# プログラミング演習Ｉ　第２回

**C言語入門**
**今日の目標：変数宣言と数値入力**

## 1. C言語入門（教科書１章、２章）

(1)　特徴
（i）　標準的なプログラム言語
・　あらゆる場面で活躍：組み込みプログラム（携帯､家電､…)、ゲーム､ビジネスソフトなど

・　IC の設計、ハードの設計などもＣ言語をベースとした開発言語
　　UNIX などの OS もＣ言語

（ii）　メモリや入出力のかなり細かい操作ができる
　　→逆にメモリ管理は自分で行う必要→バグ（プログラムミス）多し！

（iii）　関数の集合体として記述
　　プログラムの実行は一連の関数の実行として実現

（iv）　豊富な演算、データ型

(2)　プログラム記述時の注意

（i）　基本的に小文字で書く。大文字を使う時は慣習がある。
　　慣習を守らないと他人が見にくいソースになる。

（ii）　基本的に一行に一文

（iii）　インデント（字下げ）を使う→徐々に慣れて行こう

‘{‘と’}’の間（ブロックという）を１段階（タブを使うとよい）下げる

例：インデントなし
```
int main(void)
{
int i,j,k;
for (i=1;i<100;i++)
{
j=i++;
...
```

インデントあり
```
int main(void)
{
        int i,j,k;
        for (i=1;i<100;i++)
        {
                j = i++;
```

```
        }
                ...
```

http://spandaudiolab.yz.yamagata-u.ac.jp/kkondo/programming1/memo/1.html

を参考にせよ。

（iv）　コメントをたくさん付ける→自分でも忘れる
/*　と　*/　　で囲むとコメントになる

文法の解説ではなく，処理の内容（概要）を述べよ。
http://www.spandaudiolab.yz.yamagata-u.ac.jp/~kkondo/programming1/memo/2.html
を参考にせよ。

（v）　　コメント内、文字列内以外は全て半角英数字で記述
全角文字､特に全角スペースがコメントと文字列以外に入るとエラーになるので注意。

（vi）　その他、見やすいソースのコツはたくさん。少しずつ慣れよう。

(3)　　ソースファイル（プログラムを記述したテキストファイル）を保存したら､コンパイルして実行ファイル（実行できるファイル）に変換する。（教科書 p.8）

**$ gcc -Wall -o 実行ファイル名 ソースファイル名**

例：gcc -Wall -o printName.exe printName.c
gcc: Ubuntu で使う標準的な C 言語コンパイラ
  -Wall: 全ての警告､注意を出力→より正しい文法
  -o 実行ファイル名：実行ファイル名指定（ファイル名.exe としましょう）

ソースファイル名：ファイル名.c としましょう
ファイル名には特殊文字(/.,~スペースなど)、日本語（可能だが､トラブルの元)、内容を表さないもの（単に a.c や program.c など、何のプログラムか分らない）は避けよう。

(4)　　コンパイルした実行ファイルを実行してみる。（教科書 p.9）

(3)で作成した実行ファイルをコンソールで実行してみる。実行ファイルがあるディレクトリも指定しなくてはならない。現在のディレクトリ（カレントディレクトリ）に実行ファイルがある場合は、これを”./”を先頭につけて明示する。

例：

```
$./printName.exe[Enter]
```

./(ピリオドとスラッシュ)はカレントディレクトリ（現在作業中のディレクトリ）にあるファイルであることを明示している。

## 2. C 言語ソース（プログラム）の構造

(i) ヘッダファイル
プログラムの先頭にはヘッダファイルを指定する。（教科書 p.16）
#include <stdio.h>
stdio.h がファイル名。標準の入出力で必要な宣言が入っている。今はおまじないと思うこと。st**u**dio.h ではないことに注意！（ミススペルする人多数）
(興味ある人は/usr/include/stdio.hがこのファイルなので、中身を less などを使って覗いてみよ。)

(ii) main 関数
プログラムには必ず main 関数がある。(教科書 p. 17)
関数は関数名や引数を宣言する文の後、{ と }（カーリー・ブレース）で囲まれる。
int main(void)の int は戻り値の型(整数型)、void は引数（関数に入力されるデータ）がないことを示す。
戻り値の型を宣言しているので、main の最後に戻り値が必要。
例：return 0;
この例では値 0 を整数で返している。

