

ステップ30

# C++ ワークブック

Step 01	C++とは	6
	1.1 C++言語	
	1.2 C++プログラムを作成するには	
	1.3 C++プログラムの作成環境	
	1.4 プログラム環境のチェック	
	1.5 コメント文の書き方	
Step 02	画面に出力するには	10
	2.1 数字を出力する	
	2.2 文字を出力する	
Step 03	計算するには 変数の利用	14
	3.1 変数の型宣言と変数の型	
	3.2 初期化	
Step 04	変数を使った計算	18
	4.1 四則演算と代入と画面出力	
	4.2 演算の優先順位	
Step 05	数式の特殊な書き方と除算の注意	22
	5.1 代入演算子	
	5.2 インクリメント、デクリメント	
	5.3 除算の注意	
Step 06	型を変える・値を固定する	26
	6.1 自動型変換とキャスト演算子	
	6.2 値の固定	
Step 07	sinやlogを計算する・#で表される処理	30
	7.1 三角関数や対数関数の計算	
	7.2 プリプロセッサ	
Step 08	キーボードから入力するには	34
Step 09	フローチャートとは	38

Step 10	条件によって処理を変えるには	43
	10.1 条件による分岐はなぜ必要か	
	10.2 if文の利用法	
	10.3 条件の書き方	
Step 11	if文の特別な書き方	47
	11.1 if文の特別な書き方	
	11.2 else ifの利用法	
Step 12	if文に複数の条件を書く	51
	12.1 &&   の利用法	
	12.2 if文の落とし穴 フローチャートの重要性	
Step 13	多分岐の処理するには (switch文)	55
Step 14	繰り返し処理を行うには	60
	14.1 繰り返し処理はなぜ必要か	
	14.2 for文の書き方	
	14.3 同じ処理を繰り返す	
Step 15	徐々に数値を変えて処理する	64
	15.1 for文で変化している変数を使う	
	15.2 for文にfor文を入れる	
Step 16	条件を満たす限り繰り返すには	68
	16.1 while文とdo~while文	
	16.2 whileの利用方法	
	16.3 do~whileの利用方法	
Step 17	繰り返し文中断する処理	72
	17.1 break文	
	17.2 continue文	
Step 18	配列とは	76
	18.1 配列	
	18.2 配列の必要性	
	18.3 配列の利用方法	

Step 19	配列の初期化と多次元配列	80
	19.1 配列の初期化	
	19.2 多次元配列	
Step 20	ポインタとは	84
	20.1 変数とメモリとアドレス	
	20.2 ポインタとは	
	20.3 実際にポインタにアドレスを入れる	
	20.4 ポインタが指しているものは？	
Step 21	ポインタと配列、文字列との関係・参照	89
	21.1 ポインタの必要性	
	21.2 ポインタと配列の関係	
	21.3 文字列とポインタ	
	21.4 参照とは	
Step 22	文字列の取り扱い	94
	22.1 文字列と配列	
	22.2 文字列の初期化	
	22.3 文字列の入力	
	22.4 文字列の操作	
Step 23	自分で関数を作るには	98
	23.1 関数の意味	
	23.2 関数の作り方・使い方	
	23.3 関数名に型がある？	
Step 24	関数の型と戻り値・関数の位置とプロトタイプ宣言	102
	24.1 関数の型 戻り値のある関数、ない関数	
	24.2 プロトタイプ宣言とは	
Step 25	関数に複数の値を戻す	106
	25.1 関数に複数の値を戻したいときには（ポインタ編）	
	25.2 関数に複数の値を戻したいときには（参照編）	
	25.3 関数に複数の値を戻したいときには（配列編）	
Step 26	グローバル変数とローカル変数	110
	26.1 グローバル変数とローカル変数	
	26.2 ローカル変数の中身を保存する	

