

プログラミングの簡単な手引き

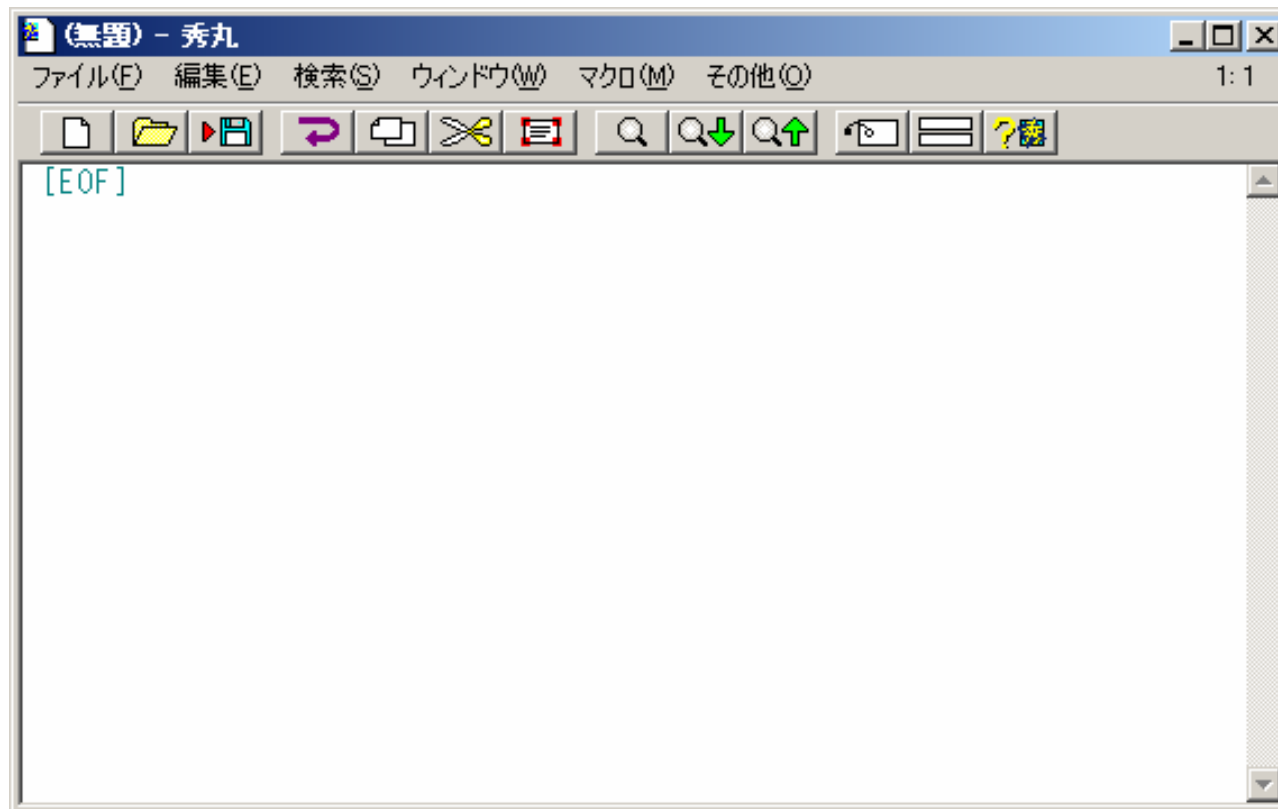
2008/09/25

v1.02

プログラムを書くための手引書

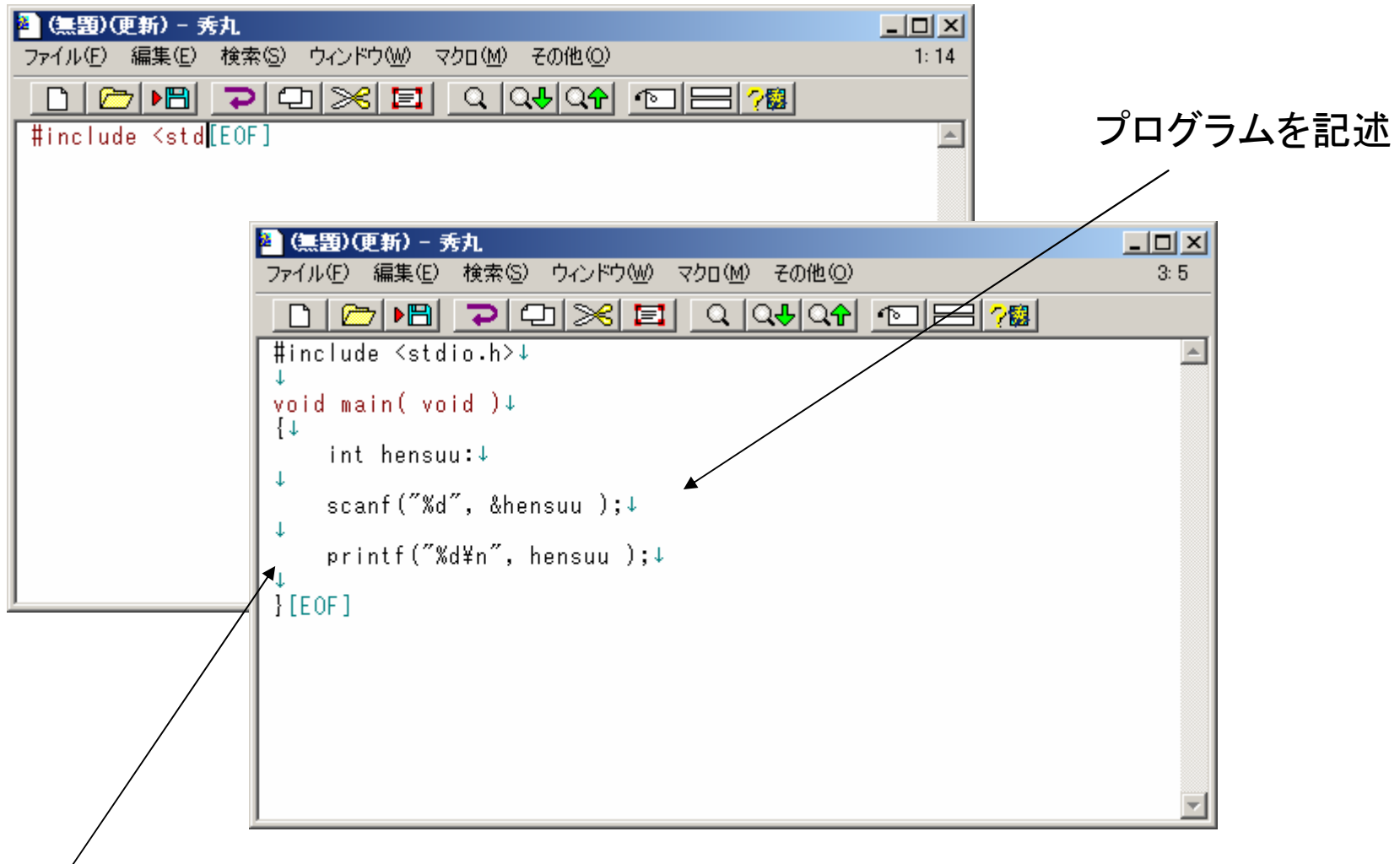
1. エディタの起動

秀丸エディタの起動



起動すると
