1 プログラムミングの基礎知識

1.1 プログラムとプログラム言語

プログラム (program) の語源は、

pro(before)+gram(write)

で、あらかじめ書いておくという意味である。コンサートのプログラムというような使われ方をする。この講義でプログラムとは**コンピュータにまとまった仕事をさせるための命令を順に並べたもの**である。

プログラム、コード、ファイル

プログラム コンピュータにまとまった仕事をさせるための命令を順に並べたもの。プログラムに用いる人工言語 (artificial language) をプログラム言語 (programming languege) という。
プログラミング (programming) とはプログラム言語を用いてプログラムを作成すること。

コード プログラムをプログラム言語で符号化したもの。

ファイル プログラムを補助記憶装置に保存したもの。

プログラム言語は、機械語 (machine language)、アセンブリ言語 (assembly language)、高級 (高水準) 言語 (high-level language) の 3 つに分けられる。

機械語 (machine language) コンピュータ (CPU) が理解できるように、0 と 1 の並び (ビット列という) で記述される。

アセンブリ言語 (assembly language) 機械語命令と 1:1 に対応する英文字列 (ニーモニック mnemonic) により記述される

高級言語 (high-level language) 自然言語 (たいていは英語) と数字と記号により記述される。人間の思考に適した概念と構造を持つ。

例 1.1 キーボードから 2 つの数を入力し、その和をディスプレに出力するプログラム

- 機械語プログラム (16 進表示)

```
0000000 457f 464c 0101 0001 0000 0000 0000 0000
0000020 0002 0003 0001 0000 82f4 0804 0034 0000
0000040 07ec 0000 0000 0000 0034 0020 0007 0028

中略

0011340 6572 6e69 7469 615f 7272 7961 735f 6174
0011360 7472 5f00 675f 6f6d 5f6e 7473 7261 5f74
0011400 005f
```

- アセンブリ言語によるプログラム

```
    .file "add.c"
    .section .rodata
.LC0:
    .string "%d"
.LC1:
    .string "a+b=%d\n"
    .text
.globl main
    .type main, @function
main:
    pushl %ebp
    movl %esp, %ebp
    subl $24,%esp

中略

    call printf
    addl $16, %esp
    movl $0, %eax
    leave
    ret
    .size main, .-main
    .section .note.GNU-stack,"",@progbits
    .ident "GCC: (GNU) 3.3.6 release (Vine Linux 3.3.6-0vl7)"
```

- 高級言語 (C 言語) によるプログラム

```
#include <stdio.h>

int main(void)
{
  int a,b,sum;

  scanf("%d",&a);
  scanf("%d",&b);

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

  return 0;
}
```

参考　C 言語によるプログラムのファイル名を add.c とする。上記の機械語プログラムとアセンブリ言語プログラムは C コンパイラ gcc により作成した。

```
$ gcc -S add.c <-- アセンブリ言語プログラム作成
$ ls
add.c add.s
$ cat add.s  <-- アセンブリ言語プログラム表示
$ gcc add.c  <-- 機械語プログラム (実行可能ファイル) 作成
$ ls
a.out add.c add.s
$ od -x a.out  <-- 機械語プログラム表示
```

1.2　高級言語

人間の思考に適した概念と構造を持つプログラム言語を高級言語という。高級言語には、コンパイラ言語とインタプリタ言語がある。

コンパイラ言語 コンパイラ (compiler) と呼ばれる翻訳プログラムにより、高級言語で表現されたプログラムを中間言語、アセンブリ言語、または機械語で表現されたプログラムに翻訳する。このことをコンパイルする (compile) という。コンパイルする前のプログラムをソースコード (source code)、コンパイルした後のプログラムをオブジェクトコード (object code) という。
例 C、FORTRAN

インタプリタ言語 インタプリタ (interpreter) とよばれるプログラムを使い、高級言語で書かれたプログラムを 1 命令ずつ解釈しながら実行する言語である。
例 BASIC、LISP、Perl

注 1.1 高級言語は必ずしもコンパイラ言語とインタープリタ言語に二分できない。Java はソースファイル (拡張子 java) をコンパイラ javac でバイトコード (拡張子 class) に変換し、インタープリタ java で実行する。バイトコードは仮想マシン (JVM) の機械語と考えられる。

例 1.2 キーボードから 2 つの数を入力し、その和をディスプレに出力するプログラム (例 1.1 と同じ) を C 言語以外のいくつかの高級言語により示す。

- FORTRAN

```
      INTEGER A,B,SUM
      READ(5,500) A,B
500   FORMAT(2I5)
      SUM=A+B
      WRITE(6,600) SUM
600   FORMAT('A+B=',I5)
      STOP
      END
```

