

ステップ30

# C言語

## ワークブック

Step 01	C言語とは	6	Step 08	繰り返し文(while)	34
	1.1 C言語でできること			8.1 while文とは	
	1.2 C言語のメリット			8.2 具体的な使い方	
	1.3 C言語の歴史			8.3 While文のフローチャート	
	1.4 C言語でのプログラム作成		Step 09	[応用]2重ループ	38
	1.5 プログラムを作ってみよう			9.1 forを使った2重ループ	
				9.2 whileを使った2重ループ	
Step 02	数の種類と表示	10	Step 10	関数	42
	2.1 数の分類			10.1 関数とは	
	2.2 変数の定義			10.2 関数を定義する順番	
	2.3 数値の表示		Step 11	ヘッダー	46
Step 03	数値の入力	14		11.1 ヘッダーファイルとは	
	3.1 scanf文			11.2 #defineの使い方	
	3.2 scanf文で起こしやすい間違い		Step 12	ライブラリー	50
	3.3 数値の種類の変換			12.1 ライブラリーの分類	
Step 04	[応用]図形の面積	18		12.2 ライブラリーの利用	
	4.1 図形の面積を求めよう		Step 13	数値計算(2次方程式の解)とモンテカルロ法	54
	4.2 プログラム作成			13.1 2次方程式の解の公式	
	4.3 フローチャート			13.2 モンテカルロ法	
Step 05	条件分岐(if文)	22	Step 14	例外処理	58
	5.1 if文とは			14.1 例外処理とは	
	5.2 if文の使い方			14.2 例外処理の例	
	5.3 条件分岐のフローチャート			14.3 例外処理のフローチャート	
Step 06	条件分岐(if文複合条件)	26	Step 15	ファイルへの書き出し	62
	6.1 複雑なif文			15.1 ファイルポインタ	
	6.2 プログラム例			15.2 ファイルの書き出し例	
Step 07	繰り返し文(for)	30		15.3 図形の描画	
	7.1 for文とは				
	7.2 具体的な使い方				
	7.3 For文のフローチャート				

Step 16	情報の可視化	66
	16.1 Excel のためのデータ書き出し	
	16.2 Excel によるグラフの表示	
Step 17	[ 応用 ] バイオリズム	70
	17.1 バイオリズムの計算	
Step 18	配列	74
	18.1 配列とは	
	18.2 配列の使い方	
	18.3 配列の初期化	
	18.4 配列のコピー	
Step 19	多次元配列	78
	19.1 多次元配列を使うメリット	
	19.2 多次元配列の利用	
	19.3 多次元配列の使い方	
	19.4 多次元配列の具体的な応用	
Step 20	[ 応用 ] カレンダーの表示	82
	20.1 配列の準備	
	20.2 月始めの曜日の計算	
	20.3 月の最終日の計算	
	20.4 プログラムの説明	
Step 21	関数と配列	86
	21.1 配列を利用した関数	
	21.2 配列を使用した関数の応用	
Step 22	ポインタ	90
	22.1 ポインタの説明	
	22.2 ポインタの簡単な使い方 ~ アドレス演算子と参照演算子 ~	
Step 23	ポインタと配列	94
	23.1 ポインタと配列の仕組み	
	23.2 ポインタと多次元配列	

Step 24	ポインタと関数	98
	24.1 ポインタを使った関数の利用	
	24.2 ポインタと関数の注意点	
Step 25	構造体	102
	25.1 なぜ構造体を使うのか?	
	25.2 構造体を用いた実現	
	25.3 構造体と関数	
Step 26	文字列	106
	26.1 文字の利用	
	26.2 文字列の利用	
	26.3 文字列処理関数	
Step 27	[ 応用 ] アドレス帳	110
	27.1 住所録データの作成	
	27.2 構造体と文字列処理	
Step 28	[ 応用 ] 素数の計算	114
	28.1 エラトステネスのふるいとは	
	28.2 C言語による実現	
Step 29	[ 応用 ] 図形の描画	118
	29.1 設計指針	
	29.2 PGMフォーマット	
	29.3 円形ゾーンプレートの生成	
Step 30	[ 応用 ] Web ページの作成	122
	30.1 HTML言語	
	30.2 HTMLタグ	
	30.3 タグを生成する関数	
	30.4 簡単なページの作成	

