プログラムを作ってみる

三池　克明

まずはプログラムを作ってみましょう。
そして、その作業をとおしてプログラム開発の概要を解説します。

—目　次—

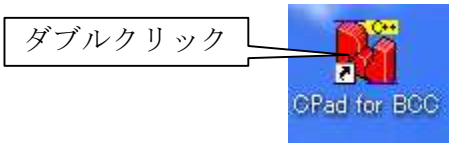
1.　「Hello World!」を表示するプログラム

1.1. ソースファイルを作成する

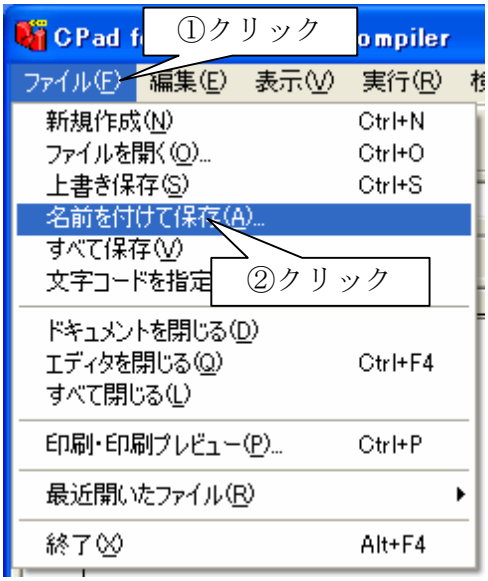
プログラムとはコンピュータに動作を指示する「指示書」のようなものです。
またコンピュータはプログラムの内容を**たとえ間違っていたとしてもそのまま**実行します。
よって、

- 指示の文脈に間違いがある
- 内容の辻褄が合わない
- 危険な指示が含まれている（故意、他意は関係ない）

の場合は動作しないだけでなく、ファイルやパソコンが壊れることもあります。

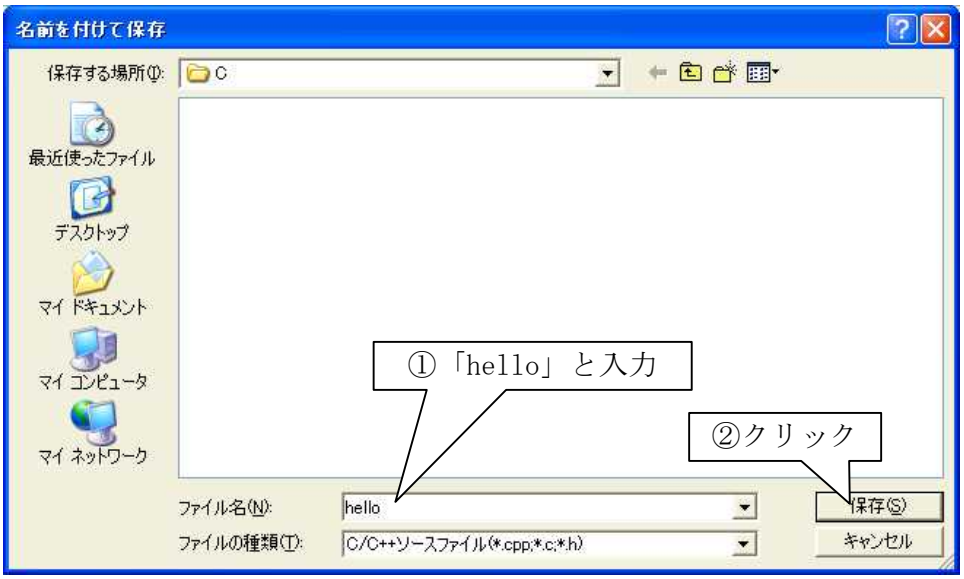


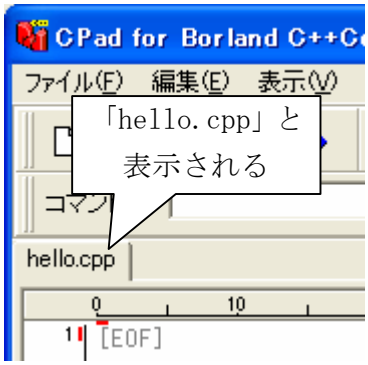
デスクトップのある「CPad for BCC」アイコンをダブルクリックして起動させます。