- Pascal

```
program add(input, output);
var a,b,sum:  integer;
begin
    read(a);
    read(b);
    sum := a+b;
    writeln(sum)
end.
```

- BASIC

```
10 INPUT A
20 INPUT B
30 LET SUM=A+B
40 PRINT "A+B=";SUM
50 END
```

- Java

```
import java.io.*;
class Add {
  public static void main(String[] args)
      throws java.io.IOException {
    BufferedReader x =
      new BufferedReader(new InputStreamReader(System.in), 1);
    String sa = x.readLine();
    int a = Integer.parseInt(sa);
    String sb = x.readLine();
    int b = Integer.parseInt(sb);
    int sum = a+b;
    System.out.println("a+b=" + sum);
    }
}
```

2　C 言語の基本

2.1　なぜ C 言語か

プログラム言語 C(以下では C 言語あるいは単に C と呼ぶ) は、構造化言語でありフリーのコンパイラが利用できるので、Java と並び多くの大学でプログラミング教育に利用されている。

C 言語は UNIX の開発の過程で作られたため、システムの記述に向いている。そのため、現在でも C 言語は業務用開発やフリーソフトウェア開発、ファームウェア (Firmware：ハードウェアに組み込まれたハードウェアを制御するプログラム) などで、幅広く利用されている。また、携帯ゲーム機のゲームソフトも C で記述されている。ただし、Web 系のソフトは Java が使われることが多い。

基本情報技術者試験 (情報処理推進機構 `http://www.jitec.jp/`) のプログラム言語の試験は、C(JIS X3010)、COBOL(JIS X3002)、Java(JLS3.0)、アセンブラ (CASL II) の 4 つの言語から選択することになっている。

オブジェクト指向の言語 C++と Java は C を参考に作られているため、C を習得しておくと C++と Java の習得は容易になる。

2.2　C 言語の歴史 — 名前の由来

1960	ALGOL60	欧米の計算機科学者のコミュニティーが、アルゴリズムの研究開発に用いる目的で、世界共通のプログラミング言語として開発した。ALGOL (ALGOrithmic Language)
1963	CPL	Cambridge 大学と London 大学が共同で作る。ALGOL60 の影響 (Combined Programming Language)
1967	BCPL	リチャード Martin Richard が CPL を簡素化し、システムプログラムを目指す (Basic CPL)
1970	B	トンプソン Ken Thompson が DEC PDP-7 上の最初の UNIX システム用に開発したインタープリタ言語。(BCPL の B)
1972	C	リッチー Dennis Ritchie が PDP-11 の UNIX 上で開発したコンパイラ言語。(next to B)
1978	K&R C	B. W. Kernighan and D. M. Ritchie, The C Programming Language, Prentice Hall 出版
1982	C++	ストラウストラップ B. Stroustrup が C 言語を拡張。
1989	ANSI C	ANSI(American National Standards Institute) が規格化。規格番号 ANSI X3.159-1989
1990	C90	ANSI X3.159 と同内容で章番号のみが異なる。規格番号 ISO/IEC 9899:1990(E) Programming Language-C
1992	ANSI C	章番号を ISO/IEC 9899:1990 に合わせる。現在の米国規格。規格番号 ANSI/ISO 9899:1990[1992]
1993	JIS C	ISO 9899:1990 の日本語訳。規格番号 JIS X 3010-1993
1999	C99	ISO C の 改訂。ISO/IEC 9899:1999(E) Programming Language-C (Second Edition)
2003	JIS C(C99)	ISO 9899:1999 の日本語訳。規格番号 JIS X 3010-2003 日本工業標準調査会 `http://www.jisc.go.jp/` のトップページにある 「JIS 検索」をクリックし、「JIS 規格番号から JIS を検索」の検索欄に X3010 と入力すると閲覧できる。

本講義で扱う C 言語は JIS C(C99) に基づく。講義で用いる gcc の C コンパイラ cc は C99 をサポートしている。

2.3 プログラム作成から実行まで

東京女子大の Mac OS X を使ったプログラム作成から実行までをのべる。

2.3.1 エディタの起動

Mac OS X で標準で利用される日本語コードは Unicode(UTF-8) であるが、利用できる C コンパイラ gcc は UTF-8 ではなく EUC に対応している。したがって、プログラミングに使えるエディタとしては、EUC で保存でき、文字飾り等が入らないものがいいので、Emacs, Jedit の２つのうち気に入った方を使えばよい。