## C言語とは

C言語とはどのようなものなのでしょうか？ どうしたら Word や Excel のようなアプリケーションを作成できるようになるのでしょうか？ ここでは C言語の歴史や使い方等について説明します。

### 1.1 C言語でできること

C言語は主に Windows に代表されるようなオペレーションシステムの開発に用いられてきました。いわゆるシステムプログラミングといった分野に適しています。歴史のところでも述べていますが UNIX と呼ばれる OS は C言語と共に発達しました。また近年話題になっている Linux も C言語で書かれています。その他のアプリケーションも C言語で書かれています。また、携帯電話や家電製品の内部で動いているプログラム（組み込み）も C言語で書かれる場合が多くあります。

### 1.2 C言語のメリット

C言語はコンパイラ型言語です。これは C言語で書かれたプログラム（ソース）をコンピュータが実行可能なコード（オブジェクトコード）にすべて変換するやり方です。このためプログラムが大変高速に動作します。またメモリーの消費も少なく済みます。このため、C言語は家電製品やロボットのシステム（図1.1）にも使われています。



図1.1：無人海底探査ロボット MR-X1（独立行政法人海洋研究開発機構）

その他にも、C言語を使うといろいろなメリットがあります。たとえば、大型コンピュータ（図1.2）や WS（図1.3）、PC、1chip マイコン（図1.4）など、大きさや能力の異なるコンピュータのプログラムを C言語ならば共通するプログラムをほぼそのままで作成することができます。また、プログラムの事実上の標準言語となっており、IT 業界の SE の募集において必須となっている場合が多くあります。



図1.2：地球シミュレータ  
（独立行政法人海洋研究開発機構）



図1.3：  
ファイルサーバ  
ワークステーション  
（サン・マイクロシステムズ）

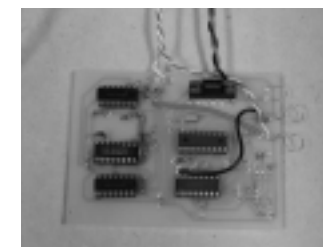


図1.4：1chip マイコン  
（MicrochipPIC シリーズ）

### 1.3 C言語の歴史

- ・初期  
C言語は 1970 年代にベル研究所で開発され、主に UNIX の開発に使用されました。UNIX が爆発的に普及するに従い C 言語も普及していきました。なお、この C言語は K&R C と呼ばれています。
- ・中期  
1983 年に ANSI C が提案され、より互換性が高まりました。現在の C言語はこの ANSI C です。また、PC でも使える C言語が普及したので一般ユーザでの使用が広がっています。
- ・現在  
C言語はエンジニアの必須言語（アルゴリズム等がこれで書かれる）です。また、オブジェクト指向を取り入れた C++ が現在の主流となっています。

### 1.4 C言語でのプログラム作成

C言語でプログラムを作る場合は以下の2つが必要です。学校により環境が異なりますので先生の指示に従って下さい。この本では Linux をベースとしていますので gcc を使用します。

- ・エディター  
プログラムを書きます。プログラムを書くためのワープロのようなものです。UNIX では「vi」や「emacs・gedit」、MS-Windowsでは「秀丸」や「ノートパッド」が有名です。
- ・コンパイラとリンカー  
コンパイラはエディターで書いた C言語のプログラムが文法的に正しいかをチェックし、コンピュータの実行形式に変換します。リンカーはプログラムとライブラリを結合するのに使いますが、最近ではコンパイラが面倒を見てくれます。コンパイラは、UNIX では「gcc」、Windows では「Visual C++」や「Borland C」が有名です。

## ワンポイント

Windows でもgcc は使えるのかといった質問がよくありますが、Windows にもあります。

### ・ Mingw

簡単に導入でき、Windows で動くプログラムが作成できる等のメリットがあります。

英語 (<http://www.mingw.org/>)、日本語 (<http://mingw.biggie.jp/>)

### ・ Cygwin

Windows 上で GNU プロジェクトを使うためのシステムです。gccの他にもいろいろなUNIX上のツールが手に入ります。

英語 (<http://www.cygwin.com/>)

また、これを機会に思いきって Linux を導入するのも手だと思われます。初心者でも CD-ROM一枚で簡単に動かせる「Kinoppix」や日本語にすぐれた「Vine」がお勧めです。C言語のみを勉強するのなら「Kinoppix」がよいでしょう。

