第2章　はじめて読むプログラム

2.1　プログラムの構造

前節では、“こんにちは” などの文字を出力するだけという、もっとも簡単なプログラムを例としてプログラム作成の手順を学んだ。この節では、プログラムの書き方について勉強する。

main() のあと、必ず _____ （“{” と “}”）の間に、プログラムの本体を記述していく。

例題 2.1.1 ユーザから何かデータを入力してもらい、それに対して何らかの処理を行なって結果を出力するようなプログラムを作成する。

ファイル twice.c

```
#include <stdio.h>

main()
{
    int num;

    printf("あなたの好きな数を入力して下さい。 ");
    scanf("%d", &num);
    printf("あなたの好きな数の 2 倍は %d です。¥n", num * 2);

    return 0;
}
```

このプログラムは、ユーザから適当な数を入力してもらい、その数の *2* 倍を計算する単純なプログラムである。 網かけ の部分は、*C* プログラムの “お約束” のような部分で、これから出てくるプログラムのほとんどに共通する。

各行の最後の “;”（セミコロン）に注意する。セミコロンは _____を表す。C のプログラムは、このような文がいくつも並べられたものであり、文は基本的には、上から順に実行される。

2.2　変数

このプログラムは int num; という行から始まっている。これは _____である。

C の変数とは、データを入れておくための箱のようなものである。（数学の方程式中の変数のように未知数を表すものではない。）ユーザから入力されたデータや、処理の中間結果などを記録するために変数を使用する。上の例ではnum という変数を用いている。変数の名前には、後で述べる変数名の規則に従う限り好きな名前をつけることができるが、変数の役割がわかりやすいような名前をつけるようにする。

変数は使用する前に必ず**宣言**する必要がある。上の例では一番最初の`int num;` という部分が変数の宣言部分である。変数の宣言は “{” の直後に、他の処理を行なう前に行なっておく必要がある。`int` はこの`num` という変数の ___であり、この変数に入れることのできる**データの種類**（データのサイズ）を表す。`int` は _____（整数）の略であり、`int num;` は`num` は整数を入れるための箱であることを示している。

C ではしばしば、変数や式の型を意識する必要があり、型の指定を間違えるとわかりにくいバグになってしまうことが多い。C の基本型にはこの他に`char`（文字型）や `double`（浮動小数点数型）などがある。これらの型については、追々説明する。

また変数名を `,`（コンマ）で区切って、同時に複数の変数を宣言をすることができる。

```
int num1, num2;
```

この例では、`num1, num2` という 2 つの変数を宣言している。 変数宣言の最後には必ず `;`（セミコロン）が必要である。

変数名の規則

C の変数名には、次のような文字を使うことが許されている。

- アルファベット （`A`〜`Z, a`〜`z`）
- アンダースコア （`_`）
- 数字 （`0`〜`9`）

このうち、数字は先頭の文字としては使うことはできない。アンダースコアも先頭の文字に使わない方が良い。アルファベットの大文字と小文字は区別するので、`China` と `china` は別の変数として扱われる。長さはいくら長くても良い。（ただし、処理系によっては制限がある。） また、`if` や `while` など、C にとって特別な意味のある単語（_____、_____という）は変数名としては使えない。

（参考）C のキーワード

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

問 2.2.1 次の中で *C* の変数名として使えないものはどれか?（複数）
abc, _x, million$, Coca-Cola, __foo, 7up, o157,
can_we_use_such_a_very_long_name_like_this_for_variables

.....
.....