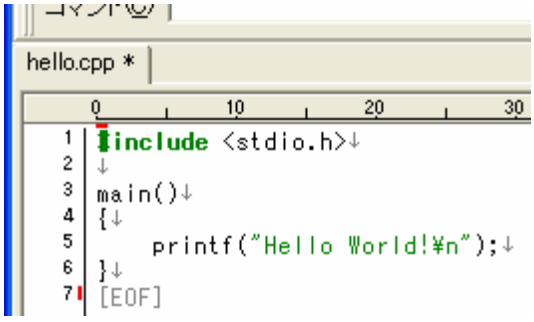
メニュー「ファイル(F)」-「名前を付けて保存(A)」をクリックします。

「名前を付けて保存」ダイアログボックスが表示されるので「ファイル名（N)」欄に“hello”（半角小文字です）と入力し「保存(S)」ボタンをクリックします。





そうするとエディットウィンドウのタブに「hello.cpp」と表示されます。



このようにソースプログラムを入力します。

なお**特定の構文は強調表示されるのでミススペルのチェックに活用しましょう。**

ツールバーの「上書き保存」ボタンをクリックします。
（「Ctrl」キーを押しながら「S」キーを押しても同じです）

―拡張子「cpp」について―

本来、C 言語で記述されたソースファイルの拡張子は「c」です。
一方、「cpp」は C++（シープラスプラス）言語のソースファイルを表します。
C++言語は C 言語を発展させたもので、C 言語のプログラムを記述することができます。よって C++言語の開発環境であったとしても C 言語の学習が可能です