(iii) 文の終端(教科書 p. 17)
文は;(セミコロン)で終わる。複数行に渡っても良い。
例：
```
printf("This is an example of a long line %d, %d, %d\n",
        i, j, k);
```

(iv) コメント(教科書 p. 25)
コメントは/* と */で囲む。複数行に渡っても良い。
例：
```
/********************
*                     *
*********************/
```

(v) データの宣言文(計算などをせずに型宣言などのみ行い、実行がない分)、C の実行文(実際に計算やメモリ操作などを実行する文)
今週から少しずつ解説していく。

(vi) 関数(教科書 p. 16)
C 言語は基本的に実行する内容を一まとめにして関数を呼び出す形で実行していく。関数は自分で定義してもよいし、あらかじめ用意したもの（標準関数と呼ぶ）もある。先週のプログラムで使用した printf は標準関数、 main は自分で定義した関数になる。

今週は変数の宣言と基本演算子を使ってみる。

## 3. 変数の宣言、型

C では変数は使う前に必ず名前とデータの型を先頭で宣言する。宣言しないとコンパイルエラーとなる。
（先週のプログラムは宣言していないように見えるが、ヘッダファイルの中で宣言している）

例：
```
int main()
{
        int i, j;              →整数型(int)で i と j を宣言
        i=1;
```

```
        j=i + 1;
        return 0;
}
```

(i)　代表的変数の型（教科書 p.37）

| 変数型 | 型名 | 大きさ（バイト幅） | 範囲 |
|---|---|---|---|
| 文字 | char | 1 | -128～127(-$2^7$～$2^7$-1) |
| 整数 | int | 4 | -$2^{31}$～$2^{31}$-1 |
| 単精度実数 | float | 4 | 約 $10^{-37}$～$10^{38}$、有効 6 桁 |
| 倍精度実数 | double | 8 | 約 $10^{-307}$～$10^{308}$、有効 15 桁 |

1 バイト（Byte）は 8 ビット(bit)、2 進数 8 桁を表す。上記以外の型は教科書 p51 参照。文字型は文字を扱うデータに限らず、単に 1 バイトの数を扱うデータ型として考えよ。

(ii)　定数（教科書 p.28～33)）
各データ型の定まった値。

(a) 整数型の例
```
int i;
i = 413;        (整数型の定数 413 を i に代入)
```
参考：上記は 10 進数の定数だが、8 進、16 進も扱える（教科書 p.39）

(b) 文字型の例
```
char c;
c = 22;         (文字型の定数 22 を c に代入)
c = 'A';        (文字 A を表す文字コードの値が入る)
```

(c) 実数型の例
```
float f;
f = 123.456;            (実数型、小数点式定数を f に代入)
f = 12.3456e+1;         (同上、指数形式の定数 12.3456×10+1)
```

(d) 文字定数と文字列

・文字定数：**一文字**を表す文字コード（アスキーコード)。'（シングルクオート）で囲む

例：　'A'　→A を表すアスキーコード（65）

・文字列（文字列リテラル）：**複数の文字**を並べて、最後に\0(1 バイトの 0)を付与したもの。"（二重クオート）で囲む。

例："Hello"　→　'H', 'e', 'l', 'l', 'o', '\0'を並べた定数

**文字定数と文字列で混乱**しないように。文字列は後で詳しく扱う。

## 4. printf を使った出力（教科書 p.24、112）

printf の引数に文字列を与えると、表示できる。
例：先週扱った printf("Hello\n");
なお、'¥'は'\'と等価なので、どちらを使っても良い。
第一引数に書式指定を含めると、変数の値も表示できる。
例：printf("A = %d, B = %d\n", i, j);
10 進数で変数 i と j が
A = 1, B = 2
といった形式で表示される。

代表的な書式指定（教科書 p.33）
%d: 整数型 10 進数
%f: 実数型小数点形式
%e: 実数型指数形式
%s: 文字列
参考：表示する桁数も指定できる
例： %5d → 5 桁整数型表示
%6.2f → 実数型、全体で 6 桁、小数点以下 2 桁表示
%6s → 文字列を右詰 6 文字表示