2.3 実行処理

2.3.1 C 言語での出力

C 言語で画面に出力を行なうときは次のような形を用いる。

```
printf(文字列);
```

前章でも述べたように、文字列は「"」と「"」に囲まれた部分である。

より一般的に、printf は次のような形式で用いる。

```
printf(文字列, 式1, 式2, ... );
```

基本的には printf は文字列をそのまま出力する。しかし文字列の中に% から始まる “変換指定” が現れると、式 1 以下に順に置き換えられる。つまり、printf の最初の文字列以外の式の数は、文字列中の変換指定の数になる。 変換指定は対応する式の型によっていろいろなものを使い分ける必要がある。

printf の変換指定には次のようなものがある。

%d	int 型の式に対応する。その数の 10 進法での表現を出力する
%x, %X	int 型の式に対応する。その数の 16 進法での表現を出力する
%c	int 型（ただし通常 char 型）の式に対応する。単一文字に置き換えられる
%s	char [] 型に対応する。 文字列に置き換えられる
%f	double 型に対応する。
%g	double 型に対応する。
%%	%そのものを出力する。（これは “変換指定” ではない）

その他の変換文字や、より細かな指定も可能である。詳しくはヘルプや参考書を参照のこと。

例えば、単に整数（int 型）を出力したい時は、

```
printf("%d", 式);
```

と書けば良いし、実数（double 型）を出力したい時は、

```
printf("%f", 式);
```

と書けば良い。また、

```
printf("%d 時 %d 分", 9, 10);
```

と書けば、最初の %d が ”9”に、次の %d が ”10”に置き換わって、”9 時 10 分”と出力される。

問 2.3.1 次のような、文字によるグラフィックスを *printf* を使って作成せよ。

```
K   K     A      GGGG     A     W   W     A
K  K     A A    G        A A    W   W    A A
KKK     AAAAA   G  GG   AAAAA   W   W   AAAAA
K  K    A   A   G   G   A   A   W W W   A   A
K   K   A   A    GGG    A   A    W W    A   A
```

問 2.3.2 %x と %X の違いについて調べよ。%f と %g の違いについて、いくつかの例で試せ。また、%2d や%02d、%02X という形の変換指定について調べよ。

.....

.....

.....

2.3.2　C 言語での入力

一方、キーボードから入力を受け取って変数に格納する場合は、

scanf(文字列, &変数 $_1$, &変数 $_2$, ...);

という形を用いる。変数の前の&は、アドレス演算子というものであるが、ここでは説明を省略する。

この形はprintf はとは逆に、入力された文字列を文字列中の変換指定に従って変換し、変数 $_1$ 以下に格納する。scanf の変換指定の意味はprintf とほぼ同じで以下のようになる。

- %d　int 型の変数に対応する。
- %c　char 型の変数に対応する。
- %lf　double 型の変数に対応する

つまり、キーボードから入力した整数をint 型の変数に格納する場合、

scanf("%d", &変数);

と書けば良いし、キーボードから入力した数をdouble 型の変数に格納する場合、

scanf("%lf", &変数);

と書けば良い。

例題 2.3.3 「*h:m*」形式の入力を 「*h* 時 *m* 分」に書き換えて出力する。

ファイル *scanf.c*

```
#include <stdio.h>

main()
{
    int h, m;
    scanf("%d:%d", &h, &m);
    printf("%d 時 %d 分です。¥n", h, m);

    return 0;
}
```

また、一文字だけを入力する時は次のような形を用いることもできる。

変数 = getchar ();

この時、変数はint 型のものを用いる。

例題 2.3.4 *getchar* の使用例

ファイル *getchar.c*

```
#include <stdio.h>

main()
{
    int c = getchar ();
    printf("あなたが入力した文字は%c です。¥n", c);

    return 0;
}
```

プログラムの最初の #include<stdio.h>は、実は、「このプログラムでは printf, scanf などの入出力用の命令を利用します。」という意味の宣言である。

2.3.3 変数への代入

例題 2.3.5 長方形の幅と高さを入力して、面積を出力するプログラムを作成する。

ファイル area.c

```
#include <stdio.h>

main()
{
    int width, height, area;

    printf("幅を入力して下さい。 ");
    scanf("%d", &width);
    printf("高さを入力して下さい。 ");
    scanf("%d", &height);
    area = width * height;
    printf("長方形の面積は %d です。¥n", area);

    return 0;
}
```

```
area = width * height;
```

という文は、width * height（幅×高さ）という**式**の値を、area という変数の中に入れている。このように変数に値を入れることを＿＿＿＿という。=は代入を指示する記号（演算子）である。（数学では、掛け算の記号×は省略できることがあるが、C 言語では *は必ず必要である。）