Emacs を利用する場合は、初めての起動のときは、

[Finder] ⇒ アプリケーション ⇒ TWCU

に Emacs のアイコンがあるので、Dock にドラッグアンドドロップする。二度目以降の起動のときはドックにおいてある Emacs のアイコンをダブルクリックする。

Jedit X を利用する場合は、Jedit X のアイコンをドックにドラッグアンドドロップし、C のプログラミングに適したように設定しておくとよい。

2.3.2 ソースファイルの作成

エディタで C 言語の文法に基づくソースプログラムを作り、ファイルとして保存する。ファイル名には拡張子.c をつける。

- 英数字、記号はすべて半角を用いる。
- 全角の空白を用いるとエラーになる。(エラー発見は難しい。)
- 大文字と小文字は区別される。
- ;(セミコロン) と:(コロン)、1(数字) と l(小文字のエル)、0(ゼロ) と o(小文字のオー) 等に注意する。

2.3.3 ターミナルの起動

コンパイルには、東京女子大学では gcc(GNU Compiler Collection [1]) の c コンパイラ gcc(ソースプログラムを実行可能プログラムに変換するプログラム) を用いる。C 言語のコンパイルや実行はターミナルから行う。C コンパイラ gcc は日本語コードが EUC-JP なので、EUC に対応したターミナルを利用する。

アップルが提供している「ターミナル」のデフォルトの文字コードは UTF-8 であるので、東京女子大学では日本語文字コードを EUC に変更した EUC-JP.term を用意している。EUC-JP.term は Dock に入れられないため毎回 Finder から起動する必要が出てくるので、ここでは、「ターミナル」の文字コードを EUC に変更する方法をのべる。

[Finder] ⇒ アプリケーション ⇒ ユーティリティ

とするとターミナルのアイコンが表示されるので、ターミナルのアイコンを Dock にドラッグアンドドロップする。

メニューバーの、[ターミナル] → [ウインドウ設定...]、ターミナルインスペクタのポップアップメニュー (「シェル」と表示してある) をディスプレに変更する。

文字セットエンコーディングを「日本語 (EUC)」に変更し、[設定をデフォルトとして使用]、ターミナルインスペクタを閉じる。一旦ターミナルを終了し、再起動すると文字コードは EUC になる。

[1] 以前は、GNU C Compiler の略とされていた。現在は、C, C++, Objective-C, Fortran, Java, and Ada, の言語のコンパイラやライブラリを含んでいる。
http://gcc.gnu.org/

2.3.4 コンパイル

コンパイルとは人間が読めるソースプログラムをコンピュータが実行可能な機械語プログラムに翻訳することをいう。

1. ターミナルを起動する。
2. コンパイルする。

```
$ cc ソースプログラム名
```

このとき、ソースプログラムにエラーがなければ、実行可能ファイル a.out が作られる。

3. 実行する。

```
$ ./a.out
```

演習 2.1 1. ホームディレクトリに comp2a(~/comp2a) というディレクトリを作れ。

```
$ mkdir comp2a
$ cd comp2a
```

2. ディレクトリ~/comp2a に hello.c (Hello World プログラム) を作れ。

```
/* hello.c */
#include <stdio.h>

int main(void)
{
  printf("Hello World\n");

  return 0;
}
```

3. hello.c をコンパイルせよ。

```
$ cc hello.c
```

4. どのようなファイルが作成されたか。

```
$ ls -l
```

5. 実行せよ。

```
$ ./a.out
```

演習 2.2

```
ようこそ、C の世界へ!
```

と出力するプログラム youkoso.c を作りコンパイルし実行してみよ。

注 2.1 東女のシステムでは、二重引用符"で囲まれた部分では日本語 (全角文字) を使うことができる。日本語の文字コードは EUC-JP である。

2.4 Emacs の使い方

2.4.1 Emacs の画面

Emacs の画面は、6 つの部分から成り立つ。Mac Os X ではメニューバーは他のアプリケーションと同様デスクトップの最上部にある。メニューバー (左端はアップルマーク、右端は Spotlight) にはモードに応じたメニューが並ぶ。残りの 5 つは Emacs のフレーム[2]の中にある。

タイトルバー Emacs@端末名左端の ⊗ は「閉じるボタン」、左から 2 番目の ⊖ は「しまうボタン」(アイコン化ボタン)、左から 3 番目の ⊕ は「ズームボタン」である。(Mac OS X のほとんどのアプリケーションに共通)