Step 27	クラスとは	114
	27.1 クラスの概念	
	27.2 クラスの定義と宣言	
Step 28	オブジェクトの利用方法	118
	28.1 オブジェクトを使う	
	28.2 アクセス指定子	
Step 29	クラスを再利用する 継承	121
	29.1 クラスの継承	
Step 30	ファイルを利用した入出力を行うには	124
特別講義 1	繰り返し処理の勘所 等差数列と等比数列	130
	等差数列とfor文	
	等比数列とfor文	
特別講義 2	for, while, do ~ whileの共通点・相違点	132
特別講義 3	並べ替え	135
特別講義 4	関数の再帰	137
	索引	138

## 1.1 C++ 言語

C++ 言語は、C 言語のようなプログラミング言語と異なり、「オブジェクト指向プログラミング言語」と呼ばれるものです。

オブジェクト指向プログラミングとは、簡単に言えばある決まった処理を1つにまとめ(このまとまりをオブジェクトと呼ぶ)、プログラムをなるべく簡単に書けるようにしたものです。そのため、C 言語や Basic などと比べて、比較的簡単にプログラムが作成できるようになっています。

C++ には「クラス」という概念があり、これで「オブジェクト」を作成でき、既に存在しているクラスを引き継いだ形で新しい「クラス」を作成できるなど、オブジェクト指向プログラミングが可能となる機能が備わっています。

この本では、最初のほうで C++ の基本的な使い方や計算方法を教え、中ほどでプログラミングで重要な条件分岐と繰り返し文を、次にメモリ上の値の操作方法と関数の作成方法、最後にオブジェクト指向プログラミングで重要なクラスを教えます。

## 1.2 C++ プログラムを作成するには

ここでは、まず C++ でプログラムをする上で最低限しなければいけないことを述べます。

まずは、「プログラムを書く」ことです。以下は、C++ 言語でプログラムするとき最低書かなければならないものです。

```
#include <iostream>
using namespace std;

int main() {

    .....;
    .....;

    return 0;
}
```

それぞれがどのような意味かはここでは説明しませんが、C++ を理解していくとわかってくるでしょう。<iostream> は <iostream.h> と書かなければならない場合があります。

プログラムを作る、ということは、.....の部分に C++ の文や式を書いていくことにな

ります。C++ における文とは、ほぼ1行分の処理の単位をいいます。また、プログラムの命令ごとに、区切り文字として「;」(セミコロン)を入れます。#から}までの全体を、プログラムとか、ソースとか、ソースファイルと呼びます。この本では、主にプログラムと呼ぶことにします。プログラムを保存する際には、ファイル名の最後に拡張子として必ず「.cpp」を書かなければいけません。

プログラムを書くには、「エディタ」が必要です。エディタには、Windows系では「メモ帳」や「ワードパッド」、Mac OS系では「SimpleText」、UNIX系では「vi」や「Emacs」などがよいでしょう。

次は「コンパイルする」ことです。コンパイルとは、上記のプログラムをコンピュータで理解できる形に変換するものです。上記のプログラムは、あくまでも人間がわかるように書かれたものなので、それをコンピュータがわかるものにしなければならないのです。

コンパイルするには「コンパイラ」というソフトウェアが必要です。コンパイルは、UNIX系のOS上で実現する場合、端末エミュレータなどから、

```
c++ source.cpp
```

などとしてください。上記の c++ が「コンパイラ」を実行するコマンドです。もし、プログラムが間違っていてコンパイル時にエラーが出た場合にはプログラムを修正します。

その後、コンパイルが成功すると「実行ファイル」(UNIX系OSの場合、a.exe や a.out など)が生成されます。それを

```
a.out
```

とタイプして実行すると、プログラムしたとおりの結果が画面に反映されます。

## 1.3 C++ プログラムの作成環境

C++ プログラムを作って、それを実行するにはいろいろな方法があります。以下に各OSでの作成環境について簡単に述べます。

まず、Windowsでは市販のコンパイラを使うことが多いです。Microsoft社のVisual C++などがその代表です。また、Cygwinという、Windows上にUNIX環境を構築できるものがあるので、これを利用してもよいです(一般に無料で手に入ります)。