## 5. scanf を使った数値入力（教科書 p.106）

scanf(*書式指定*、&*変数名*);
変数はあらかじめ宣言してあること。
変数名の前に必ず **&** を付ける→今はおまじないとして覚える。
使用例：

```
int i;
scanf("%d", &i);
```

注意：書式指定に改行を入れないこと！例 scanf("%d~~\n~~", &i);
scanf の書式指定に桁数指定を含めないこと！例 scanf("%~~6.2~~f", &x);

## 6. 基本算術演算子（教科書 p.56）

(1) *:乗算、/:除算、%:剰余（余り）
(2) +:加算、-:減算
(1)の演算が(2)の演算に優先されるので、注意。予想外の結果になることあり。
計算順番を明示したい場合は () 括弧を付ける。

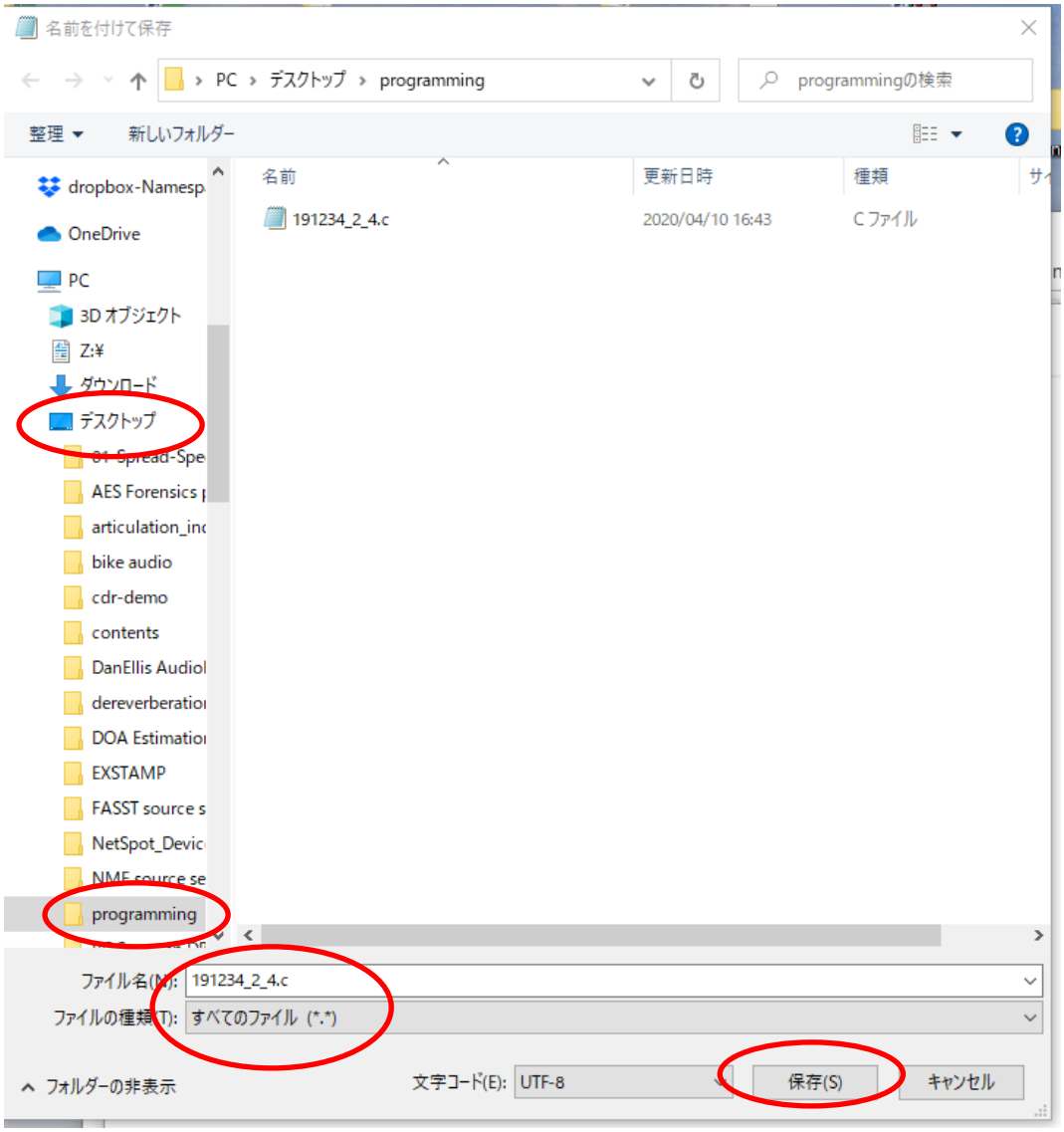
例：本当は $\frac{a+b}{c}$ を計算したい

```
int a, b, c, d;
a = 2; b = 4; c = 2;
d = a + b / c;        (d = 4)
d = (a + b) / c;      (d = 3)        どっちが正しい？
```