左の図のプ
ログラムが
表示される

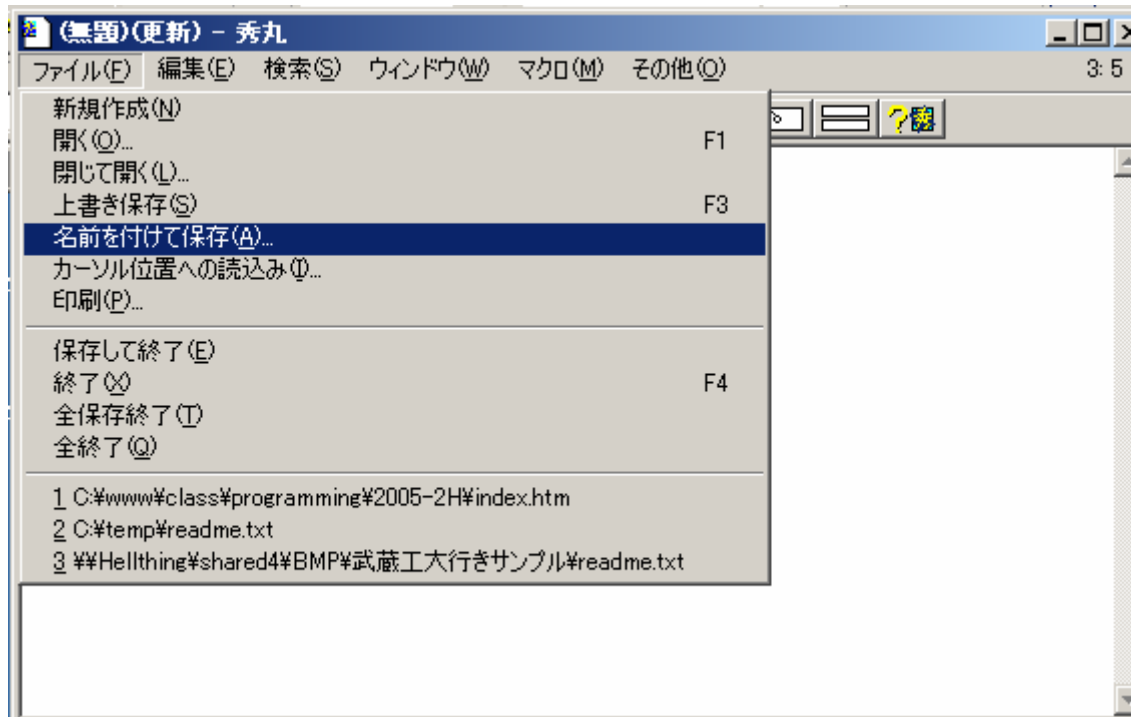
2. プログラムの記述



プログラムを見やすくするためインデント(空白)を入れる。
空白は半角スペース4個程度が良く用いられる。

3. プログラムの保存

- まだ一回も保存していない場合→【名前を付けて保存】を選択