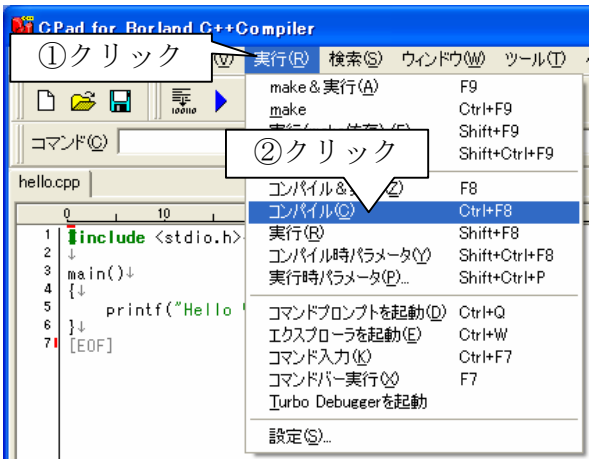
なお本書では、

- コンパイラと開発環境が無償で入手できる
- Windows に完全に対応している
- コンパイル、実行の手順が簡単である

という理由により「Borland C++Compiler」と「CPad for BCC」を選びました。

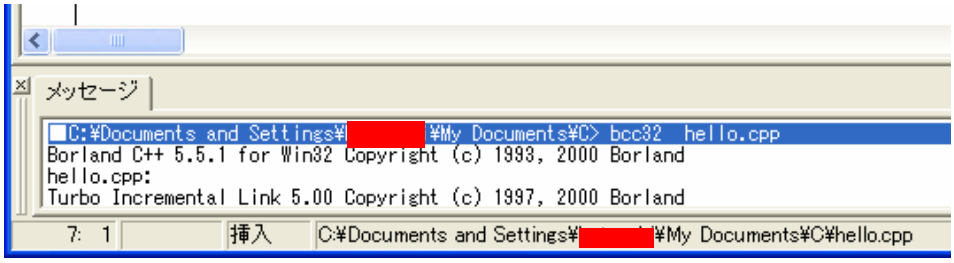
1.2. コンパイルする

ソースプログラムはそのままでは実行できません。
実行させるには「コンパイル」という作業を行い、コンピュータが理解できるプログラムに変換する必要があります。

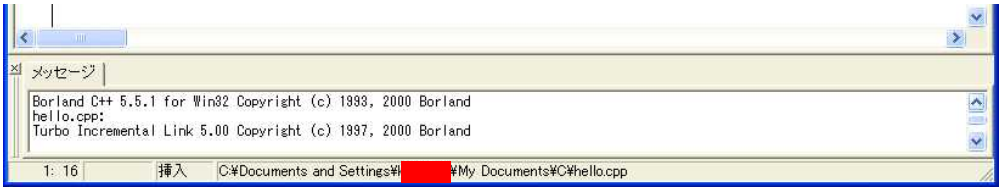


メニュー「実行(R)」-「コンパイル(C)」をクリックします。

「メッセージ」ウィンドウにコンパイルメッセージが表示されます。



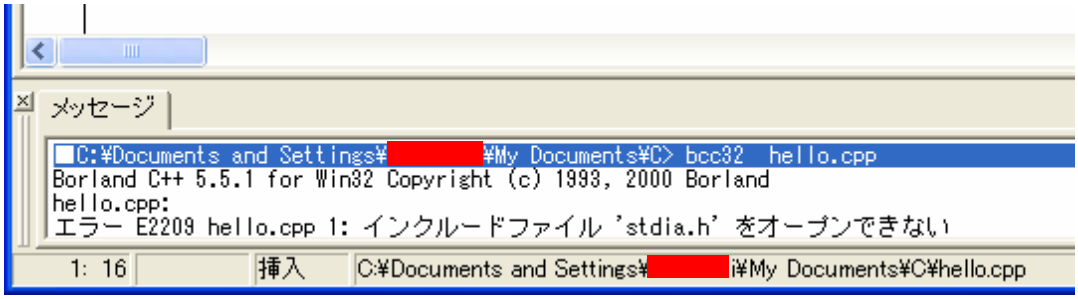
「メッセージ」ウィンドウをスクロールさせ、メッセージに「エラー」が書かれていないか確認します。「エラー」がなければコンパイルは成功です。