## 7. 補足説明：WSL/Ubuntu や Cygwin を用いたプログラムの入力からコンパイルまで

プログラムの入力からコンパイルまでを簡単に説明する。

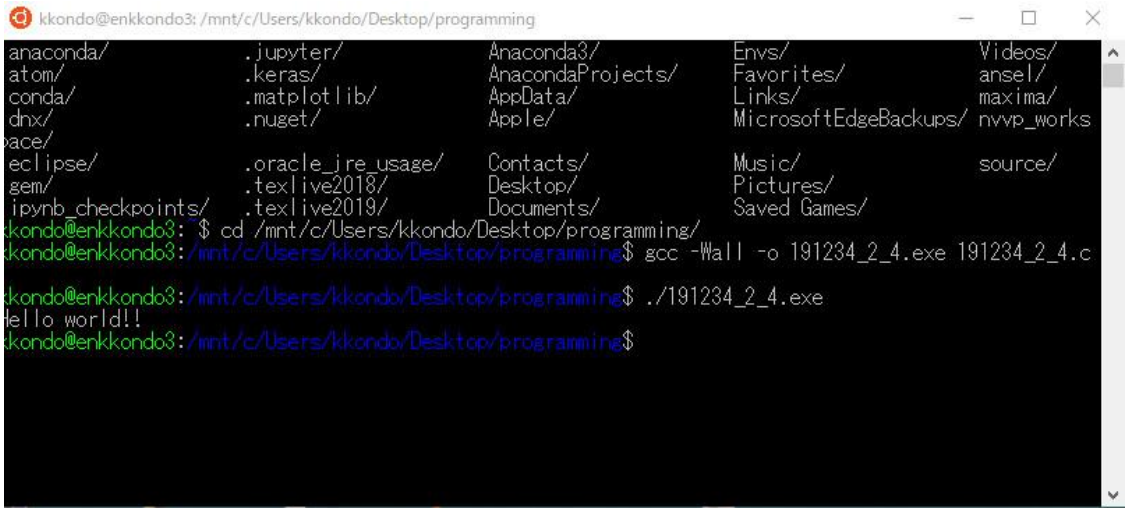
**Windows Subsystem for Linux (WSL)/Ubuntu** を使っている場合の説明：
デスクトップに programming という名前のディレクトリを作って、ここにプログラムファイルを作るとします。
スタートメニュー→Windows アクセサリ→メモ帳
でメモ帳を開いて、プログラムを入力します。
（他のエディタを使いたい人はそちらを使ってください）
入力が終わったらメモ帳の
ファイル→名前を付けて保存
を選んで、
デスクトップ→programming（ダブルクリック）




ファイル名:
の箱に例えば
191234_2_4.c
と入力し、ファイルの種類は
すべてのファイル(*.*)
を選んで、
保存(S)
のボタンを押す。以上でプログラムのソースコードが入ったファイルが保存される。

次にこのファイルをコンパイルする。
スタートメニュー→Ubuntu
を選んで Ubuntu のウィンドウを呼び出し、コンソールに
cd /mnt/c/Users/（Ubuntu のユーザ名）/Desktop/programming
と先ほど保存したディレクトリに移動し、
gcc -Wall -o 191234_2_4.exe 191234_2_4.c
と入力すると 191234_2_4.c をコンパイルした実行ファイル