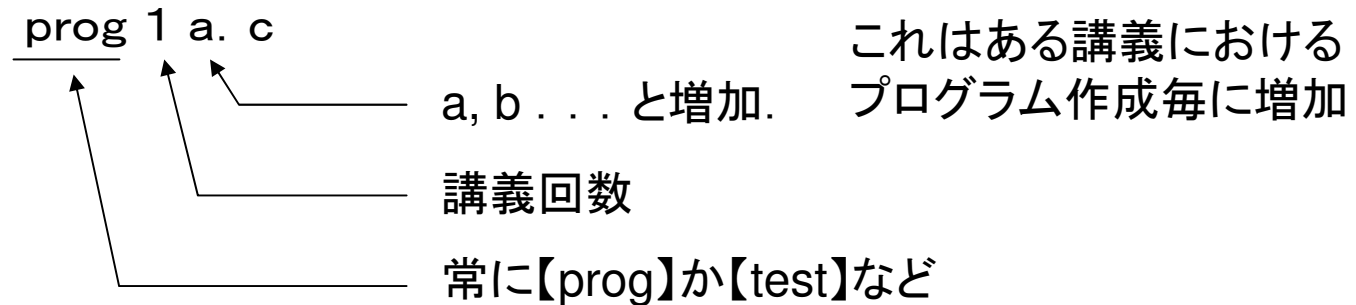
ファイルはZドライブに保存

保存するフォルダは任意で良いが、
自分で分かりやすいフォルダ名に保存すること
例えば【programming】など



ファイル名は ファイル名 .c のように、拡張子には .c をつけること

