言語処理プログラミング 資料

１．目的

本科目は，C 言語を用いて，比較的簡単なプログラミング言語のコンパイラを作成することにより，コンパイラの基本的な構造とテキスト処理の手法を理解することを主目的とする．また，比較的大きなプログラムを作成する経験を得ることができる．

２．構成とスケジュール

本科目は，次の４ステップからなる．後の課題はそれ以前の課題で作成したプログラムを再利用もしくは修正することで達成できるように設定されている．なお，以下のプログラム作成課題は，ANSI-C の範囲内の C 言語及びその標準ライブラリ（ただし，C99 以降に追加された機能はコンパイラによってはサポートされていないことがあるので使わないこと）のみを用いて作成すること．

（１）課題１：字句の出現頻度表の作成

課題内容関連講義予定日：9/28

演習期間：9/28 - 10/18

レポート・プログラム提出期間：10/19 - 10/26(月)

課題の概略：プログラミング言語 MPPL（別紙参照）で書かれたプログラムらしきものを読み込み，字句（トークン）がそれぞれ何個出現したかを数え，出力するプログラムを作成する．

キー技術：テキスト処理，字句解析

（２）課題２：プリティプリンタの作成

課題内容関連講義予定日：10/19

演習期間：10/19 - 11/15

レポート・プログラム提出期間：11/16 - 11/23(月)

課題の概略：プログラミング言語 MPPL で書かれたプログラムを読み込み，構文エラーがなければ，入力されたプログラムをプリティプリントした結果を出力し，構文エラーがあれば，そのエラーの情報（エラーの箇所，内容等）を少なくとも一つ出力するプログラムを作成する．

キー技術：再帰呼び出し，構文解析

（３）課題３：クロスリファレンサの作成

課題内容関連講義予定日：11/16

演習期間：11/16 - 12/20

レポート・プログラム提出期間：12/21 - 1/7(木)

課題の概略：プログラミング言語 MPPL で書かれたプログラムを読み込み，コンパイルエラー，すなわち，構文エラーもしくは制約エラー（型の不一致や未定義な変数の出現等）がなければ，クロスリファレンス表を出力し，エラーがあれば，そのエラーの情報（エラーの箇所，内容等）を少なくとも一つ出力するプログラムを作成する．

キー技術：データ構造，ポインタ，記号表

（４）課題４：コンパイラの作成

課題内容関連講義予定日：12/21

演習期間：12/21 - 1/25

レポート・プログラム提出期間：1/26 - 2/1(月)

課題の概略：プログラミング言語 MPPL で書かれたプログラムを読み込み，コンパイルエラー，すなわち，構文エラーもしくは制約エラー（型の不一致や未定義な変数の出現等）があれば，そのエラーの情報（エラーの箇所，内容等）を少なくとも一つ出力し，エラーがなければ，オブジェクトプログラムとして，CASLII（別紙参照）のプログラムを出力するプログラム（すなわちコンパイラ）を作成する．

キー技術：コンピュータアーキテクチャ（命令セットレベルアーキテクチャ），コード生成

３．レポートについて

各課題について，レポートを提出すること．一つの課題でもレポートが提出されない場合には，言語処理プログラミング全体が成績評価対象外になる．

レポートには，次の事項を含むこと．

（１）作成したプログラムの設計情報

(1-1)全体構成：どのようなモジュールがあるか，それらの依存関係（呼び出し関係やデータ参照関係）

(1-2)各モジュールごとの構成：使用されているデータ構造の説明と各変数の意味

(1-3)各関数の外部（入出力）仕様：その関数の機能．
引数や戻り値の意味と参照する大域変数，変更する大域変数などを含む

（注）ただし，それ以前のレポートに記載した内容と同じである場合には，その旨のみを記述するだけでよい．たとえば，関数○○が課題１で説明済みであり何も変更されていなければ，

関数○○：課題 1 レポートで記載済み

と書くだけでよい．

なお，**モジュール**とは，意味的にまとまったプログラムの部分である．小さなプログラムの場合は関数一つ一つをモジュールと考える場合もあるし，いくつかの大域変数を共有して使う関数群を（その大域変数も含めて）モジュールとする場合もある．たとえば，課題１で作成する字句解析系はモジュールである．