ツールーバー 各種のアイコンが並んでいる。

テキストウィンドウ バッファの内容を表示

モードライン ウィンドウに表示されているバッファの状態を表示する。灰色に網かけされており、文字モード、バッファの変更状態、バッファ名、モード名、表示位置等を表わす。

エコー領域 Emacs からのメッセージやミニバッファが表示される。yes/no やファイル名等の入力に用いる。

テキストウィンドウとモードラインを合わせウィンドウという。

2.4.2 起動直後の 3 行のメッセージ

Emacs を起動し何か入力しようとすると、

```
;; This buffer is for notes you don't want to save, and for Lisp evaluation.
;; If you want to create a file, visit that file with C-x C-f,
;; then enter the text in that file's own buffer.
```

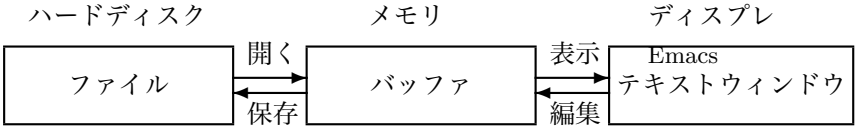
が表示される。ファイルを読み込むと消えるが、新規作成のときは、この 3 行をマウスで選択しハサミのアイコンで削除する。

2.4.3 Emacs の仕組み

文書はファイルとしてハードディスクに保存される。Emacs はファイルを直接編集するのではなく、ある程度編集した後ファイルに書き込む。(メモリ上の) 編集する場所をバッファという。バッファは Emacs の画面 (テキストウィンドウという) に表示されるので、バッファの編集はテキストウィンドウを見ながら行う。新規にファイルを作成する場合、Emacs の画面を見ながらバッファを編集する。バッファに [名前をつけて] 保存すると、[その名前の] ファイルがハードディスクに保存される。**バッファを保存するまでは、編集結果はファイルに反映されない。**モードラインの左から 4 つ目あたりに**となっているときは、編集されたバッファはまだ保存されていない。保存すると--となる。なお、%%となっているときは、書き込み禁止バッファである。

Emacs は同時に複数のバッファを編集することができる。編集するバッファはメニューバーの [Buffers] で選択できる。

[2]Mac OS X の他のアプリケーションの場合はウィンドウというが、Emacs ではウィンドウは別の意味で使うのでフレームと呼ぶ。



2.4.4 基本操作

Emacs の操作方法は 3 通りある。

入力	操作
キーボード	control と esc
マウス	メニューバーから選択
	ツールバーのアイコンをクリック

キーボードによる操作は、タッチタイピングにより操作可能なので入力が速くなる。さらに、プログラミングを行うときには便利である。キーボードによる操作は Tutorial を実行するとよい。[Help] → [Emacs tutorial] で起動できる。

演習 2.3 Tutorial を実行せよ。
(Tutorial の画面が壊れても次回に Tutorial を立ち上げれば元のものが起動される。)

2.4.5 記法

表示	操作
control-x	control を押しながら x を押す
esc,x	esc を押してすぐ離し、x を押す
[ファイル]	メニューの [ファイル] を左クリック
Read an existing file	アイコン Read an existing file をクリック

2.4.6 終了

キーボード	control-x control-c
マウス	[File] → [Exit Emacs]
	タイトルバー左端の×ボタンをダブルクリック

2.4.7 コマンドの中断/取り消し

中断	control-g	ベルをならして中断
取り消し	Undo	
	control-x u	

取り消しは必ずできるとは限らない。

困ったとき

- 操作を元に戻すには、Undo アイコンをクリックする。
- トラブルに出会ったら、control-g (control キーを押しながら g を押す)

2.4.8 カーソルの移動

移動	矢印キー	コントロールキー
前の文字	←	control-b (backward)
後の文字	→	control-f (forward)
前の行	↑	control-p (previous)
後の行	↓	control-n (next)
前のページ		esc, v
次のページ		control-v
バッファの先頭		esc, <
バッファの最後		esc, >