ファイル名は任意でよいが、英数字だけを使うこと。
または以下のようなルールにしておくとう分かりやすい。



* 拡張子 .c をファイル名に付け忘れると拡張子.txtで保存されるので注意

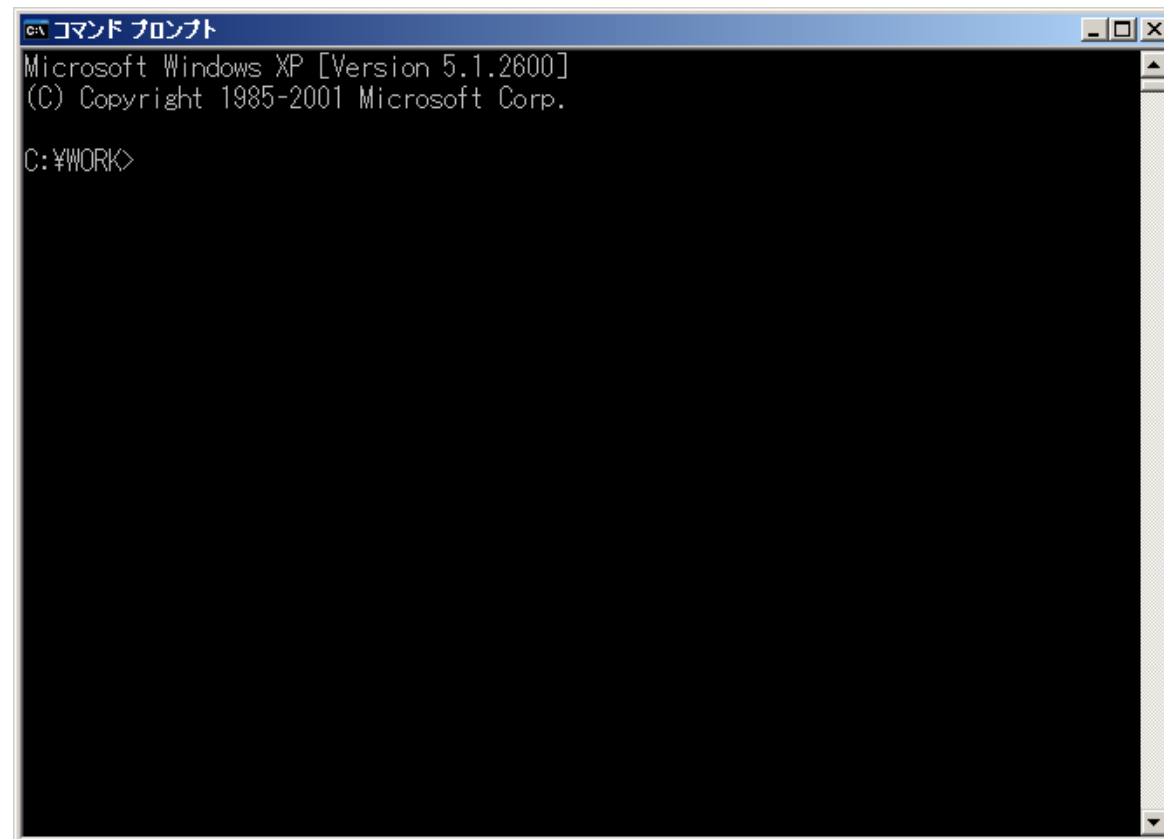
4. コマンドプロンプトの起動

保存したC言語のソースファイルをコンパイルする。
