情報処理 第10回
初めてのC言語

2000年6月28日

目 次

1 本日の目標 **1**

2 前回の補足 **1**

3 この講義では **1**

4 C 言語の欠点 **2**

5 Hello, world **2**

5.1 ソースファイル hello.c の作成 2

5.2 実行ファイルを作成 3

5.3 Hello, world プログラムの説明 3

6 cc の動作と用語説明等 **4**

7 繰り返し **5**

8 付録 **6**

8.1 参考文献について 6

8.2 プログラミング言語の種類 8

1　本日の目標

- C 言語の簡単なプログラムに触れる.

2　前回の補足

(1) LaTeX　のファイルを書く場合の最低限のルールは

```
\documentstyle{jarticle}
\begin{document}


（本文）


\end{document}
```

です. title, author, date をつけたいときは, \begin{document} のあとに,

```
\title{xxxxx}
\author{xxxx}
\date{xxxx}
\maketitle
```

を書き, そのあとに本文を続けていけばよろしいです.
そしてセーブするときは拡張端子の.tex を忘れずに.

(2) 先週のレポートは LaTeX のファイルを指定の場所に送って下さい. dvi ファイルはだめです.

3　この講義では

C 言語を用いたプログラミングの入門を, 講義します. 計算機を実際に動かしてプログラムを作成してもらいます. 内容は, 主に C 言語の文法の話です. 後期セメスターの原先生の授業で詳しく触れるはずです.
何故 C 言語をするかは次の理由からです.

1. 最近の流行である.
   書店のコンピュータプログラミングのコーナーにいきますと, C 言語の本が多数並んでいます. また, Netnews でも C 言語に関する newsgroup がありいろいろな議論がかわされています. この様に, 勉強するにあたって色々な情報が身近にあるというのは都合がいいことです.
2. UNIX との親和性が高い
   C 言語はその UNIX の開発言語です. 従って, UNIX には C 言語でのプログラミングを補助する色々なツールがそろっています. (補助する道具については, どの程度授業で述べられるかは, わかりませんが...)

4　C 言語の欠点

C 言語は, UNIX の開発に際して assembly 言語 (意味は後記のおまけ参照。以下同様) の代わりとなるべく作られました. 従って, 比較的低級な言語です. と言うことは, computer の Hardware の簡単な仕組を理解していることが暗黙裏に要求されたりすることがあります. また, 初期の UNIX 開発のターゲットマシンだった Dec PDP-11 の機械語命令の影響を受けていまして, 必ずしも汎用性のある言語仕様だとは言えない部分があります.

さらに, 1980 年代には C 言語に様々な方言が存在しました. これは, 初期の C 言語の言語仕様がきっちりしていなかったことが原因なのですが, これを何とかしようということで, ANSI で標準化作業がなされました. (これに対して例えば Pascal は, 開発者が言語仕様をきちんと決めています.) そして出来たのが ANSI-C です. しかし, この標準化作業がいい格好で終わったとは言えない部分が多くあります. ということで, C 言語は初心者がいきなりはじめるには不都合なところが残念ながら多くあります.

また, 初心者にとっては C 言語の言語仕様の小ささ (機能が少ない) も欠点となります. プログラミング言語としては, 言語仕様の小ささというのは大した問題ではないのですが, 初心者には取っ付きにくいものになります. 例えば, コンピュータのプログラムを書く上で, 入出力というのは基本的に必要なものなのですが, C 言語そのものには, 入出力の仕様がありません. C 言語では, 多くの処理をあらかじめ作られたライブラリというプログラムに任せます. ということは, このライブラリ[1] の使い方をある程度勉強する必要があります.

5　Hello, world

5.1　ソースファイル hello.c の作成

まずは C 言語を使った最も簡単なプログラミングを作ってみましょう . 次のプログラムは , C 言語の入門では必ず一番最初に書くものです .