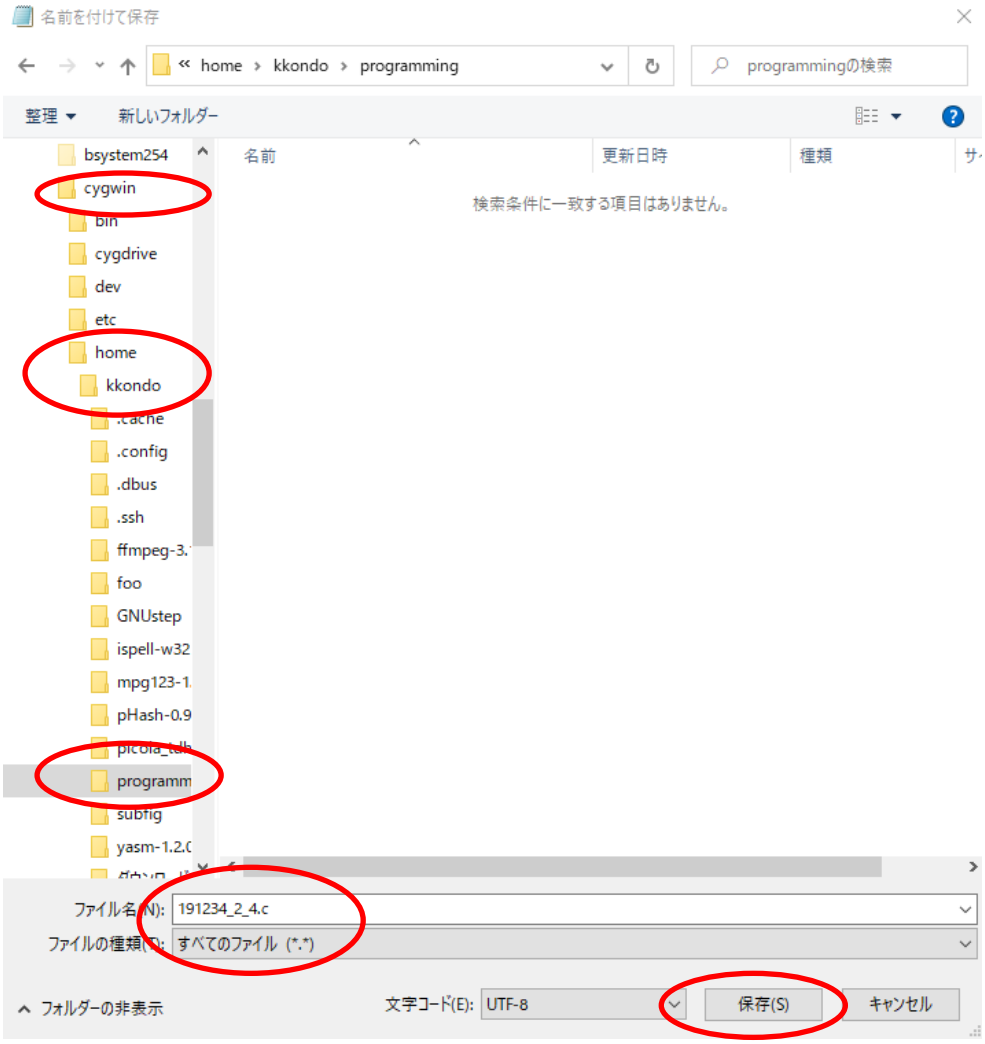
191234_2_4.exe
ができているはずである。
もしここでエラーが出たら、再びメモ帳でエラーを訂正して、再度コンパイルする。
エラーがでなくなったら、Ubuntu のウィンドウ内で
./191234_2_4.exe
と入力するとこのファイルが実行できる。