2.4.9 文字および行の削除

機能	特殊キー	コントロールキー
カーソルの位置の文字を削除		control-d
カーソルの位置の前の文字を削除	delete	
カーソルの位置から行末までを削除		control-k
バッファを削除		control-x k → (ミニバッファで) return → yes return

control-k で削除した文字列は control-y で貼付けることができる。

2.4.10 複写、削除、移動

マウス左ボタンを押したまま移動することをドラッグという。ドラッグした部分は背景が灰色になる。

複写	複写する部分をドラッグ → 複写先で中央ボタンクリック
	複写する部分をドラッグ → 複写先で control-y
	複写する部分をドラッグ → [Edit] → [Copy] → 複写先をクリック → [Edit] → [Paste]
	複写する部分をドラッグ → Copy → 複写先をクリック → Paste
削除	削除する部分をドラッグ → control-w
	削除する部分をドラッグ → [Edit] → [Cut]
	削除する部分をドラッグ → Cut
移動	移動する部分をドラッグ → control-w → 移動先で中央ボタンクリック
	移動する部分をドラッグ → [Edit] → [Cut] → 移動先をクリック → [Edit] → [Paste]
	移動する部分をドラッグ → Cut → 移動先をクリック → Paste

2.4.11 ファイル操作

ファイル	操作	バッファ
開く	control-x control-f → ミニバッファからファイル名を入力	ファイルをバッファに読み込む
	[File] → [Open File...] → 表示されるファイルのリスト (Choose a file) から入力するファイルを選ぶ	
	Read an existing file → 表示されるファイルのリスト (Choose a file) から入力するファイルを選ぶ	
	Finder で開きたいファイルのアイコンを Emacs のアイコンにドラッグ&ドロップ	
挿入	control-x i → ファイル名はミニバッファから入力	カーソルのある位置に指定ファイルを挿入
	[File] → [Insert File...] → 表示されるファイルのリスト (Choose a file) から入力するファイルを選ぶ	
新規保存	control-x control-w → ミニバッファからファイル名を入力	現在のバッファを新しいファイルに保存
	[File] → [Save as...] → ファイル名入力ウィンドウ (Enter name) からディレクトリを選びファイル名を入力	
上書き保存	control-x control-s	現在のバッファをファイルに保存 保存されたファイル名がエコー領域に表示される。ふるいファイルはファイル名の最後に~がついたものに変る
	[File] → [Save (current buffer)]	

2.4.12 Emacs にコピーアンドペースト

ターミナルでの実行結果を Emacs にコピーアンドペーストするには、

1. コピーする範囲をマウスでドラッグ (背景が灰色になる)。
2. [編集][コピー]
3. Emacs の貼付ける場所をマウスでクリックしマウスのホイール (中ボタン) を押す。
4. プログラムに実行結果を貼付けたときは、/* と */ で挟み注釈にする。

2.4.13 日本語の入ったファイルの印刷

いったん実行結果を貼付けたプログラムを保存 ([File][Save]) し、[File][Print Buffer] とする。
文字化けするときは、ターミナルから

```
$ a2ps ファイル名 | lpr
```

とする。a2ps (Any to PostScript filter) は任意のファイルをポストスクリプトファイルに変換するフィルタ、| はパイプ (| の左側の結果を右のコマンドに渡す)、lpr はプリンタに出力するコマンドである。

2.5 Hello World プログラム

“Hello World” を出力するプログラムは、K&R が冒頭で取り上げたため最も有名な C によるプログラムである。このプログラムを C99(ANSI C も同じ) により表わすと、

```
1  /* hello.c */
2  #include <stdio.h>
3
4  int main(void)
5  {
6    printf("Hello World\n");
7
8    return 0;
9  }
```

となる。(左端の行番号は説明のためにつけたもので、プログラムは枠の中のみ。)
このプログラムは、すべての C プログラムに共通する多くの特徴を持っているので、1 行づつ説明する。

1. 1 行目の /*と*/で挟まれた部分は**注釈** (comment) である。/*と*/の間はコンパイラは無視して処理する。注釈はプログラムを読む人のために書く。Hello World プログラムのソースファイル名を `hello.c` としたことをコメントしている。

2. 2 行目は**前処理指令** (preprocessing directive) で、
   `#include <stdio.h>`
   は、`stdio.h` (**st**an**d**ard **i**nput/**o**utput) というファイル (ヘッダファイルという) を取り込んでいる。`<stdio.h>`を**ヘッダ** (header) という。`<stdio.h>`は `printf()` 関数[3]に関する情報を含んでいる。

3. 3 行目の空行は、プログラムを読みやすくするために、前処理指令と `main()` 関数の間に付けたもの。空行は無くてもよい。

4. 4 行目
   `int main(void)`
   は、`main()` 関数を定義している。(p.12 を見よ。) `int` は `main()` 関数の返却値が `int` 型であること、`void` は `main()` 関数が仮引数を持たないことを表わしている。`int` と `void` は C のキーワード (Keyword) であり、オブジェクトや関数などの識別子 (名前) に使うことはできない。

