基本的なプログラムの書き方を理解する

Cの基本――書式と予約語，データの型宣言

岸 哲夫

プログラムは闇雲に書き並べれば動くというものではない．C言語にはC言語の確固たる書式があり，それに従わなければならない．ここでは，データの型宣言とデータを表す定数，変数の扱い方について解説する．　（筆者）

プログラムを書くうえでの決まり

● 書式について

▶ プログラム例にそって書き方を覚えよう

それでは，まずはじめに**リスト1**に示す簡単なプログラムを例にとって説明します．

① #include 文で必要なヘッダ・ファイルを読み込む．このstdio.hは名前のとおり標準入出力に関係する関数の宣言（定義）が記述される．ここでは⑥のprintfを使用するときに必要になる

② 関数の宣言．test という関数は引き数が整数で3個必要で，整数を返す定義になっている

③ 実際の関数mainがここに記述されている

④ 実際の関数testがここに記述されている

リスト1　簡単なプログラム例（text1.c）

```
#include <stdio.h>                    ①使用する関数定義が記されている
                                        ヘッダ・ファイル
int test(int a,int b,int c);          ②関数宣言
int main(void)                        ③実際の関数の記述
{
        return test(1,2,3);
}
int test(int a,int b,int c)           ④実際の関数の記述
{
        int aa;                       ⑤ローカル変数の宣言
        aa  =    a + b + c;
        printf("%d¥n",a);            ⑥変数aの表示
        return aa;
}
```

```
岸@main ~
$ gcc test1.c -o test1.exe

岸@main ~
$
```

図1　gccでtest1.cをコンパイルするようす

⑤ ローカルの変数int型のaaという変数が定義されている

⑥ 命令文．標準関数printfを呼び出している

基本は，

- include文（プリプロセッサ文）
- 関数の宣言（プロトタイプ宣言）
- 実際の関数記述
- 変数の定義
- 命令文

という構成です．

include文などを含む「プリプロセッサ文」については後述します．簡単なプログラム例では，#include文は標準で用意されている関数を使用する宣言をするのに使用します．

関数の宣言（プロトタイプ宣言）は自分で作った関数がどのような形式かをコンパイラに定義します．引き数が二つなのに，三つで呼び出していたらエラーにしてくれます．

また，整数を返す定義がしてあるのに，何も返さないように使用していたらエラーになります．

実際の関数記述は必要な関数を記述します．実際のプログラムはmainからスタートします．場合によってはmainがないプログラムも存在します．その場合は呼び出された関数が動作すると思ってください．

変数の定義は関数の外で行うか内で行うかで大きく意味が違います．この件に関しては後述します．

命令文は演算したり，比較したり，変数の中身をセットしたり読み込む命令です．もちろん関数の呼び出しも命令文です．

リスト1をコンパイルするようすを**図1**に示します．

ここでgcc test1.c -o test1.exeと指定したのはCygwinを使っているので，実行形式ファイルの拡張子がexeになるためです．Cygwinを実行しているターミナル上で実行すると実行可能です（**図2**）．

次に，引き数が宣言と違う呼び出しをしている例を**リスト2**に示します．そして，コンパイルした結果を**図3**に示します．

KEYWORD ―― 書式，関数宣言，main，include，printf，gcc，ANSI-1989，C99，型宣言，_Bool型，long long int型，float型，double型，double long型，_Imaginary型，_Complex型，整数型，浮動小数点型，固定小数点型，複素数型，グローバル変数，ローカル変数，文字列ポインタ，配列

```
岸@main ~
$ ./test1.exe
1
```

図2　gccでtest1.cをコンパイルして実行したようす

```
岸@main ~
$ gcc test2.c -o test2.exe
test2.c: In function 'main':
test2.c:5: error: ｴﾗｰ   test' ､ﾘﾝﾎｰ    ｬｾｯﾊﾟｹﾝｮﾞﾝｹ

岸@main ~
$ gcc test2.c -o test2.exe
test2.c: In function 'main':
test2.c:5: error: 関数 'test' への引き数が少なすぎます

游main ~
$
```

図3　gccでtest2.cをコンパイルした結果

実行リストで文字化けしているのは，GCCがEUCコードでメッセージを出しているからです．このような場合，一時的にターミナルをEUCコードに変えれば正しいメッセージが読み取れます．

序章で紹介したPoderosaでエンコードをEUCに変えるのは簡単です．ウィンドウ上部のバーの右側にある「エンコーディング」リスト・ボックスで「euc-jp」を選択すればよいのです．メッセージを確認後はShift-JISに戻しましょう．

このように「関数'test'への引き数が少なすぎます」とエラーにしてくれます．

▶ 関数の書き方

ここで，蛇足ですが，関数の書き方について説明します．

【①】

```
int main(void){
        xxxxx......
        xxxxx......
}
```

リスト2　引き数が宣言と違う呼び出しをしている例(test2.c)

```
#include <stdio.h>
int test(int a,int b,int c);
int main(void)
{
        return test(1,2);
}
int test(int a,int b,int c)
{
        int aa;
        aa  =   a + b + c;
        printf("%d¥n",a);
        return aa;
}
```

【②】

```
int main(void)
{
        xxxxx......
        xxxxx......
}
```

{の位置が2種類見受けられると思いますが，ここでは②の方式に統一します．いろいろ異論はあると思いますが，このほうが{の始まりと終わりが明白になるからです．

● 予約語――ANSI-1989とC99の規定

C言語には，「予約語」というものが存在します．これは，プログラム中で自由に利用できない識別名のことを言います．たとえば，intという単語が予約語だとすると，intという名前を変数名や関数名として利用できないということです．

表1にANSI-1989で決定された従来の予約語を示します．そして，**表2**にC99規格で決定された，新しい予約語を示します．

予約語以外の識別名は自由に使用できます．ただし，標準の関数名などにはもちろん使用できません．

そのほか，予約語の制約は次のとおりです．

① 31文字以内であること
② 大文字，小文字は区別される
③ 半角英数字および半角アンダスコア(_)のみ使用でき，最初の1文字は英字である．大文字，小文字は区別される

表1　ANSI-1989で決定された予約語

auto	const	double	float	int	short	struct	unsigned
break	continue	else	for	long	signed	switch	void
case	default	enum	goto	register	sizeof	typedef	volatile
char	do	extern	if	return	static	union	while

表2　C99規格で決定された新しい予約語

名　称	対　象	説　明
inline	関数を修飾	高速に呼び出すようコンパイラに指示することが可能． アセンブラに展開した際に，この関数を呼び出した部分を関数の内部コードそのもので置き換える
restrict	ポインタを修飾	そのポインタの指す対象にエイリアスがないことを保証する指定． コンパイラはこの情報を元に最適化を行う
_Bool	型	値0か1を保持できる型．#include <stdbool.h>とすることでシンボルbool，true，falseが使えるようになる
_Complex/ _Imaginary	float型を修飾	Complex，_Imaginaryで，複素数演算をサポートする． #include <complex.h>とすることで，complex，imaginary，Iといったシンボルが使用可能になる． (double complex)(1.2+I * 5.4);のような複素数の表記，演算などが可能になる