そのコンパイル作業のためにコマンドプロンプトを起動する。



コマンド プロンプト

コマンドプロンプト
を起動すると、右
のような画面が表
示される。

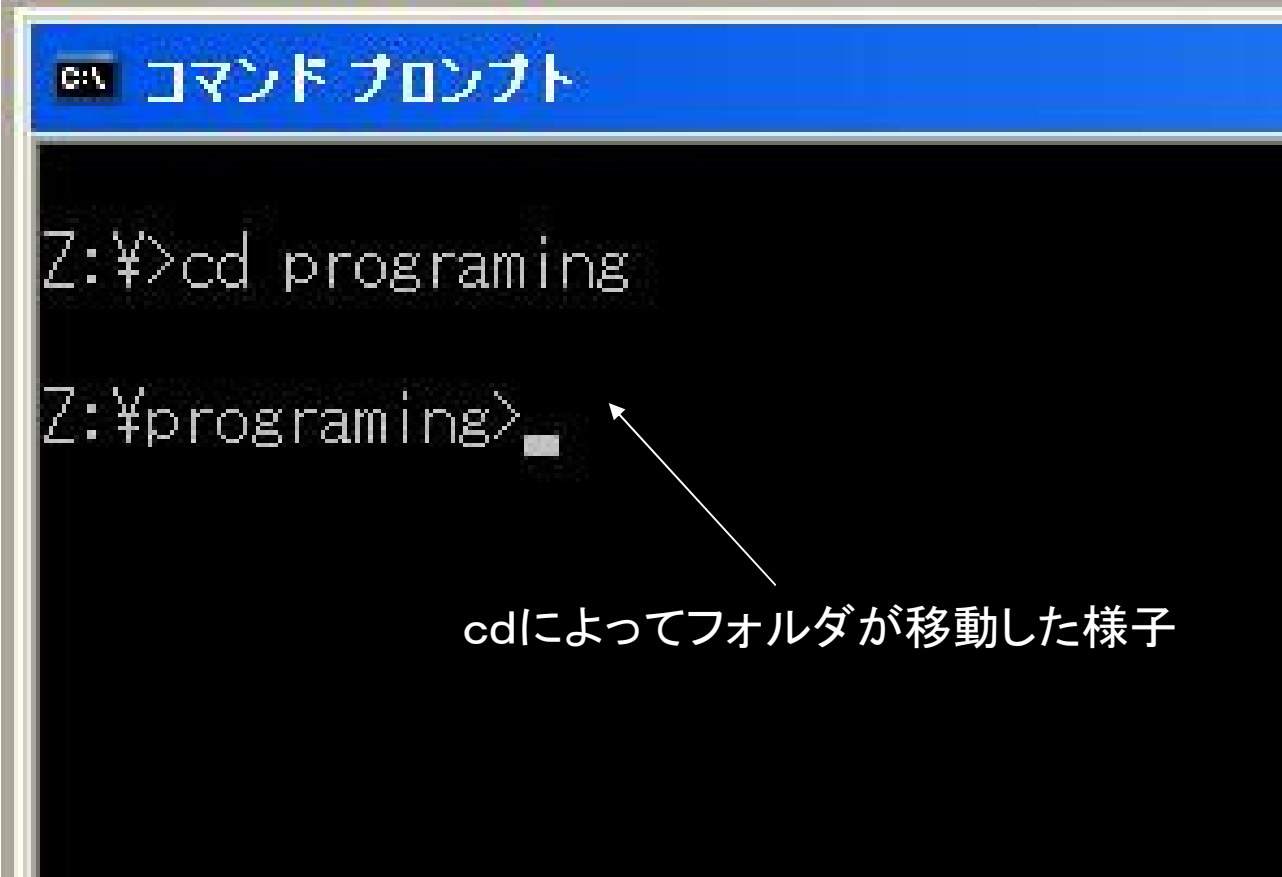


5. ファイルを保存したフォルダへ移動

ファイルを保存したフォルダへ移動させる.