5. 5-9 行目
   ```
   {
     printf("Hello World\n");

     return 0;
   }
   ```
   は、`main()` 関数の**本体** (body) である。

6. 中括弧 (波括弧)(`{`と`}`) とそれに囲まれた部分を**複合文** (compound statement) という。

7. 6 行目の
   `"Hello World\n"`
   は、文字列リテラル (string literal) である。**文字列リテラル**とは二重引用符`"`で囲まれた文字の並びである。大文字’H’、小文字’e’、...、小文字’d’ と**改行文字**’`\n`’ からなる。

8. 6 行目は、ライブラリ関数 `printf()` の呼び出しを行う式文である。

[3]語源は print format である。転送文字数を返す。出力エラーが発生したときは負の値を返す。

9. `printf("Hello World\n");`
   は、文字列リテラル`"Hello World\n"`を画面に出力する。すなわち、Hello World を画面に出力し、`\n` により改行 (new line) している。

10. 8 行目は、`return` 文である。return 文は分岐文である。
    ```
    return 0;
    ```
    は、`main()` 関数の実行を終了し、値 0 を実行環境に返すことにより、プログラムが正常終了したことを実行環境に知らせる。`return` は C のキーワードである。

11. 式文、分岐文はセミコロン; で終わる。

注 2.2 日本語 (全角文字) は注釈 (`/*`と`*/`で挟まれた部分または`\\`と改行文字で挟まれた部分) と文字列リテラル (二重引用符`"`で囲まれた文字の並び) でのみ使うことができる。それら以外で用いるとエラーになる。

C 言語の規則 (その 1)

1. `printf()` 関数を呼び出す場合、前処理指令で`<stdio.h>`ヘッダを取り込む必要がある。
2. 式文、分岐文はセミコロン ; で終わる。

`main()` 関数

1. `main()` 関数はプログラム開始処理において呼び出される。
2. `main()` 関数は仮引数を持たない関数
   ```
   int main(void)
   ```
   または、2 つの仮引数を持つ関数
   ```
   int main(int argc, char *argv[])
   ```
   として定義する。(および、これらと等価な方法。)
3. `main()` 関数の本体は、{ から } までである。

2.6 Windows パソコンで gcc を使うには

コンピュータ IIA で利用される gcc は、Mac OS X では、X11 と Xcode Tools をインストールすることにより、使える。

Windows PC では、gcc をデフォルトでは使えないので、gcc を含む UNIX の様々なフリーソフトウエアを Windows で利用できるようにした Cygwin(シグウィン) を Windows にインストールすればよい。

2.6.1 Cygwin のインストール

インストールに先立ち 2 つのことを行うとよい。

1. コンピュータ名が全角の場合、コンピュータ名を半角に変更する。(Windows XP の場合)
   (a) [コントロールパネル]
   (b) パフォーマンスとメンテナンス
   (c) コンピュータの基本的な情報を表示
   (d) [コンピュータ名]

(e) コンピュータ名に全角文字が使われている場合 [変更]

(f) コンピュータ名に半角英数で入力

(g) 再起動

2. ユーザ名に全角を用いている場合は、新規にユーザ名が半角のアカウントを作成し、半角のアカウントでログオンする。
(たとえば、ユーザ名が”長田”の場合、新たに”osada”を作り、cygwin をインストールするときや使うときは、osada でログオンする)

インストールには 2 通りの方法

- インストール用の CD-ROM が付録についた本を利用する。
`http://www.amazon.co.jp` で調べたところ、2007 年 5 月現在 11 点ある。
- インターネットがらダウンロードする。

がある。

本講義ではインターネットからダウンロードする方法を述べる。

`http://www.cygwin.com/`

に行き、トップページで [Install or update now!] をクリックして setup.exe をダウンロードする。

詳細は

`http://www.twcu.ac.jp/~osada/comp2a/cygwin.html`

を見よ。

2.6.2 Cygwin の使い方

1. メモ帳などのエディタでソースプログラムをつくる。ソースファイル名には拡張子.c[4]をつける。(ソースファイルはホームディレクトリ `C:/Cygwin/home/`ユーザ名 またはその下にサブディレクトリをつくり保存する)

