、プログラムが正常に終了しなかった場合に返すコードであることが考えられる。