画面に Hello, world と表示するプログラム . ファイル名 hello.c

Mule を用いて , hello.c という名前のファイルに次の内容を入力し , セーブしてください . /* と*/ で囲まれた部分はタイプしなくて良いです .

```
-----この下から-----
/* Hello, World を出力するプログラム */

#include <stdio.h>

main()
{
        printf(" Hello, world\n");
}
-----この上まで-----
```

/* と*/ で囲まれた部分はコメント文と呼び , プログラムの説明などを書きます . この部分はなくてもプログラムの実行には関係しません .

[1] 文字, 文字列, 入出力, 数値関数, 日付の変更, 動的な記憶領域確保など

授業中はタイプの時間の節約のため，コメント文を書く必要はありませんが，レポート課題ではきちんとコメントをタイプするようにして下さい．

注意 コメント文が必要な理由の一つは, 基本的にプログラムは他人も見ることを前提に書かれるべきだからです. それ故, 誰が見てもわかるようにコメントを入れておくとそのプログラムの改良などの作業がしやすいわけで, 例の 2000 年問題はその点修正が苦労したそうです. 構造化プログラミングの思想にのっとって, 誰にもわかりやすい,「数学屋のような癖」のないプログラムを書くようにしましょう.

5.2 実行ファイルを作成

ソースファイル hello.c から実行ファイル a.out を作成します．

(1) `cc  hello.c` とタイプします．
3 - 5 秒後に再びプロンプトが現われます．a.out という新しいファイルが出来ていることを ls コマンドで確認しましょう．
(2) このファイルは，実行可能ファイルです．
(3) `./a.out` と入力します．

上の操作をソースファイルをコンパイルして実行するといいます．

コマンド cc がうまく実行できなかった人は，次をチェックしてください．

1. ファイル名は，hello.c となっているか．
2. コマンド又はファイル名を正しくタイプしたか ?
3. hello.c の中味の括弧や quotation は，正しく入力されているか ?
4. #include の # は行の先頭にあるか ?
5. printf 文の最後にセミコロン ; がついているか ?

5.3 Hello, world プログラムの説明

```
#include <stdio.h>
```

#で始まるこの部分は，C プリプロセッサ (preprocessor) cpp への命令です．(とりあえずは，おまじないと思っていて下さい．)

```
main()
```

main と言う関数の始まりを示します．C 言語では，必ず main と言う名前の関数が必要で，ここからプログラムの処理がはじまるようになっています．() の中には，この関数のパラメータ (通常プログラミングとか計算機の分野では，引数という．) が入ります．

```
{
        printf("Hello, world\n");
}
```

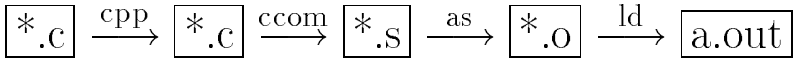
ここが，このプログラムの本体です．C 言語では，一連の処理は中括弧 { と　} で挟まれた部分に書きます．中央の

```
        printf("Hello, world\n");
```

が，このプログラムがする全てです．printf は，標準入出力ライブラリにある関数の名前です．2 重引用符 (") で囲まれた文字列を標準出力 (ディスプレイ) に表示します．最後の\n は改行を表わします．

6　cc の動作と用語説明等

コマンド cc は一つのプログラムではなく，少なくとも次のプログラム (プロセス) を作り出します．下の図の四角く囲まれた部分に書かれているのは，その時に使うファイル名です．



cpp C preprosessor: C compiler (ccom) が処理をする前に，文字列の処理の部分のほとんどを受け持って，事前に処理をするものだと思ってください．

ccom C compiler : C 言語の文法で記述された処理内容を，assembly 言語に変換するプログラム．

as Assembler : assembly 言語で書かれた内容を，機械語 (すなわち 2 進法で記述されもの) に変換するプログラム．

ld Link editor : リンカともいう．上の hello.c ではコンパイルした結果は，`Hello, world\n` という文字の並びを関数 printf に受け渡すという命令だけが，作り出されます． printf の実体は，ライブラリとしてコンピュータのファイル

```
/usr/lib/libc.a
```