（２）テスト情報

(2-1)テストデータ（既に用意されているテストデータについてはファイル名のみでよい）

(2-2)テスト結果（テストしたすべてのテストデータについて）

(2-3)テストデータの十分性（それだけのテストでどの程度バグがないことが保証できるか）の説明

（３）この課題を行うための事前計画（スケジュール）と実際の進捗状況

(3-1)事前計画（スケジュール）

当初の計画と，演習中に計画を大きく修正した場合には修正後の計画（最終分だけでよい）を記載すること．計画を立て，〆切までに完成したプログラムとレポートを提出するまでが演習である．

(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 1 を除く）

(3-3)実際の進捗状況

(3-4)当初の事前計画と実際の進捗との差の原因

事前計画と進捗状況は，開始(予定)日，終了(予定)日，使用(見積もり)時間，作業(予定)内容の４項目

をカラムとし行は日付順とする表形式で記述すること(実務では線表で書かれる)．作業(予定)内容は1行程度でよい．詳細な説明は課題1の付録参照．

レポートの形式：

pdf もしくは，Word の形式のファイル．フォーマットは，A4 の用紙サイズで，表紙（1 ページ目）に課題名（「言語処理プログラミング 課題1」など)，提出日の日付，学籍番号，氏名のみを記載すること

提出先：Moodle 上の指示されたところ

提出期間：各課題提出期間

４．プログラム提出について

各課題で作成したプログラム（ソースプログラム）もレポートと同様の提出期間中に Moodle を経由して提出すること．一つの課題でもプログラムが提出されない場合には，言語処理プログラミング全体が成績評価対象外になる．

提出されるべきものは，ソースプログラムファイルのみ(*.c, *.h のファイル)である．それらを同一のフォルダ（ディレクトリ）に置いて，まとめてコンパイルすれば，実行形式ファイルが生成されるものを提出せよ．

本演習は，個人演習である．各自がプログラムとレポートを提出すること．共同でのプログラム作成や他人のプログラムやレポートをコピーすることは厳禁である．他人のプログラムを見てはいけないし，見せてもいけない．類似したプログラムが発見された場合には，双方に再提出を求める（再提出する余裕時間がある場合）か，ゼロ評価（再提出する余裕時間がない場合）となる．

なお，みなさんの作成環境とこちらのテスト環境が異なっている場合に（OS，文字コード，コンパイラなどの違いにより）問題（こちらでコンパイルエラーが出るなど）を生ずることがあった（過去の例)．コンパイルエラーなどでこちらのテストを通らない場合，最低評価となる．特に，コメント内を含めて提出プログラムには，日本語等の複数バイト文字（いわゆる全角文字など）を使わず，１バイト文字（いわゆる半角英数字）のみを用いることを勧める．

どのような環境でもコンパイル，実行できることを期待するが，みなさんがそれを確認することはかなり困難である．そのため，標準環境として，情報工学課程の演習室(8-205)の Linux 環境にある gcc を指定する（Linux 上の Eclipse 環境ではないことに注意)．ここで，コンパイル，実行ができることを確かめること．すなわち，作成したプログラムファイルが，foo1.c, foo2.c, foo3.c であるとき，

gcc foo1.c foo2.c foo3.c

のように直接 gcc でコンパイルして，できた実行形式プログラムが思った通りに動くことを確かめること．

従って，作成中は自分の使いやすい環境で，Eclipse 等の統合環境を用いるのが便利と思われるが，最終的な確認は Linux 上の shell で Makefile を用意して行うのが確実である．

提出プログラムが複数のファイルになる場合には，zip など標準的な形式で一つのファイルにまとめて提出してもよい．zip 以外の形式でまとめられた場合など，こちらで取り出せない場合には評価されない．

５．質問，連絡，資料等の配布について

・教員から受講生のみなさんへの連絡や資料配付は，各課題の説明会の他，Moodle を用いて行う．

・質問は，いつでも電子メールで受け付ける．質問は，

tsujino@kit.ac.jp

へ送ること．メールの本文の最初に学生番号と氏名を明記すること．プログラムを見せながら質問したいな

ど対面での質問を希望する場合には，時間割上の演習時間(月曜日 2 時限）の初めに 8-205 演習室で受け付ける(質問者が途切れるまで)．また，月曜日２時限以外の時間に，直接，辻野教授室(8 号館 2 階 8-203 室）まで質問しに来てもよいが，不在や会議等で質問を受け付けられない場合がある．