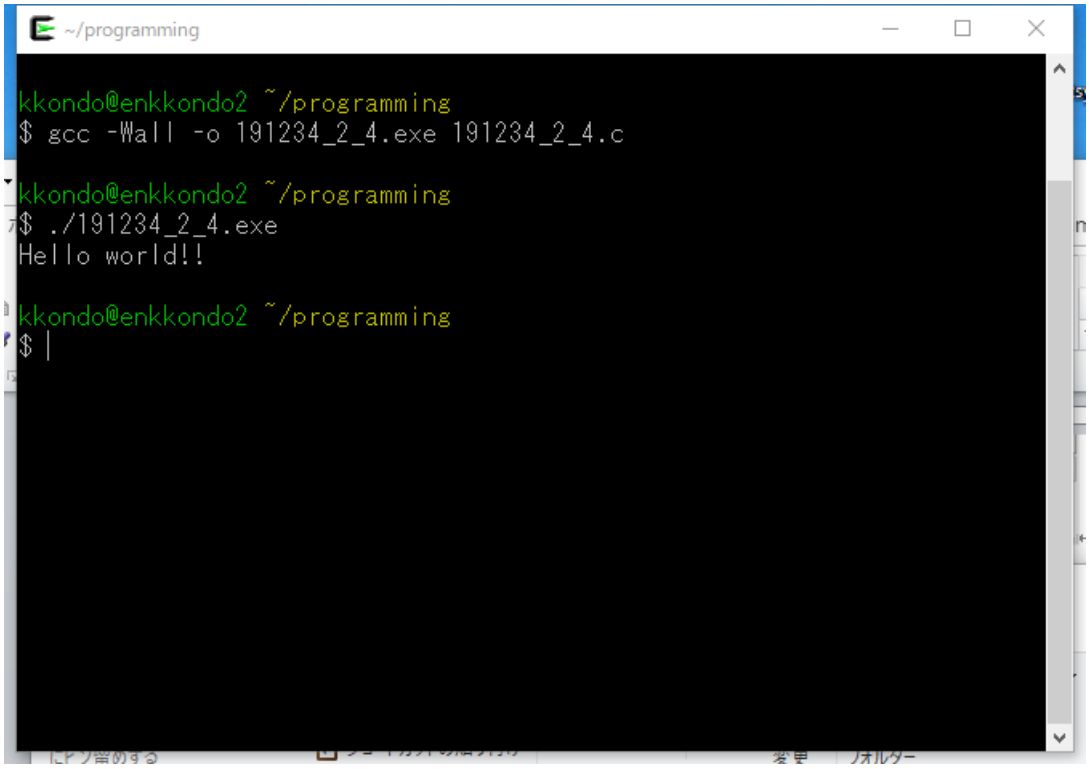

**Cygwin の場合：**

スタートメニュー→Cygwin→Cygwin Terminal（あるいは Cygwin64 Terminal）を選んで Cygwin の端末起動。この時点でユーザのホームディレクトリに設定されている。
例えばユーザ名が taro であれば
/home/taro
ここにプログラムを入れておくディレクトリを作る。例えばこれを programming とすると
mkdir programming
このディレクトリに移動
cd programming
このディレクトリにプログラムを入れていく。
スタートメニュー→Windows アクセサリ→メモ帳
でメモ帳を開いて、プログラムを入力する。
（他のエディタを使いたい人はそちらを使ってください）
入力が終わったらメモ帳の
ファイル→名前を付けて保存
を選んで、
(Cygwin をインストールしたディレクトリ )/home/(cygiwn ユーザ名)/programming
例えば C:cygwin にインストールしていれば
C:cygwin/home/taro/programming
とダブルクリックしていき
ファイル名:
の箱に例えば
191234_2_4.c
と入力し、ファイルの種類は
すべてのファイル(*.*)
を選んで、

保存(S)
のボタンを押す。以上でプログラムのソースコードが入ったファイルが保存される。




次にこのファイルをコンパイルする。
Cygwin Terminal 内にに
cd programming
と先ほど保存したディレクトリに移動し、
gcc -Wall -o 191234_2_4.exe 191234_2_4.c
と入力すると 191234_2_4.c をコンパイルした実行ファイル 191234_2_4.exe
もしここでエラーが出たら、再びメモ帳でエラーを訂正して、ファイルを保存し、再度コンパイルする。
エラーがでなくなったら、Cygwin Terminal のウィンドウ内で
./191234_2_4.exe
と入力するとこのファイルが実行できる。