に入っています．ld は，hello.c のコンパイルしてできた結果と上のファイルにある printf を結び付けて (リンクするという)，さらにシステムが，実行時に要求する情報をつけて実行可能ファイルを作り上げます．

Library プログラムを組むと分かりますが，画面の入出力などはどのプログラムでも同じ様なことをします．これをいちいちゼロからプログラムしたのでは，無駄が多いのです．そこで，よく使う処理はあらかじめ 2 進法レベルのものを作成，まとめたのが必ずあります．これがライブラリ (library) です．ライブラリは，compiler(実は，リンカ) に指示することによって，自分のプログラムにくっつけることが出来ます．また，C 言語では標準ライブラリというのが ANSI により定義されており，多くの処理はこれを使って実現します．

ファイル名 C 言語のプログラミングにおいてファイル名は，次の規則にしたがって，名前をつけます．

1. *.c :C 言語のプログラムを記述したファイル．
2. *.h :ヘッダ (header) ファイル．定数とか関数の定義を記述する．
3. *.s :assembly 言語のプログラムを記述したファイル．
4. *.o :オブジェクト (object) ファイル．2 進法で記述した機械語が入っている．
5. a.out :特別なオプションを指定せずにコンパイルしたときに出来る実行可能ファイル．

上のファイルのうちいくつかは，hello.c をコンパイルするときには現われませんでしたが，それは単に目に見えないだけで，コンピュータの内部ではちゃんと作られています．

7　繰り返し

次のプログラムを 10-1.c という名前で作成して下さい．出来たら，コンパイルして実行してみて下さい．

```
#include <stdio.h>

main()
{
    int     i;          /* i = 0, 1, ..., 5 */
    int     sum;        /* 0 から 4 までの和 */

    sum = 0;    /* 変数を初期化します */

    /* for 文は繰り返しの計算に使われます */
    for (i = 0; i < 5; i++) {
        sum = sum + i;
    }

    printf("sum = %d\n", sum);

}
```

まずは，for (...) の後ろの {} で括られた部分を見てみましょう．

```
sum = sum + i;
```

はsum の値にi の値を加えたものを新しいsum の値とせよ．という意味です．この部分は

```
sum += i;
```

と書くことも出来ます．

for 文は次のような形で使用します．

```
for（式１; 式２; 式３）{
    …;
}
```

式 1 は，for 文の中の処理を行う前に 1 度だけ行います．次に式 2 を判定し，その結果が真ならば，{} で囲まれた部分を実行します．その後，式 3 を実行し，再び式 2 を判定します．あとはこの繰り返しです．

上の例の式 3 には

```
i++
```

が書かれています．これは i を 1 増やせという意味でインクリメントと呼ばれます．この部分は，

```
i = i + 1;
```

あるいは，

```
i += 1;
```

と書いても同じことです．

次の課題に取り組んでみてください.

問題 **1** 自分の名前を *15* 回出すプログラムを書け. *(*ヒント*: for* 文を用いる*)*

問題 **2** 初項 *11* 公差 *3* の等差数列の第百項までの和を求めるプログラムを書け. *(*ヒント*:* プログラム *10-1.c* を改良せよ*.)*

それぞれの subject 名を, ereportno.10-2, ereportno.10-3 とし, `osaka-report@se.ritsumei.ac.jp`までに投稿せよ.
締め切りは 7 月 4 日 (火) です.