代入は、数学での等号 = と同じ記号だが、区別する必要がある。例えば次は正しい C の文である。

```
x = x + 1;
```

これは、この代入実行以降の変数 x の値を、代入実行までの変数 x の値に 1 を足したものにせよ、という意味になる。

また次のように変数を宣言する時に同時に初期値を代入することもできる。

```
double pi = 3.1416;
```

double は小数（浮動小数点数）のデータ型である。

問 2.3.6 三角形、円、台形などの面積を求めるプログラムを書け。
（ヒント: 小数を入れる可能性のある変数は *double* 型として宣言せよ。）

問 2.3.7 円からドルなどの通貨の変換やメートルからフィートへの計量単位の変換、摂氏から華氏への変換を行なうプログラムを書け。

2.3.4　関数呼出し

例題 2.3.8 入力された数の平方根を出力する。

ファイル sqrt.c

```
#include <stdio.h>
#include <math.h>

main()
{
    double num1, num2 ;

    printf("あなたの好きな数を入力して下さい。 ");
    scanf("%lf", &num1);
    num2 = sqrt(num1);
    printf("あなたの好きな数の平方根は %f です。¥n", num2);

    return 0;
}
```

このプログラムでは *sqrt*（***sq**uare **r**oo**t**,* 平方根, $\sqrt{\ }$）という関数を用いている。

関数とは、まとまった処理を行なうプログラムの一部を部品化して、なんどでも再利用できるようにしたものである。自分で関数を定義する方法は、後ほど紹介するが、ここではすでに用意されている関数の利用方法を説明する。

関数呼出しの一般形は次の通りである。

関数名 (引数 $_1$, 引数 $_2$, ... , 引数 $_n$)

引数（＿＿＿＿＿＿と読む）は、関数に渡すデータである。引数を変えることによって関数の振舞いを変えることができる。また関数から呼出し側にデータを返すこともできる。

関数が返すデータ（＿＿＿＿と呼ぶ）は、そのまま変数に代入することもできるし、他の関数の引数として用いることもできる。例えば

```
num2 = sqrt(num1);
```

では sqrt の返り値は、変数 num2 に代入されている。log(sqrt(num1)) や 1+sqrt(num1) では、他の関数（log）や演算子（+）の引数として使われている。

注意: sqrt や log などの数学関数を使う時は、プログラムの最初の方（#include < ... >という行が続いている後が良い）に

```
#include <math.h>
```

という一行を付け足す必要がある。（最後にセミコロンを入れては**いけない**。） math.h は、これらの関数の型を宣言（後述）しているファイルである。

問 2.3.9

1. ユーザから入力された数値に対して、*sin*, *cos*, *log*, *sqrt* などの数学関数（すべて *double* から *double* への関数である）の値を出力するプログラムを書け。
2. 直角三角形の *2* 辺の長さを受け取って、斜辺の長さを出力するプログラムを書け。

*3. 2*次方程式の係数を受け取って、その解（実数解と仮定して良い）を出力するプログラムを書け。

このように同じ“関数”という言葉を使っても、C 言語でいう関数は数学の関数とは少しニュアンスが異なる。C 言語でいう関数とは、あくまでも「まとまった処理を行なうプログラムの一部を部品化して、なんども再利用できるようにしたもの」である。他のプログラミング言語では「_____」、「_____」などと呼ばれることもある。

2.4 コメント

_____とはプログラム中に挿入するプログラマのメモ・覚え書きのことである。C 言語のコンパイラは、コメントは単なる空白として扱う。

C のコメントは、__ から __までの間

```
area = width * height;    /* 面積 = 幅×高さ */
```

に記述する。

適切なところにコメントを入れることが、メンテナンスしやすいプログラムを書くための大切なコツである。

キーワード:

文、変数、宣言、型、int、キーワード、stdio.h、printf、scanf、getchar、代入（=）、double、コメント（/*～*/）、関数、引数、戻り値、演算子、