Kinoppix ..... <http://unit.aist.go.jp/it/knoppix/>

Vine ..... <http://vinelinux.org/>

Mac OS X には開発環境が標準で準備されています。アップルよりダウンロードして下さい。

この例でもわかると思いますが C言語には、以下のような特徴があることに注目することが大切です。これが、C言語のもっとも簡単なプログラムとなります。

- ・ { } は必ずペアである
- ・ 文の終わりは ; である
- ・ **main** が必ずある
- ・ **/\*\*/** はコメントである

## 演習

- (1) HELLO を2回表示するプログラムを作成しなさい。
- (2) 演習(1)のプログラムをプリンターで印刷し、プログラムの右にコメントを書きなさい。
- (3) Linux の操作に慣れ、端末、エディター、コンパイラが使えるようになりなさい。

## ワンポイント

プログラムの書き方ですが、人によりいろいろな書き方があります。例としてあげられるのは、

```
main(){
    printf("HELLO\n");
}
```

というように、; をつけてはならない関数や命令の前に { を置くといった手法です。これは初心者にありがちな何でも ; をつけるといった間違いを防ぎ、リストを見やすくできます。また、プログラムが長くなり見にくくなる場合は、字下げ(インデント)を使うと見やすいコードが書けます。この時スペースを使うのではなく、TAB を使うと効果的です。どのような書き方でも C言語としては文法的に正しいのですが、なるべく見やすいプログラムを作ることが上達のポイントです。

## 1.5 プログラムを作ってみよう

今回は、簡単なプログラムを入力し実行してみます。「sample1.c」がそのプログラムです。なお、左端の行番号を入力してはいけません。これは説明のために記載してあります。右側はコメントです。プログラム作成の手順は、以下のようになります。

1. エディターでプログラムを入力し、「**sample.c**」で保存。
2. 端末上で「**gcc sample.c**」と入力、エラーがあれば1へ戻る。
3. 端末上に「**./a.out**」と入力しリターン。
4. 端末上に「**HELLO**」が出力される。

sample1.c	(コメント)
01 #include<stdio.h>	01 stdio.hを読み込む
02 main()	02 メイン関数の定義
03 {	03 メイン関数ののはじまりの{
04     /*表示*/	04 コメント
05     printf("HELLO\n");	05 HELLOの表示
06 }	06 メイン関数の終わりの}

## 数の種類と表示

ここでは、変数の宣言、数の種類、表示の基礎について説明します。C言語では数の種類に注意しなければなりません。また、表示についても同様です。このステップでは、これらを中心に説明します。

### 2.1 数の分類

C言語で計算をすることは非常に簡単です。しかし、数値の種類に注意しなければいけません。C言語では、以下の数値の種類がよく使われます。初めは int、float についてのみ説明しますが、その他にも long、char、short といった数値の種類があるので興味のある人は調べてみるとよいでしょう。なお、数値の精度は Linux で使用されているものを基準としています。本書では、初めは int と float のみでプログラムを書くので、これだけは必ずマスターして下さい。また数学的意味として、整数とは 0 を含む正負の自然数になります。実数はすべての数を示します。よって、整数は実数に含まれます。つまり、3 3.0 であるため、int a; a=3.0; は実行できません。また、初心者ありがちな間違いとして、何でも float で定義すればよいと考える場合がありますが、これはやめるべきです。

表2.1

	C言語	例
整数(32bit)	int	int a; a=5;
実数(32bit)	float	float b; b=2.27;
倍精度実数(64bit)	double	double c; c=3.145;

### 2.2 変数の定義

C言語では、使用する変数を必ず定義する必要があります。まずはサンプルプログラムを見てください。この例からもわかるように、変数の定義は main のすぐあとに記入します。つまり、変数の定義とは使用する箱(メモリ)を確保し名前を付けることになります。そして数値の代入は、X=1; のように = を使って記述します。

できたら「sample2\_1.c」を入力し、コンパイルして実行してみましょう。なお、int 型の変数に float 型の数値を代入できないことに注意してください。

```
int a;          a=5;
```

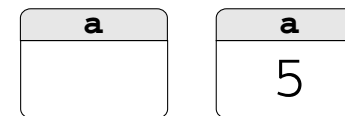


図2.1: 変数の定義と代入

sample2\_1.c

```
01 main()
02 {
03     int x,y,z;
04     int ans;
05     x=1;
06     y=2;
07     z=3;
08     ans=x+y+z;
09     printf("%d\n",ans);
10 }
```