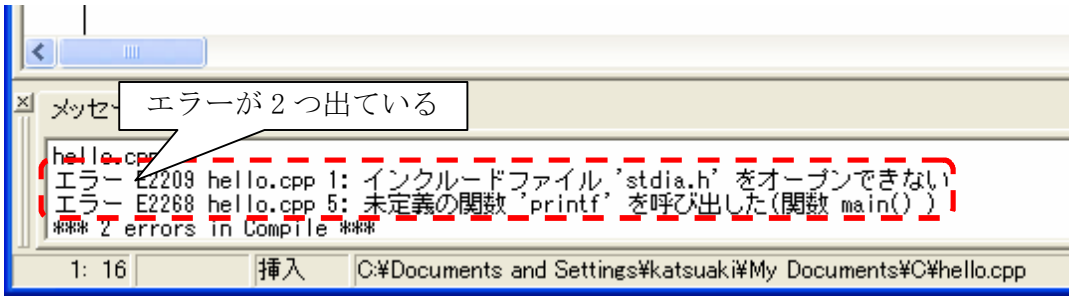
1.3. エラーをつぶす（デバッグ）

コンパイルメッセージに「エラー」が出てしまった場合の対応例です。
エラーの主な原因は**文法の間違い**です。
誤字、脱字がないか確認しましょう。

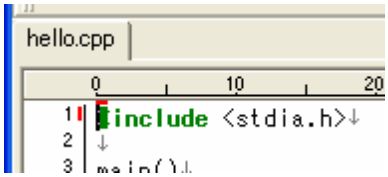
「メッセージ」ウィンドウに表示されているエラーの内容を確認します。



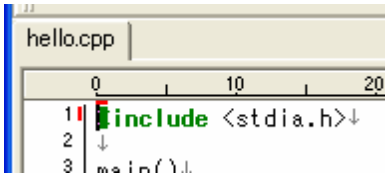
どうやらエラーは２つのようです。最初のエラーから対応しましょう。



最初の「エラー　～」と表示されている行をダブルクリックします。

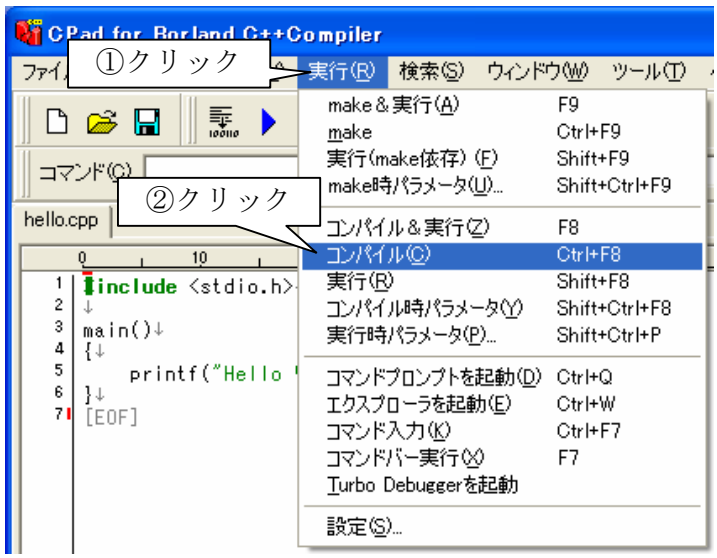


エラーが出た行にキーボードカーソルが移動します。



よく見ると「<stdia.h>」となっていますが、正しくは「<stdi**o**.h>」です。

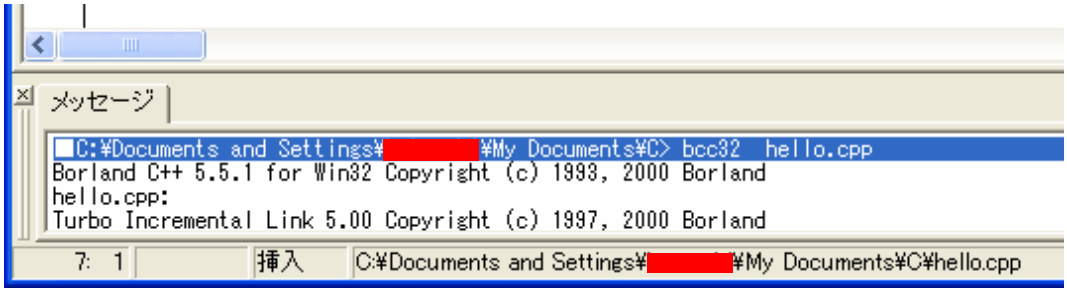
修正しましょう。



修正したら、メニュー「実行(R)」-「コンパイル(C)」をクリックします。

「メッセージ」ウィンドウを見てみると、エラーが無くなったのが分かります。

よって一箇所のミスにより二つのエラーが出ていたことが分かります。

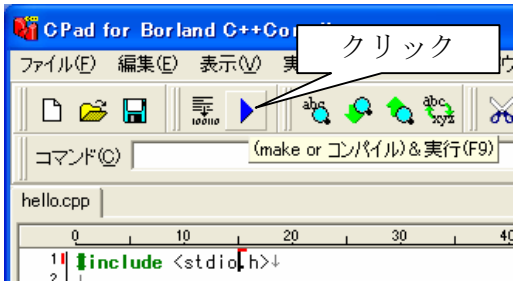


―プログラム開発とエラー―

このように一箇所のミスが原因で複数のエラーを出すことはよくあります。
また、この作業を「デバッグ（虫取り）」といいます。

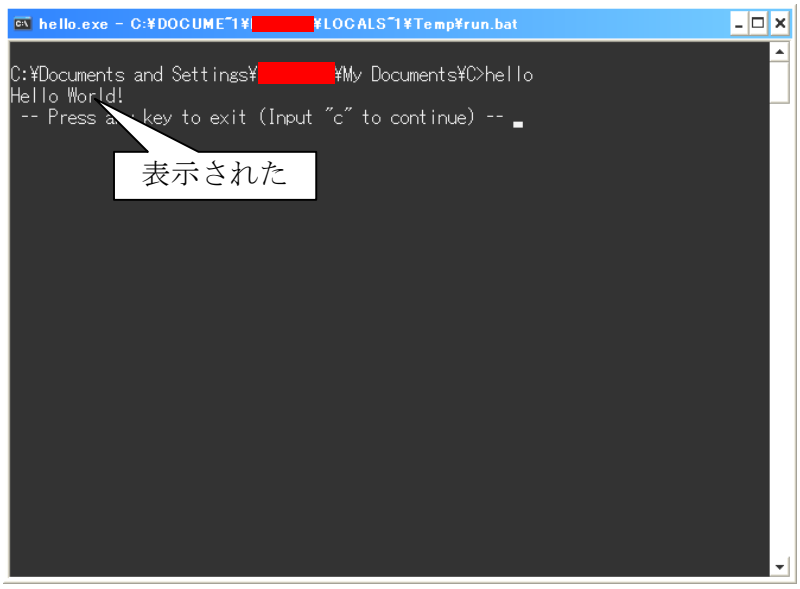
1.4. コンパイルしたプログラムを実行する