またUNIX(Solarisなど)やLinuxを使っている場合には、市販のCコンパイラを購入するか、g++というフリーのコンパイラをダウンロードするなどして入手します。特にLinuxの場合は、インストール時にDevelopmentというパッケージにg++があるので、これを選択すれば自動的にインストールされます。

Mac OSの場合は、Mac OS Xであれば標準でインストールされることが多いので、「Terminal」からg++を実行して、「g++はありません」というメッセージではなく、「No Input File...」などのコメントが表示されれば、そのままの環境でC++プログラムをコンパイルできます。

最近、ネット上などで無料で手に入るC++コンパイラが多くなりました。この本のプログラムは、それを利用するので十分です。

## 1.4 プログラム環境のチェック

ここで、これからプログラムを作成していく上で、使っているシステムがどのようなものかをチェックします。

次のプログラムを、使うシステムで書きます。このときは上で説明した通り、エディタを使います。

sample1-1.cpp

```
01: #include <iostream>
02: using namespace std;
03:
04: int main() {
05:
06:     cout << 1;
07:
08:     return 0;
09: }
```

次にこれをコンパイラでコンパイルします。UNIXかLINUX、Cygwin、Mac OSなどを使っている場合は「c++ ファイル名」または「g++ ファイル名」です。また、Windowsなどの統合環境（Visual C++など）を使っている場合は、「コンパイル」、「ビルド」などをメニューから選びます。

ここでプログラムは完全なのに、エラーが出る場合は、上の<iostream>が<iostream.h>でなければならない環境で使っている可能性があるため、<iostream.h>に変更してみます。この変更でコンパイルが完成したなら、以降の<iostream>はすべて<iostream.h>に変更してください。

次に、コンパイルをしたらUNIXかLINUX、Cygwin、Mac OSなどの環境の場合「a.out」や「a.exe」などのファイルが作られるので、それを実行します。Windowsなどの場合は、メニューから「実行」などを選びます。

これらがうまくいけば、これからの例題や演習問題はすべて実行できます。

## 1.5 コメント文の書き方

プログラムを作成する際、特に人に見せたり、後輩に引き継ぐ予定であったり、自分だけが見るものであっても大きかったりするプログラムの場合には、それぞれの文や式が何を意味しているのかのコメントを書いておくべきです。

C++におけるコメントは

```
a = 1 + 2 + 3 + 4; //1から4までの足し算をしました。
```

のように、//を記述すると、//からその行の終わりまでがコメント文となります。

また、C++言語のもととなっているC言語で使われていた/\*と\*/とで囲われた部分もコメント文になります。よくプログラムの初めに、

```
/*
*****

このプログラムは、成績
を分析するためのプログ
ラムである。

*****
*/
```

と書いてある場合がありますが、これは、/\*と\*/とで囲われたコメント文となっています。つまり、//は1行のコメントのときに、/\*と\*/は複数行のコメントに用います。

## 2.1 数字を出力する

画面に文字や数字を出力するには、`cout`と`<<`を使います。例えば、数字の1を出力するには、次のようなプログラムを書きます。

sample2-1.cpp

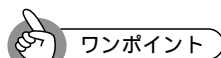
```
01: #include <iostream>
02: using namespace std;
03:
04: int main() {
05:
06:     cout << 1;
07:
08:     return 0;
09: }
```