2. コンパイルする。

```
$ cc hello.c
```

このとき、ソース・プログラムに綴りの間違えなどのエラーがなければ、実行可能ファイル `a.exe` が作られる。

3. 実行する

```
$ ./a.exe
```

日本語を含むファイルを読むには、cat コマンドを使う。

```
$ cat hello.c
```

Cygwin のプログラムや出力結果をコピーするには、Cygwin の窓の左上の Cygwin のアイコンをクリックするとプルダウンメニューが表示されるので、[編集][範囲指定] を順に選んでから、マウスでコピーしたい部分をドラックし、Cygwin のアイコンをクリックし、[編集][コピー] を選ぶ。

[4] メモ帳のバージョンによっては拡張子に `.txt` がついてしまう。このときは、ファイル名を ”hello.c”のように二重引用符” で囲む。

2.7　2 数の和を求めるプログラム

例 1.2(p.3) で取り上げた、2 つの整数をキーボードから入力しその和を出力するプログラムを少し使いやすくした addition.c を考える。

```
1  /*****************************************
2  * addition.c                              *
3  * 2 つの 10 進整数を読み込みその和を表示する    *
4  *****************************************/
5  #include <stdio.h>
6
7  int main(void)
8  {
9    int a,b,sum;
10
11   printf("2 数 a,b の和を求めるプログラム \n");
12   printf("a=");
13   scanf("%d",&a);
14   printf("b=");
15   scanf("%d",&b);
16   sum = a + b;
17   printf("a + b =%d\n",sum);
18
19   return 0;
20 }
```

hello.c プログラムにないプログラミングの概念について説明する。

このプログラムでは、オブジェクト (object) が使われ、入力 (input)、計算 (computation)、出力 (output) が行われる。入力にはライブラリ関数 scanf()[5]が用いられる。scanf() 関数と printf() 関数により入出力を行う場合は、変換指定子 (conversion specifier) が必要である。計算には、加算演算子と代入演算子が用いられている。

オブジェクト[6] とは、コンピュータのメモリ上の名前付きの場所で、プログラムを実行する際に値がその場所に格納される。オブジェクトの型によって、場所の大きさとメモリに格納される際の内部表現が異なる。オブジェクトの名前を**識別子** (identifier) と言い、その値を参照するときに用いる。オブジェクトを用いるときは、オブジェクトの型と識別子の**宣言** (declare) が必要である。同じ型の複数のオブジェクトを宣言するときは、カンマ”,” で並べる。宣言はセミコロン”;” で終わる。

1. 1-4 行目
   注釈である。注釈は複数行にまたがってもよい。プログラムの先頭にファイル名やプログラムのドキュメントを書いておくと保守が容易になる。

2. 9 行目
   ```
   int a,b,sum;
   ```
   は 3 つの int 型オブジェクト a,b,sum を用いることを宣言している。

3. プログラムの 11-15 行目で入力を行っている。11 行目では、ディスプレに、
   2 数 a,b の和を求めるプログラム
   と出力し改行している。12 行目では、ディスプレに、
   a=
   と表示し、a の入力を求めている。13 行目の
   ```
   scanf("%d",&a);
   ```
   は、scanf() 関数の呼び出しで、キーボードから a の値を入力しリターンキーを押すとオブジェクト a にその値が格納される。%d は、10 進整数 (decimal integer) として入力することを指定する**変換指**

[5]語源は scan format である。入力項目の個数を返す。変換が 1 つも行われないまま入力エラーが発生したときはマクロ EOF を返す。
[6]多くのテキストでは**変数** (variable) と呼ばれる。

定子 (conversion specifier) である。`&a` はオブジェクト `a` に格納することを表わしている。`scanf()` 関数では`&` が必要である。14-15 行目は 12-13 行目と同様である。

4. 16 行目の`+`は、加算演算子と呼ばれ、`+`の前にある値 (第 1 オペランド) と後ろにある値 (第 2 オペランド) の和を結果とする。

5. 16 行目の右辺
   ```
   a + b
   ```
   は式 (expression) といわれる。

6. 16 行目
   ```
   sum = a + b;
   ```
   は `a` の値と `b` の値を加えた値を `sum` の値とするという代入文 (assignment statement) である。代入文は、代入演算子`=`の右辺の値を左辺のオブジェクトに代入する。

7. 17 行目はライブラリ関数 `printf()` の呼び出しである。
   ```
   printf("a + b = %d\n",sum);
   ```
   は引数として文字列リテラルとオブジェクト `sum` の 2 つを持っている。第 1 引数の文字列リテラルの中にある%d は、第 2 引数 `sum` を 10 進整数で出力する変換指定子である。文字列リテラルを出力するときは第 2 引数のオブジェクトを変換指定子に従い出力する。ディスプレに a + b = と出力し、その右に `sum` の値を 10 進整数により表示する。