コンパイルはソースプログラムの文法だけをチェックします。
決して「あなたの意図どおりに動くかどうか」ではありません。
よってコンパイルして変換した実行プログラムは実行して動作を確認しましょう。



ツールバーの「(make or ｺﾝﾊﾟｲﾙ)&実行」ボタンをクリックします。
(「F9」キーを押しても同じです)

「コマンドプロンプト」ウィンドウが開き、プログラムが実行されます。
「Hello World！」が表示されればプログラムは完成です。



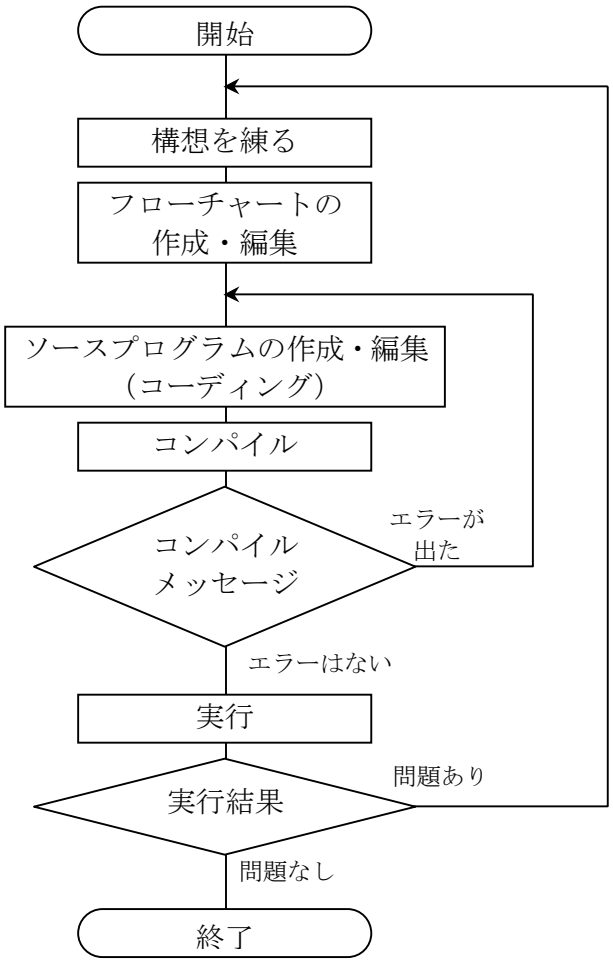
何かキーを押すと「コマンドプロンプト」ウィンドウが閉じます。

2.　プログラムを作る手順

とりあえずプログラムを作ってみました。
ここでは前節で行った作業を思い出しながら「プログラムを作る」手順について考えてみます。

2.1. フローチャート（流れ図）で表してみる

プログラムを作る手順をフローチャート(流れ図)で表すとこのようになります。

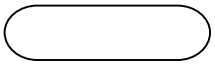
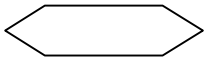
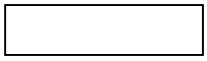
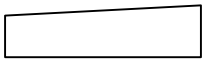
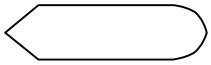
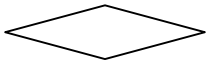
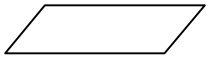
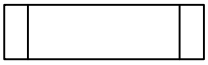