６．実際の演習について

演習環境としては，月曜日２時限に，8-205 演習室が確保してあるが，密になるのを避けるために，同時間帯に CIS 演習室も利用可能である．実際には，C コンパイラ等があれば演習可能なので，上記の確認作業以外は，個人持ちの PC を含めてどこで演習を行ってもよい．実際週 1 コマの演習だけでは完成しないと思われるので，十分な時間外演習を期待する．

[参考：「プログラミング言語論配布プリント 2008 年度版」より抜粋]

2.11 テスト

ソフトウェア開発がモジュール化に基づいて行われていなるならば，テストは単体テストと統合テストに分けられる．

統合（結合）テスト：

テスト（単体テスト）済みのモジュールをすべて接続して行うテスト．ここで問題が起こると，モジュール間の仕様の不整合を意味するので，大きな手戻りが起こる可能性がある．

単体テスト：

モジュールごと，関数や手続きごとに行われるテスト．実際の関数やモジュールは他の関数を呼んでいたり呼ばれていたりするので，単体テストを行うためには，テストされるモジュールだけではなく，次のものも準備しなくてはならない．

・テスト用メインプログラム：テストされるモジュールを呼ぶメインプログラム．モジュールの仕様に従って作成される．ドライバ(driver)やテストドライバとも呼ばれる．

・うめ草：テストされるモジュールが呼び出す関数やモジュールのうち,まだ完成していないものの代わり．通常はどのような引数で呼び出しても同じ値を返す関数であったり，呼び出されるとテスタ（人）に返す値を聞くように作られる．スタブ(stub)とも呼ばれる．

・テストデータ：入力のパターンをすべて網羅するだけのテストデータセットを用意しなくてはならない．

これらはモジュールのプログラミングと並行して準備されなくてはならない．他のモジュールができていないからと言って，テストできないということがあってはならない．

また，単体テストはモジュール内部の実装に依存するホワイトボックステストと，依存しないブラックボックステストに分けられる．

ブラックボックステスト(black box test)：

モジュールの外部仕様のみをみて，その仕様が満足されるかどうかを調べるためのテストデータを用意するテスト．標準的なテストデータの他に，境界値を用いたテストデータ，すなわち，仕様上許されるテストデータぎりぎりの値やそれを越える値を用いたテストデータを用意する．たとえば，データ数が 0 個の場合とか，上限値の場合，それを越える場合などである．境界値のあたりはバグが生じやすいところであり，場合によっては，初期値が境界値であるなどして必ず境界値での実行があることもあり，必ずテストしなくてはならない．また，入力として許されないようなデータでもテストするのは，その場合に無限ループに陥ったりシステムダウンしたりしないことや，正しくエラー検出ができることを調べるためである．

ブラックボックステストは，システムの開発者でない人が，テストデータを用意して行うことが望ましい．なぜなら，システムの内部設計を知っている人が行うと，その知識に影響され，外部仕様ではなく内部仕様に従ってテストを行いがちであるためである．その場合には，システムの開発者が外部仕様を誤って理解していた場合に生ずるバグが検出されないことになる．

ホワイトボックステスト（white box test）：

モジュールの実装に基づいてテストデータを用意するテスト．実装に基づいてテストがどの程度十分かを評価できる．標準的な基準に次の二つがある．

・C0 カバレッジ（命令網羅）：モジュール中のすべての文を実行するようにテストデータを用意する．もちろん，一組のテストデータでは無理なので，複数組のテストデータを用意する．
・C1 カバレッジ（分岐網羅）：モジュール中のあらゆる分岐の組み合わせを実行するようにテストデータを用意する．たとえば，テストしたい関数が二つの if 文が並んでいるものであれば，4 通りのテストパスがあり，4 組のテストデータが必要となる．

C1 カバレッジの方が C0 カバレッジよりも強力なテストであるのは自明であるが，モジュールサイズが大きくなった場合，コストや時間の制限から C0 カバレッジがやっと，という場合も多い．そのような場合には，C0 カバレッジが 100%，C1 カバレッジが 30%，というような表現になる．なお，通常実行されない部分，例えば，バグを早期に検出するための if 文による分岐などはカバレッジ率の分母に入れなくてよい．また，while 文などでは本体が 1 回も実行されない場合と複数回実行される場合をテストすれば，C0 でも C1 でもカバーされたとしてよい．