(コメント)

```
01 メイン
02 始まりの{
03 整数変数の定義 x,y,z
04 整数変数の定義 ans
05 xに1を代入
06 yに2を代入
07 zに3を代入
08 ansにx+y+zを計算し代入
09 ansの値を表示
10 終わりの}
```

### 2.3 数値の表示

それでは、数値を表示するにはどうしたらよいでしょう? 「sample2\_1.c」にあるように printf の中で %d と書いているところに注目してください。これは int 型の数値を出力せよというフォーマットです。" で囲まれたところに %d を記入すればよいことになります。

表2.2

int	printf("%d\n",a);
float	printf("%f\n",a);
double	printf("%lf\n",a);

もし、仮に float で定義した変数を int で表示すると、プログラマが望んだ結果が出てきません。このため、出力にも十分注意する必要があります。

C言語で使える計算は数値演算、論理演算など数多くあります。今回は四則演算についてのみ説明します。余りを求める % は、意外とよく使いますので覚えておいて下さい。

表2.3

足し算	C=A+B;
引き算	C=A-B;
掛け算	C=A*B;
割り算	C=A/B;
余りを求める	C=A%B;

sample2\_2.c

```

01 main()
02 {
03     int x,y,z;
04     int ans;
05     x=1;
06     y=2;
07     z=3;
08     ans=x+y+z;
09     printf("ans:%d+%d+%d=%d\n",x,y,z,ans);
10 }

```

(コメント)

```

01 メイン
02 始まりの{
03 整数変数の定義 x,y,z
04 整数変数の定義 ans
05 xに1を代入
06 yに2を代入
07 zに3を代入
08 ansにx+y+zを計算し代入
09 x,y,z,ansの値を表示
10 終わりの}

```

### 演習

- (1) 半径  $r=5$  の円の円周  $L$ 、面積  $S$  を求めよ。ただし、 $\pi=3.14$  とする。  
ヒント: float を使うこと
- (2)  $a=255$ 、 $b=16$  とした時  $a+b$ 、 $a-b$ 、 $a*b$ 、 $a/b$  を求めよ。  
ヒント: 割り算は小数が出ることに注意

### ワンポイント

たとえば、「1 2 3.14」のように複数の数値を表示する時はどうすればよいのでしょうか?

```

main(){
    int x,y;
    float a;
    x=1;
    y=2;
    a=3.14;
    printf("%d %d %f\n",x,y,a);
}

```

というように、3つを並べて書けばよいです。このとき表示する形式(%dなどの形式)と変数が1対1の関係になっていることが大切です。

### ワンポイント

#### C言語特有の数式表現

C言語には、数学と表現方法が異なった数式の書き方があります。これを知らなくてもプログラムは書けますが、非常に見通しの悪いプログラムになったりします。せっかくC言語を学習しているのですから、これを機会にC言語特有の数式表現も覚えておきましょう。

#### ・数値の増加と減少

C言語には、インクリメント(増加)とデクリメント(減少)という数を1つずつ増減させる数式表現があります。

インクリメント	a++	aの値に1を加え、その結果をaに代入する。
デクリメント	a--	aの値から1を減少させ、その結果をaに代入する。

たとえば、インクリメントを a++ で書かなかった場合、以下のようにプログラムが複雑になってしまいます。

```

int a,b;
a=1;
b=a+1;
a=b;

```

a++ を使うと、これが1行で書けるのですからプログラムが非常に見やすくなります。これは、後で説明する for 文でもよく利用されます。

#### ・変数自身への計算結果の代入

これは総和を求める場合などによく使われます。たとえば、以下のように数式を表現します。

- (1)  $sum=sum+1$ ; は 5 に 1 を足し、その結果を  $sum$  に代入します。総和を計算する場合などに便利です(5行目)。
- (2)  $sum+=2$ ; は、 $sum$  に 2 を足し、その結果を  $sum$  に代入します。同様に  $sum-=3$ ; は、 $sum$  から 3 を引き、結果を  $sum$  に代入します(6行目)。

```

main()
{
    int sum;
    sum=5;
    sum=sum+1;
    sum+=2;
    sum-=3;
    printf("%d\n",sum);
}

```

(2)の表現方法はエレガントですが、初心者はバグを起こしやすいので(1)の表現方法をお勧めします。このプログラムでは結果として5が出力されます。