8 付録

8.1 参考文献について

はじめのほうは, C 言語に対する参考文献で後半は, それ以外のものを挙げました.
[1] は, C 言語のバイブルです. C でのプログラミングを真面目にしたいのでしたら, 必ず買うようにしてください. なお, 最近日本語訳を修正した版が出版されています. この事実からわかるように, この本の訳は必ずしも良いとは言えず, 明らかな誤訳が多数含まれています. そういう意味では, 原書を手に入れたほうがよいかもしれません.
[9],[10] は一人でＣ言語勉強するのに最適な本であると思います. これらを読んだあと, [1] を読むことを奨めます.
[2], [3] は, プログラムを書く上での心構えを主に書いた本です. プロフェッショナルを目指すなら, 一読すべきでしょう.
[5], [4] は, C 言語を用いた数値計算法の本です. [5] には, 英語版ですが Pascal を同じ内容の本があります. [4] には, プログラムのソースファイルが入ったフロッピーディスクがついてきます.
[6] は数学者が書いた本です. そういう意味では, 取っ付きやすいかもしれません. ただそこに書いてあるプログラムのプログラミングスタイルに少々問題があります. [7] は, [6] の結び目理論にかかわる部分を, 追加して際構成した本です。
UNIX のシステムコールに関連したプログラミングの本として [8], [12] を挙げておきます. この二つは共に古いので, 取り敢えず挙げただけです.
[13] は UNIX のネットワークプログラミングの本です.
[14], [15], [16] はプログラミング全般にわたる古典的名著です.
[14] は, わかりやすいプログラミングの心得を書いたものです. 処理系には ratfor と呼ばれる独自の言語を用いて書いてあります. ratfor は皆さんが今使っている機械にもあったと思います. なお, 日本語訳はありませんが, これとまったく同じ内容を Pascal で書いた本が, 同じ著者で出版されています.
[15] は, データ構造とプログラムの関係を書いた本です. 処理系は Pascal を使っています. プログラムというのは, しょせんデータの処理方法の記述ですが, そのデータ構造をうまく解析して与えないと, よいプログラムが書けないと言う視点でプログラミングを解説しています.
[16] は, プログラムに現われる数学的構造を面白く書いた本です. 第 2 巻までは, 日本語訳があります. 著者は, この本の出来上がりが気に入らないといって TEX を開発したのは有名です.
皆さんの使っている機械では, C 言語以外でもコンパイラ系で, Fortran, Pascal, インタプリタ系で, Awk, Perl が動きます. Fortran 以外について [17], [18], [19] に各々の処理系の原典を挙げておきます.

世の中には, プログラミングの本も C 言語の本も山のように出版されています. この中で, 良い本を捜すのは大変です. 伝え聞く限りでは, 多くの本が出版されている割には良書と呼ばれるものは少ないようです. また, 翻訳本にはひどい誤訳もあるそうです. この様な情報は, たまに Net News などでも議論になっているようですから, Net News の C 関係の news group を定期的に読むことをお勧めします.

参考文献

[1] カーニハン・リッチー著, The C Programminng Language,
日本語訳 石田晴久訳, プログラミング言語 C 第 2 版, 共立出版.

[2] S. Oualline 著 岩谷宏訳, Practical C Programming 現実的な C プログラミング, ソフトバンク.

[3] 藤原博文著, C プログラミング診断室, 技術評論社.

[4] L. Baker 著 吉田弘一郎訳, C 言語数学関数ハンドブック, 技術評論社.

[5] W. H. Press 他著 丹慶勝市他訳, Numerical recipies in C, 技術評論社.

[6] 落合豊行著, C 言語による数学解析, 近代科学社 1988 年.

[7] 落合豊行, 山田修司, 豊田英美子著, コンピュータによる結び目理論入門, 牧野書店 1996 年.

[8] Curry 著 アスキー編集部訳, UNIX C プログラミング, アスキー 1991 年.

[9] 内田智史著, C言語によるプログラミング [基礎編], [応用編], オーム社 1991 年,
C言語によるプログラミング [スーパーレファレンス編], 2000 年.

[10] 結城 浩著, C言語プログラミングレッスン [入門編], [文法編], ソフトバンク 1994/1995 年.

[11] エイチアイ著, 実用 UNIX C/C++言語ハンドブック, ナツメ社 1999 年.

[12] Rochkind 著 福崎訳, UNIX システムコールプログラミング, アスキー 1987 年.

[13] W. R. Stevens 著 篠田訳, Unix ネットワークプログラミング, トッパン 1992 年.

[14] カーニハン・プローガー著 木村 泉訳, ソフトウェア作法, 共立出版 1981 年.

[15] N. Wirth 著 片山卓也訳, アルゴリズム + データ構造 = プログラム日本コンピュータ協会 1979 年.