Z:> cd 【保存したフォルダ名】

【cd】というコマンドを使ってフォルダを移動する



```
コマンド プロンプト

Z:¥>cd programing

Z:¥programing>
```

cdによってフォルダが移動した様子

6. フォルダ内のファイルの確認

ファイルを保存したフォルダの中を表示させて、3. で保存したC言語のファイルがあることを確認する

Z:> dir

【dir】というコマンドを使ってフォルダの中を表示させる

```

C:\WORK\programming>dir
ドライブ C のボリューム ラベルは ドライブC です
ボリューム シリアル番号は 80E8-C534 です

C:\WORK\programming のディレクトリ

2005/10/15  03:49    <DIR>          .
2005/10/15  03:49    <DIR>          ..
2005/10/15  03:49                117 prog3a.c
               1 個のファイル                117 バイト
               2 個のディレクトリ  22,519,500,800 バイトの空き領域

C:\WORK\programming>
    
```

dirによってフォルダの中を表示させた様子
c言語のソースファイルがあることを確認

7. プログラムのコンパイル作業

プログラムのコンパイルは【cl】コマンドを使う

z:> cl 【ファイル名. c】

↑ ファイル名に【. c】を付けること

```

c:\ コマンド プロンプト
Z:\programming>cl prog3a.c
Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 12.00.8168 for 80x86
Copyright (C) Microsoft Corp 1984-1998. All rights reserved.

prog3a.c
Microsoft (R) Incremental Linker Version 6.00.8168
Copyright (C) Microsoft Corp 1992-1998. All rights reserved.

/out:prog3a.exe
prog3a.obj
Z:\programming>
    
```

cl コマンドによってファイルをコンパイルしている様子
これは正常に終了している様子

エラーの場合には 2. まで戻り, プログラムを修正

コンパイルの結果がエラー場合

エラーの内容を確認 → プログラムを修正(次ページ参照)

```

C:\> コマンド プロンプト

Z:\programming>cl prog3a.c
Microsoft (R) 32-bit C/C++ Optimizing Compiler Version 12.00.8168 for 80x86
Copyright (C) Microsoft Corp 1984-1998. All rights reserved.

prog3a.c
prog3a.c(18) : error C2018: 文字 '0x81' は認識できません。
prog3a.c(18) : error C2018: 文字 '0x40' は認識できません。
prog3a.c(19) : error C2146: 構文エラー: ';' が、識別子 'printf' の前に必要です。
Z:\programming>
    
```