フローチャートとは処理の流れを図に表したもので、基本的に

- **処理の分類を表す記号**
- **記号の中に書かれた説明**
- **前後の処理の接続を表す線**

で構成されます。

また、よく使われる記号とその意味は以下の通りです。

記号	呼び方	意味など
	端子	プログラムや処理単位の開始や終了
	準備	変数の宣言など準備段階の処理
	処理	計算などプログラム内部で行われる処理
	手操作入力	キーボードなどで入力
	表示	ディスプレイなどに表示
	判断	処理の分岐させる
	データ 入出力	ファイルなどからデータを読み書きする
	定義済み 処理	詳細が別紙などに記述されている処理
	結合子	同じ番号（文字）の結合子に続く

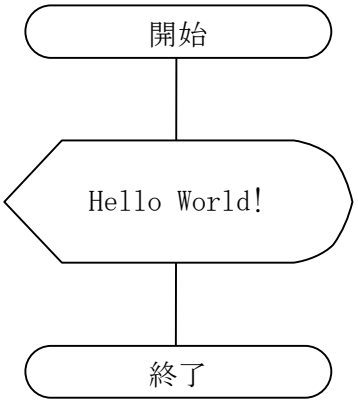
これらの記号や意味を理解してから、前ページのフローチャートを見るとより理解できると思います。

2.2. 「hello.cpp」をフローチャートで表す

本来の順番とは逆になってしまいましたが、さきほど作成した「hello.cpp」をフローチャートで表してみましょう。

「hello.cpp」は、単に「画面に“Hello World!”」と表示するプログラムでした。

これをフローチャートで表すと以下のようになります。



何とも短いフローチャートですね。

2.3. コーディング

ソースプログラムの作成をコーディングといいます。
ここでは既にコーディングしたプログラムの解説をします。

既にコーディングしたソースプログラム「hello.cpp」を見てみましょう。

hello.cpp

```
1:  #include <stdio.h>
2:
3:  main()
4:  {
5:          printf("Hello World!¥n");
6:  }
```

※見やすくするために左側に行番号をつけました。

- 1行目：#include ～

5行目の関数printf()を使うにはこの構文が必要です。
今は理由を考えず「そういうものだ」と思ってください。

- 3行目：main()
- 4行目：{
- 6行目：}

プログラムの実行する内容はmain(){ ～ }の間に記述します。

- 5行目：printf(～

関数 printf()は文字列などを表示する関数です（関数については別の章で解説します)。
また、文字列は必ず ″（ダブルクォーテーション)で囲んでください。

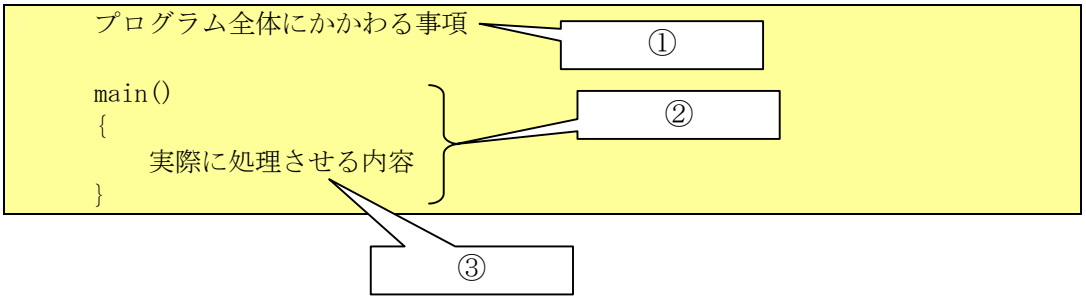
○…″Hello World¥n″　　×…Hello World¥n

3.　C 言語の基本文法

3.1. 全体で見る

「hello.cpp」のような簡単な構造を解説します。

C 言語のプログラムを単純に考えると以下のような構造になります。



① プログラム全体に関わる事項

使用する関数に応じて#include 文を一つ（あるいは複数）記述します。
またそれ以外の構文を記述することもあります。

② main(){ ～ }

実際に処理させる内容をこの中に記述します。

③ 実際に処理させる内容

上から一つずつ順番に実行されます。
また各構文の区切りには;（セミコロン）を付けなければなりません。

○…printf("Hello World¥n");　×…printf("Hello World¥n")