[16] Donald E. Knuth, The art of computer programing, vol 1 - vol 3 Addison-Wesley (1 巻と 2 巻は日本語訳有).

[17] K. イェンゼン・N. ヴィルト著 原田賢一訳, Pascal 第 4 版 培風館.

[18] A. W. エイホ, B. W. カーニハン, P. J. ワインバーガー著足立高徳訳,
プログラミング言語, AWK アジソン ウェスレイ, トッパン 1988 年.

[19] L. Wall, R. L. Schwartz 著 近藤嘉雪訳, Perl プログラミング, ソフトバンク 1993 年.

8.2 プログラミング言語の種類

von Neumann 型のコンピュータが生みだされて以来約 40 年たちますが, その間に様々なプログラミング言語が開発されました.

まず, プログラミング言語はプログラムを処理する形態で, 次の二つに大きく分類されます.

Interpreter(通訳系と訳す事がある)
: 処理を記述したものから, 一部分のまとまりのある処理内容を機械語に変換しては実行し, それが終わると次の部分に移る, ということを繰り返して, 処理をします. つまり, 同時通訳の人が外国語から日本語に通訳するように, プログラミング言語の内容をその場で機械語に通訳して, 計算機に実行させているのです. awk は interpreter です.

Compiler(翻訳系と訳す事がある)
: 処理の記述を一度に全部機械語に翻訳してしまい, その後にそれを実行します. コンピュータは, 機械語に翻訳された内容を読みこみ, それを実行します. この授業では, compiler を取り上げます.

また, プログラミング言語は, その文法が違えば異なる言語です. 文法のなかで, 良く知られているものを挙げると次のようなものがあります. 注意して欲しいのは, 同じ文法を持つ言語でも compiler だったり interpreter だったりすることがあることです.

assembly 言語 プログラミング言語を作るに当って, 最も単純な発想はコンピュータに対する命令である 2 進法の並びを, 一対一で人間が理解できる言葉に対応させることです. この対応で一つの言語が作れます. それが, assembly 言語です. assembly 言語の処理系のことを assembler といいます. この言語を作るのは簡単ですが, 機械語で出来ることがあまりにも単純な事しかできないため, この言語を使うプログラミングは人間にとってはやさしいことではありません. 例えば, 画面に 1 を表示させるのも, assembly 言語を使うと結構大変な仕事になります. また, 機械語と一対一に対応しているということから, コンピュータが違えば, assembly 言語も違うということに注意してください. すなわち, assembly 言語という名前の全てのコンピュータに共通の言語があるわけではありません.

assembly 言語のようにコンピュータに密着した言語を, 低級言語といいます. それに対し, より人間に近い言語を高級言語といいます. 高級言語と呼ばれるものの特徴は, コンピュータが違っても (原理上は) まったく同じ記述でプログラミングが出来るということです.

Fortran Formula transformation の略です. 最も早い時期に作られた, 高級言語です. 主に科学技術用数値計算をするためのプログラミング言語です. 古くからあるために, 数値処理のためのパッケージとかが豊富で, 現在でも用いられています. ただ, プログラミング理論とかが無い時代の産物なので, 最近の理論から見ると良いプログラミング言語とはいえないようです. 実際問題として, 特に Fortran 用のパッケージを使う必要性がないのなら, もう勉強する必要もないと思います

COBOL Fortran と同時期に作られた高級言語ですが, こちらはもっぱら事務処理用言語として開発されました. 時々マスコミを賑わす 2000 年問題の多くは, COBOL を用いて作られたプログラムで発生します.

ALGOL Fortran, COBOL と同時期に開発された言語です. こちらの方は, 数学者が中心となって作ったといわれるだけあって理論的にすっきりした言語だそうです. ただ, あまり普及しませんでした. その理由はよくわかりませんが, 想像するに, あまりにも言語仕様が巨大になったことと, 標準での入出力の定義が, 与えられなかった事だろうと思います.