このプログラムをエディタで記述し、コンパイラでコンパイルした後、実行すると画面上に「1」が出力されます。

実行結果

1

つまり、`cout`はそれ以降`<<`でつながれた文字や数字を表示する命令であることがわかります。`<<`は複数の出力対象をつなぐ役目をもちます。



ここでいう数字の1は半角文字の1です。日本語を入力するモードにおいて出力される全角文字の1とは違います。

また、数字を連続して表示するには、

```
cout << 1 << 2 << 3;
```

としてもよいし、

```
cout << 123;
```

としてもよいです。

## 2.2 文字を出力する

文字を出力するにもやはり`cout`を使います。画面上にaを出力するためには次のようなプログラムを書きます。

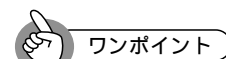
sample2-2.cpp

```
01: #include <iostream>
02: using namespace std;
03:
04: int main() {
05:
06:     cout << 'a' << '\n';
07:
08:     return 0;
09: }
```

実行結果

a

文字を出力するには、数字のときと異なり、' 'で囲って表現します。また、`'\n'`は`\n`を画面上に書くためではなく、改行（リターン）する特殊な文字です。



システムやOSによっては、`¥`が`\`（バックスラッシュ）のときがあります。

`¥`マークを付けて文字以外の機能を持たせるものをエスケープシーケンスといいます。エスケープシーケンスのいくつかの例を表に示します。

表2.1 エスケープシーケンスとその機能

エスケープシーケンス	機能
<code>¥n</code>	改行（リターン）
<code>¥t</code>	タブ
<code>¥ooo</code>	8進数oooのコードに対応した文字
<code>¥xhh</code>	16進数hhのコードに対応した文字

文字は必ず1つのコードと対応しており、例えば文字'a'の場合、8進コードは141、16進コードは61なので、`¥141`または`¥x61`とすれば、画面上にaが出力されることとなります。

数字の出力の例では、この改行を意味するエスケープシーケンスは書かなかったため、実際にプログラムを実行すると、UNIX系OSでは、

```
%a.out
1%
```

と、1のすぐ後にプロンプト（ここにコマンドが入力できるというしるし）が出てしまいましたが、この改行を入れると、

```
%a.out
a
%
```

となります。

また、文字を連続して出力するには、

```
cout << 'a' << 'b' << 'c' << '\n';
```

と書くか、

```
cout << "abc" << '\n';
```

と書くかのどちらかです。

ここで注意しなければならないのは、上の2つの表現は、数字を連続して出力するときのように簡単にはいかない、ということです。つまり、上の例では、文字を囲っているのは「'」であるのに対し、下の例では「"」であるという違いがあります。

#### ワンポイント

「"」は「'」2つではありません。また「`」2つでもありません。

これは、文字と文字列の違いに依存しているのです。

C++ではこの文字と文字列を厳密に区別しています。文字とは1文字を意味しており、文字列とは、2文字以上を意味しています。ですから、1番目の例では1文字ずつで表現しているのので'を使い、それを<<でつなげて、出力を文字列のようにしているのです。

また2番目の例では、文字列として書いているので"を使って出力しています。ただし、エスケープシーケンスは2文字でも1文字（1バイト）を意味しているので、改行は'で囲みます。また、文字列の中には、ひらがなやカタカナ、漢字が含まれます。

#### ワンポイント

数字、カタカナには、半角と全角があるので注意！

カタカナ（半角・全角とも）や漢字は1文字でも文字列の扱いになります（1文字で2バイト必要であるため）

ひらがなやカタカナ、漢字を表示するには、

```
cout << "C++プログラムを勉強しましょう！¥n";
```

このように書きます。

#### ワンポイント

数字を出力するのに、

```
cout << 123;
```

としましたが、abcを出力するときと同様、

```
cout << "123";
```

と書いても同じ出力が得られます。ただし、両者には明らかな違いがあり、前者は「百二十三」という数値を意味しますが、後者は「1」と「2」と「3」という文字を3つ並べたにすぎません。この2つはちゃんと区別してください。

### 演習

- (1) 自分の名前、生年月日、年齢、住所を画面に表示するプログラムを作成せよ。
- (2) (1)のそれぞれの行に、コメントを入れてみよ。