エラーの内容

文字 '0x81' は認識できません。
 文字 '0x40' は認識できません。 ↔ 全角スペースがある

構文エラー: ';' が、識別子 'printf' の前に必要です。
 ↔ printf の前の行で「;」を書き忘れている

(18), (19)はエラーのある行数を示す。
 この場合18行目, 19行目付近の記述が間違っている。

エラーのあるソース (.cファイル)

```

Z:\programing\prog3a.c - 秀丸
ファイル(F) 編集(E) 検索(S) ウィンドウ(W) マクロ(M) その他(O)
#inclu de <stdio.h>↓
↓
void main ( void )↓
{↓
    double dat ;↓
    ↓
    scanf( "%lf", &dat) ↓ ←「;」がない
    ↓
    printf( "input data is [%f]¥n", dat ) ;↓
}↓
↓
[EOF]

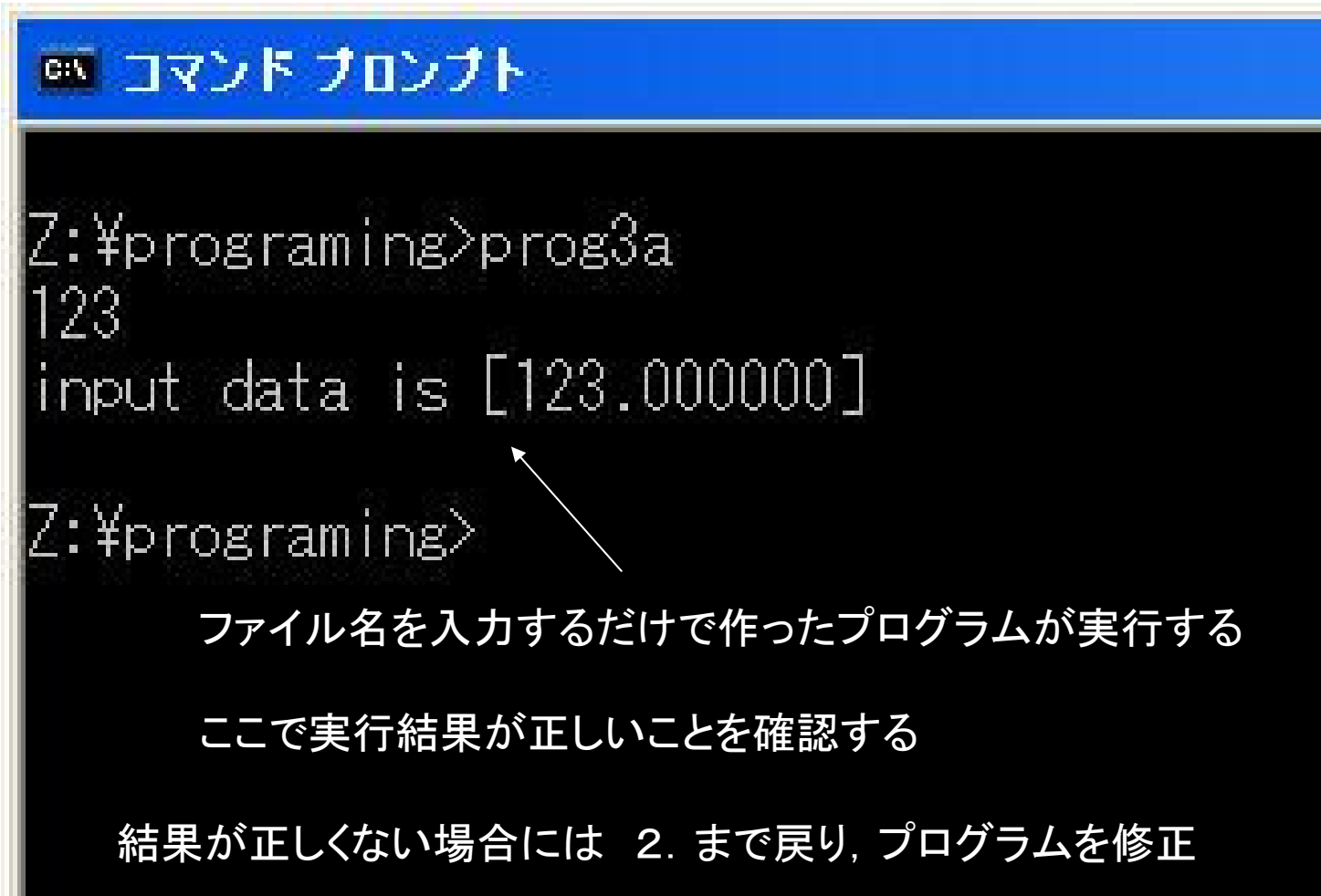
```

見えないが全角スペースがある

8. プログラムの実行

プログラムの実行は【ファイル名】だけでよい. 特に【. exe】を付ける必要はない

z:> 【ファイル名】



```
C:\> コマンド プロンプト

Z:\programing>prog3a
123
input data is [123.000000]

Z:\programing>
```

ファイル名を入力するだけで作ったプログラムが実行する
ここで実行結果が正しいことを確認する
結果が正しくない場合には 2. まで戻り, プログラムを修正

良く使うC言語の命令等の 簡単な手引

A. プログラムの書き方(例)

キーボードから入力した数字をそのまま表示するプログラム例

```
#include <stdio.h>    ...① ヘッダファイル
void main ( void )    ...② 関数の宣言
{
    double dat ;      ...④ 変数の宣言

    scanf("%lf", &dat) ; ...⑤ キーボードからの入力
    printf("input data is [%f]¥n", dat ) ; ...⑥ 画面への出力
}
...⑦ 関数の終わりの括弧
```

- A. 1 ヘッダファイル(①)
- A. 2 関数の宣言(②)
- A. 3 関数の始まりと終わりの括弧(③, ⑦)