実行結果は次のようになる。

```
% ./a.out
2 数 a,b の和を求めるプログラム
a=3
b=4
a + b = 7
```

statement	a	b	sum
`int a,b,sum;`			
`scanf("%d",&a);`	3		
`scanf("%d",&b);`	3	4	
`sum = a + b;`	3	4	7

addition.c を実行したときのオブジェクトの内容の変化

C 言語の規則 (その 2)

1. オブジェクトとは、コンピュータのメモリ上の名前付きの場所である。オブジェクトの型によって、場所の大きさとメモリの内部表現が異なる。
2. オブジェクトは宣言してから使用する。宣言には型と名前を用いる。
3. 代入文は、代入演算子=の右辺の値を左辺のオブジェクトに代入する。
4. 加算演算子+は、+の第 1 オペランドと第 2 オペランドの和を結果とする。
5. 演算子とオペランドの列を式という。
6. `scanf()` 関数や `printf()` 関数の第 1 引数の文字列リテラルに表れるパーセント記号%で始まる記号の列を変換指定子という。
7. `scanf()` 関数の第 1 引数の文字列リテラルに表れる%d は、10 進整数として入力することを指定する。
8. `printf()` 関数の第 1 引数の文字列リテラルに表れる%d は、10 進整数として出力することを表わす変換指定子である。
9. `scanf()` 関数では入力するオブジェクトの前に&を付ける。

演習 2.4 `addition.c` の 11 行目から 15 行目までを削除し、

```
a = 3;
b = 4;
```

で置き換える (ファイル名 `add1.c`) と出力結果はどうなるか。

演習 2.5 `addition.c` の 17 行目 `printf("a + b =%d\n",sum);` を `printf("%d+%d=%d\n",a,b,sum);` と変更する (ファイル名 `add2.c`) と出力結果はどうなるか。

演習 2.6 2 数 a、b の差を求めるプログラム `subtraction.c` を作れ。

2.8 プログラミングスタイル

簡潔で読みやすいプログラミングスタイルを取るとプログラムの保守が良くなる。特に他人がプログラムを利用する場合や、自分が長期に渡って利用する場合には必須である。

C では、空白、空白行 (空行)、改行、注釈は字句 (キーワード、オブジェクトや関数の識別子、演算子など) の途中以外のどこに挿入してもよいので自由にプログラムを書くことができる。以下のようなプログラミングスタイルを勧める。

1. 注釈をつける。
2. オブジェクトや関数の識別子に意味を持たせる。
3. 段落に分ける。
4. 階層構造になっているときは、段下げ (indent) を用いる。
   Emacs でソースを記述する場合は、行毎に TAB キーを押すと段下げを行ってくれる。
5. 複雑な式は括弧”()”により優先順位を明示する。

空白や中括弧の次のような使い方はプログラマの好みである。

1. 算術式は、

```
sum=a+b;
```

としても、

```
sum = a + b;
```

と演算子とオペランドの間に空白を空けてもよい。

2. 開き中括弧"{"は、

```
for(i=0; i<N; i++){
  sum += a;
  sum2 += a*a;
}
```

としても、

```
for(i=0; i<N; i++)
{
  sum += a;
  sum2 += a * a;
}
```

としてもよい。+= は一つの演算子なので+と=の間に空白を入れるとエラーになる。

プログラムの中で字下げを行うことをインデント (indent) という。インデントは Tab キーまたはスペースキーで行う。

C 言語の規則 (その 3)

1. 改行は文の区切りではない。1 行に複数の文を含むこともできる。
2. コンパイラは、空白やタブや改行や注釈 (空白類という) を 1 つの空白と見なす。
3. 字句 (キーワード、関数名など) の途中を除いて空白類を入れることができる。
4. 空白を入れられるところには、注釈を入れることができる。

2.9 キーワード

C99 では下記に示す 37 のキーワードを定めており、このキーワードはオブジェクトや関数の識別子に使用できない。(既出の 3 つのキーワードは太字で表わしてある)

ANSI C に含まれるキーワード

auto	break	case	char	const	continue
default	do	double	else	enum	extern
float	for	goto	if	**int**	long
register	**return**	short	signed	sizeof	static
struct	swich	typedef	union	unsigned	**void**
volatile	while				

C99 で追加されたキーワード

inline	restrict	_Bool	_Complex	_Imaginary

プログラミング言語 C は、キーワードと構文の組み合わせで形成されている。キーワードはすべて小文字で書く。