## 8. 本日の講義の副教材

本日の講義の内容をほぼカバーする動画が下記にあるので、見てみよう。

https://paiza.jp/works/c/primer/beginner-c1

## 9. 本日の演習

(1) 以下のリストを入力、コンパイルし、動作させよ。
次に"Hello World!\"の World の部分を自分の名前（日本語）にしてコンパイル、実行してみよ。

```
#include <stdio.h>
/* 教科書 p.10 */
int main(void)
{ /* メイン・プログラム */
    printf("Hello World!\n");
    return 0;
}
```

(2) 以下のリストを入力、コンパイルし、動作させよ。

```
#include <stdio.h>
/* 数値を入力し、2 倍にして出力 */
int main(void)
{
    int dt;
```

```
    printf("Enter number:");
    scanf("%d", &dt); /* dt に数値入力 */
    dt = dt * 2;              /* dt を 2 倍 */
    printf("Result: %d\n", dt); /* 結果出力 */
    return 0;
}
```

(3) (1)を改良して、3 回整数値を入力し、1 回の入力毎に積算値を出力プログラムに改良せよ。
動作例：(イタリックはキーボードより入力したもの、他はプログラム出力とする)

```
$ lab3_2.exe[Enter]
Enter number:1[Enter]
Sum: 1
Enter number:2[Enter]
Sum: 3
Enter number:3[Enter]
Sum: 6
$
```

(4) (2)を改良して、実数値を入出力するようし、その積算値と 2 乗積算値を表示せよ。表示は**全体が 5 桁、小数点以下 3 桁**で表示せよ。
動作例：(イタリックはキーボードより入力したもの、他はプログラム出力とする)

```
$ lab3_3.exe[Enter]
Enter number:1.1[Enter]
Sum: 1.100, Squared sum: 1.210
Enter number:2.2[Enter]
Sum: 3.300, Squared sum: 6.050
Enter number:3.3[Enter]
Sum: 6.600, Squared sum: 16.940
$
```

(5) 発展（余裕がある人はやってみてください）
半径を実数として入力し、その円周と面積、また球の体積と表面積を計算して表示するプログラムを作成せよ。計算は実数で行い、円周、面積は全体で 7 桁、小数点以下 2 桁として表示すること。円周率はプログラムの先頭に
`#include <math.h>`
を付け加えれば、プログラムの中で M_PI を$\pi$の値として参照できる。
（例えば a = 2 * M_PI;）
動作例：(イタリックはキーボードより入力したもの、他はプログラム出力とする)

```
$ lab3_4.exe[Enter]
Enter radius:1.2[Enter]
Circumference =    7.54
Area =    4.52
Volume =   7.24
Surface area =   18.10
$
```

**演習（4）まで、あるいは余裕があれば（5）まで完成させるように。**

**今回は演習（4）のプログラムをレポートとします。**
プログラムファイルを WebClass を使って提出してください。ファイル名は
**(学籍番号)_(演習の回)_(課題番号).c**
と名前を付けて提出してください。例えば学籍番号 191234 の人が第２回の課題(4)のプログラムを提出する場合は
19123_2_4.c
と名前を付けて **WebClass で提出**してください。

**提出期限は 5 月 14 日の 16：00** です。期限外は受け付けられなくなりますので、注意。動作しないプログラムは減点されるので注意。

なお、まだ自分の PC でコンパイルできる環境ができていない人は何とか提出期限までに構築を試みてください。とりあえず情報基盤センターの PC を用いて Cygwin を使ってレポートに取り組んでください。詳しくは WebClass にアップロードされている資料
「演習に使える PC が確保できない場合、設定ができない場合、ネットワーク接続できない場合」を参考にしてください。ただし今後 COVID19 感染者が出た場合、構内に立ち入れなくなるので、自分の PC を確保するようにしてください。

ノート PC を購入できる人はこれをきっかけに早急に購入してください。(今後必ず必要になります)。

**8. 次回の予習：** 教科書（明快入門Ｃ）の第 5 章を読んでおくこと。