またそれぞれの構文は printf()関数などで表示する文字列を除き、**全て半角文字で記述してください**。

3.2. #include について

構文：

#include <*「ヘッダファイル名」*>

printf()などの関数を使うときは必ず必要です。
とりあえず「そういうものである」と思ってください。
また主なヘッダファイルをリストにしました。

ヘッダファイル名	使用する関数の種類
stdio.h	標準入出力処理で使用する関数。 画面表示、キーボード入力やファイル入出力で使用する関数を使うときに必要。
stdlib.h	データの型変換、乱数生成の関数を使うときに必要。
string.h	文字列処理の関数を使うときに必要。
conio.h	ディスプレイ表示で使用する関数。 画面表示位置や色指定の関数を使うときに必要。

3.3. 関数 printf()について

構文：

printf(*「文字列」*)

ヘッダファイル：

stdio.h

文字列を画面に表示します。また文字列は ″（ダブルクォーテーション）で囲みます。

また文字列には改行などの特殊文字が使えます。

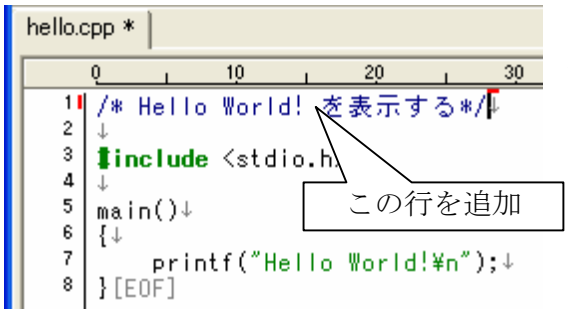
主な特殊文字をリストにまとめました。

特殊文字	その機能
¥n	改行します。 以降に表示される文字列は一行下の左端からになります。
¥a	ベルが鳴ります。
¥b	バックスペースを入力するのと同じです。
¥¥	¥そのものを表示します。

例	実行結果
printf(″Hello″); printf(″World!″);	HelloWorld!
printf(″Hello¥n″); printf(″World!″);	Hello World!
printf(″Hello¥nWorld!″);	Hello World!
printf(″Hello¥bWorld!″);	HellWorld!
printf(″Hello¥¥nWorld!″);	Hello¥nWorld!

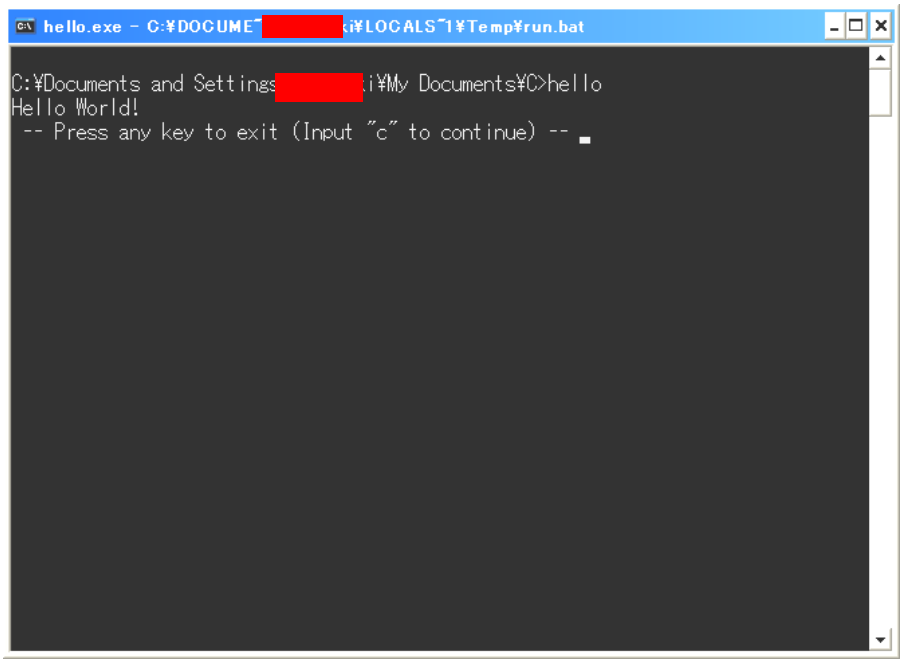
3.4. コメント

コメントはコンパイル時に読み飛ばされる文字列で、/*と*/で囲みます。
これはソースプログラムの補足や使い方などを記述するときに利用します。
なお、「hello.cpp」には記述していませんので、ここで挿入しておきましょう。