これらについては、最初はお約束として必ず書くこととして覚えること。

①

ヘッダファイル `stdio.h` などの【ファイル名.h】には色々な情報が入っています。
`#include` とはこのヘッダファイルを取り込む (include) ことを命令しています。
 その他のヘッダファイルとしては `stdlib.h` などがあり, `#include <stdlib.h>`
 のように書きます。

②

関数は `void main (void)` と書いてありますが, この書式は

【戻り値の型】【関数名】(【引数】)

となっています。

プログラムには必ず1つは **main** という関数名がなければなりません。

戻り値の型と引数は今のところ **void** としておいてください。

将来この部分の拡張について説明します。

③

関数の始まりと終わりには中括弧 `{ }` が必要です。忘れないように！

A. 4 変数の宣言(④)

関数は `double dat;` と書いてありますが, この書式は
【データの型】【変数名】;
となっています.

プログラム内で使う数値や文字を扱う場合, データの出し入れが必要なため途中でデータの内容を変更できる**変数**を宣言して使う必要があります.

データの型は整数(文字を含む)と実数で使い分けます.
整数を使うときには主に `int` を使います.
実数を使うときには主に `double` で良いでしょう.
なお, 文字を使うときには主に `char` を使います.

変数名はどんな名前でも構いませんが, 分かりやすい名前にしましょう.
複数の変数を使う場合には複数宣言する必要があります.

```
char moji;  
int data;  
double fdata;
```

A. 5 printf による画面出力(⑥)

printf の書式は

```
printf(“出力したい文字列”, 変数 );
```

となっています.

表記例	出力
<code>printf(“Hello”);</code>	Hello
<code>printf(“%d”, data);</code>	(data の値)
<code>printf(“整数の値は %d です¥n”, data);</code>	整数の値は (dataの値) です
<code>printf(“実数の値は %f です¥n”, ddata);</code>	実数の値は (ddataの値)です
<code>printf(“文字は %c です¥n”, moji);</code>	文字は (moji の文字) です

%d : 整数の表示 %f : 実数の表示 %c : 文字の表示
 ¥n : 改行

なお, 各変数は以下のような型となっています.

```
int data;
double ddata;
char moji;
```

A. 6 scanf によるキーボードからのデータの入力(⑤)

scanf の書式は

```
scanf ( “入力のデータ型”, &変数 );
```

となっています。

変数の前には **&** が必需なので、忘れないように！

入力データの型によって、キーボードから入力できるデータが決まります。

表記例	入力できるデータ
scanf (“%d”, &data);	整数 → 変数 data へ
scanf (“%lf”, &ddata);	整数または実数 → 変数 ddata へ
scanf (“%c”, &moji);	文字 → 変数 moji へ

なお、各変数は以下のような型となっています。

```
int data;
```

```
double ddata;
```

```
char moji;
```

ちなみに (& + 変数) は、変数のアドレスを示すという意味です

B 実行結果

A で示したプログラムを実行すると...

The image shows two screenshots of a Windows command prompt window. The first screenshot shows the prompt 'Z:\\$programming>prog3a' with an arrow pointing to it and the text 'キーボードからのデータの入力待ち' (Waiting for data input from the keyboard). The second screenshot shows the same prompt with '123' entered, an arrow pointing to it and the text 'キーボードから整数 123 を入力 入力データは変数 dat へ代入' (Enter integer 123 from keyboard, input data is substituted into variable dat). Below that, the output 'input data is [123.000000]' is shown with an arrow pointing to it and the text '変数 dat に入っている数値を 画面へ出力' (Output the numerical value stored in variable dat to the screen). The prompt 'Z:\\$programming>' is visible again at the bottom of the second screenshot.

```

C:\ コマンド プロンプト - prog3a

Z:\$programming>prog3a
← キーボードからのデータの入力待ち

C:\ コマンド プロンプト

Z:\$programming>prog3a
123 ← キーボードから整数 123 を入力
      入力データは変数 dat へ代入
input data is [123.000000]
      ← 変数 dat に入っている数値を
      画面へ出力

Z:\$programming>
    
```