Pascal スイスの N. Wirth という人が, 教育用目的に ALGOL を元に言語仕様を小さくし, さらに標準の入出力を定義して開発しました. 1980 年代には結構流行し, 現在でも比較的使われています. 現時点でも, プログラミング入門用言語としては, 良いものだと思います. ただ, 元の言語仕様では大きなプログラムを書くにはちょっと不便です. なお, パスカルから派生した言語として modula-2 と Ada があります. またその孫ともいえる, Oberon という処理系もあります. Oberon は, IBM-PC 互換機とか Macintosh では無料で処理系が ETH(何の略かは忘れました) から入手できます.

LISP List Processor の略です. プログラム言語としては, かなりの歴史を持っています. やたら括弧をつかってプログラムをするのが, 特徴です. これと親戚の言語として, Scheme というのもあります. 人工知能や Expert System の分野では, 良く用いられます. 例えば数学では, 数式処理系 (数式のまま微積分をしたり, 行列計算をしたりする処理系) のひとつ Reduce は, LISP をつかって開発されています. 前期の授業で取り上げた emacs の拡張機能も, LISP の親戚である Emacs LISP で作られます. 以前は方言の多い言語でしたが, 現在では, Common LISP と言う規格が作られました. ただ, Common LISP の言語仕様もかなり大きなものになっています. UNIX 用では, 京都大学で開発された KCL (Kyoto Common LISP) が, 有名です.

Prolog 日本のどこかの省庁の次世代コンピュータの開発言語として採用されました. 三段論法をそのままプログラムしていくような言語です.

C 1970 年代に UNIX のシステム記述用言語として開発されました. OS の開発にはそれまでは, assembly 言語を用いていましたから, そこに高級言語を用いたということで, これは画期的なことでした. しかし, そもそもは assembly 言語の代わりとして作られたという経緯があるため, 比較的低級な言語になっています.

SmallTalk 1970 年代に, 新しいプログラミング言語として Xerox の研究所で開発されました. オブジェクト指向という新しいプログラミング規範 (paradigm) を導入した言語です. もともとは, 言語だけでなく OS も一体となった環境でしたが, 残念ながらその環境はあまり世に受け入れられませんでした. (値段の問題だと思います) OS の環境のほうは, 良く似た考え方が Macintosh に採用され最近の主流となっています. オブジェクト指向プログラミングの考え方も, C, Pascal, Oberon LISP などに導入されて, Objective-C, C++, CLOS 等の言語となっています. 当時の Xerox では, 現在の主流となっている技術, 例えばワークステーションの概念とか, EtherNet によるネットワークがすでに考えられており, Xerox なしでは現在のコンピュータはありえないのですが, 当の Xerox がそこから余り利益を得ていないのは, 歴史の皮肉です.

BASIC アメリカで, プログラミング入門用として開発されました. (開発者たちの名前は忘れました) 多くのパソコンに入っていたため, 一躍有名になりました. 本来の BASIC の言語仕様は, 割ときちんとしているのですが, パソコンに移植されたものが言語仕様の一部分だったため, 「良い」言語としては, あまり普及していません. (本来の言語仕様を True BASIC というそうです) 手軽ではありますが, パソコンに付いてくるのを使ってプログラミングの入門をするのは, 余りいいことだとは思いません. なお, 新指導要領では高校の数学で, この BASIC を使用するそうです.

Perl 最近少し流行している処理系です. これは, compiler ではなく interpreter です. だいたい $awk + shell + C$ の感じです. つまり, shell のように command を実行させることが出来, awk のように手軽な記述が可能で, C のように何でもできる言語です. (C の中で, command 実行はしない事はないが, 処理系に依存するので大変. shell は command 実行以外はほとんど出来ない. awk も文書処理以外の処理は, 余り得意でない.) UNIX のシステム管理者を目指すなら, 勉強して損は無い言語です.

Java WWW で閲覧するページにいろんな機能を付け加えるために開発された言語です. C 言語に似た構文とオブジェクト指向な環境が備わっています. 最近の WWW の流行にあわせて, これも流行しているようです.