「hello.cpp」をこのように修正します。

上書き保存してコンパイル、実行させても結果が同じなのが分かります。



3.5. できるけどやってほしくない記述法

ここでは、プログラムとして機能はするけど、それでもやってほしくない記述法、いわゆる「見づらい」記述法について説明します。

- **その１：改行が少ない**

実はこのように改行やスペースを削除してもプログラムは機能します。

```
1:  #include<stdio.h>
2:  main(){printf("Hello World!¥n");}
```

しかし、構文の区切りが見づらいのでやめましょう。

- **その２：タブ（字下げ）がない**

あえて字下げを削除してみました。

```
1:  #include <stdio.h>
2:
3:  main()
4:  {
5:  printf("Hello World!¥n");
6:  }
```

この程度の長さのプログラムであれば気になりませんが、慣例上｛ ｝の間に記述する文はタブ文字などで字下げしましょう。

- **その３：コメントがない**

「hello.cpp」のような小さなプログラムであればコメントがなくても内容を把握できると思います。

しかし、規模の大きいプログラムにコメントが全く無いのは他人はもちろん、作った本人も内容を把握しづらくします。

よって、適度な量のコメントを付けるよう心がけましょう。

- **その４：コメントが多すぎる**

「その３」の逆で、多すぎるコメントはかえって混乱します。

```
 1:  /*-- hello.cpp ---------------------------*/
 2:  /*  Hello World! を表示するプログラム      */
 3:  /*---------------------- Katsuaki MIIKE --*/
 4:
 5:  #include <stdio.h>            /* stdio.h をインクルード */
 6:
 7:  /* main()はじまり */
 8:  main()
 9:  {
10:     printf("Hello World!¥n"); /* printf()関数で Hello World! を表示する */
11:  }
12:  /* main() 終わり*/
13:
14:  /*------以上、hello.cpp 終わり -----------*/}
```

本書のように教育目的であれば、この程度のコメントはあっても良いかもしれません。

しかし、C 言語をある程度知っている人にとっては苦痛です。

特に 5 行目以降のコメントは構文の内容をそのまま記述しただけです。

コメントは**見ただけでは把握できない**内容に対して付けるべきでしょう。

4. 演習問題

1-1. 以下の文字列を表示するプログラム「ensyu1-1.cpp」を作りなさい。

実行結果

```
あいうえお
```

ヒント：

「hello.cpp」の「Hello World!」を修正すればできますね。

1-2. 以下の文字列を表示するプログラム「ensyu1-2.cpp」を作りなさい。ただし printf()関数は一つだけにすること。

実行結果

```
かきく
けこ
```

ヒント：

改行をする特殊文字は「¥n」でしたね。

1-3. 以下の文字列を表示するプログラム「ensyu1-3.cpp」を作りなさい。ただし printf()関数は二回使うこと。

実行結果

```
１２
```

ヒント：

「¥n」を入れなければ改行は絶対しませんよね。

1-4.　以下の文字列を表示するプログラム「ensyu1-4.cpp」を作りなさい。ただし printf()関数は二回使うこと。

実行結果

```
さしす
　　せそ
```

ヒント：

スペースを使ってみてはいかがでしょう。

1-5.　以下の文字列を表示するプログラム「ensyu1-5.cpp」を作りなさい。

実行結果

```
た
  ち
    つ
      て
        と
```

1-6.　以下の文字列を表示するプログラム「ensyu1-6.cpp」を作りなさい。ただし printf()関数は二回使うこと。

実行結果

```
　　ねの
なにぬ
```

ヒント：

一行目に「　　ねの」を、二行目に「なにぬ」を表示させれば完成です。

1-7.　以下の文字列を表示するプログラム「ensyu1-7.cpp」を作りなさい。

実行結果

```
　　ほ
　　へ
　ふ
  ひ
は
```