配列

ここまでにデータを保存するための領域として int 型や double 型といった変数を利用してプログラムを作成してきた。これらの変数はデータを格納するために利用し、「データを出し入れする箱」として例えられる。

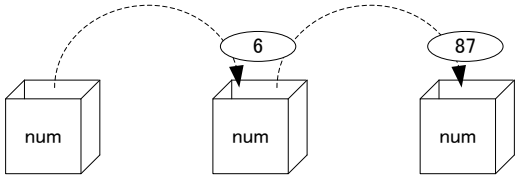

図１　変数のイメージ

プログラムを作成する際に変数が多数必要となった場合はどうなるだろうか。例えば同じようなデータを扱うために int 型の変数が 100 個必要となったとすると、変数の宣言は 100 個の変数名を記述しなくてはならない。これは非常に使い勝手が悪いので、多数の同じ型のデータを扱うために配列というデータ構造が用意されている。配列は for 文などの繰り返し処理と組み合わせることですっきりとした記述が可能となる。本日はこの配列の宣言と操作方法について説明する。

配列の宣言

同じ型のデータをまとめて管理する配列の役割と宣言方法について述べる。例えば 10 人の英語の点数を管理するプログラムについて、今までに習った int 型の変数で実現することを考えると以下のような宣言となる。

```
int a0, a1, a2, a3, a4, a5, a6, a7, a8, a9;
```

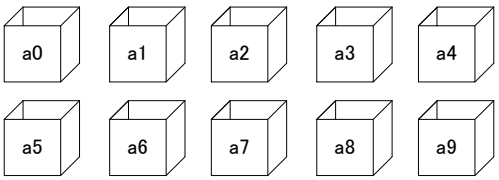

図２　多くの変数が定義された場合のイメージ

これでプログラムを作成するには、変数の宣言、値の代入の数が多いだけでなく、データ数が後から変更になった場合には、プログラム全体の見直しが必要になるため、かなりの手間がかかってしまう。

それでは、配列について説明していく。配列を使った場合でも、まずは宣言を行う必要があり、以下のように行う。

書式
データ型 配列名[個数];

以下に使用例として、int 型のデータで、「配列名」を a とし、個数を 10 として宣言したものを示す。

使用例
int a[10];

この場合、以下のように a[0]～a[9]といった「データを出し入れする連続した箱」が用意される。

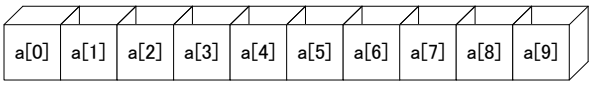

図３　配列のイメージ

この連続した箱の一つ一つは「配列の要素」と呼ばれる。この配列の要素は int 型変数を i を用いて、a[i]と書くこともできる。あるいは a[(int)(i/2)]のように式を書いて要素を指定することもできるが、実数を指定することはできない。

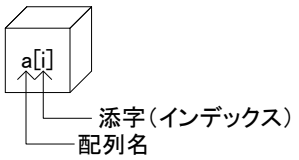

図４　配列の要素

※C 言語のコンパイラでは、配列の上限を超えた値、または 0 より小さい値を添字として利用してしまった場合、コンパイルエラーは出ない。ただし、多くの場合はプログラムが正常に動作しない。「segmentation fault」あるいは「セグメンテーション違反です」などのメッセージを出して、プログラムは止まるか想定外の動作をする。

配列は for 文と相性が良く、組み合わせてプログラムが作成することで、すっきりとした記述が可能である。以下は配列の要素にキーボードから値を代入し、その結果を出力するプログラムである。

プログラム例１

```
#include <stdio.h>

int main(void)
{
        int i, a[10];

        for(i=0; i<10; i++)
        {
                printf("値を入力してください:");
                scanf("%d", &a[i]);
        }

        printf("入力された値は以下の通りです¥n");
        for(i=0; i<10; i++)
        {
                printf("%d番目:%d¥n", i+1, a[i]);
        }
        return 0;
}
```

配列データの初期化

配列のデータを初期化する方法を述べる。上記で示した配列 a[0]、a[1]、…に英語の点数を代入するには、a[0] = 56 ; a[1] = 67 ; …というように代入文を 10 個も書かなくてはならない。配列には宣言時に値を代入する初期化の方法が用意されている。プログラム例を以下に示す。

プログラム例２

```
#include <stdio.h>

int main(void)
{
        int i;
        int a[10] = {56, 67, 99, 87, 63, 89, 90, 49, 77, 64};

        printf("初期化された配列の値は以下の通りです¥n");
        for(i=0; i<10; i++)
        {
                printf("%2d番目:%d¥n", i+1, a[i]);
        }
        return 0;
}
```

また、配列の宣言時に初期化データがあるときは[ ]の中に書く配列の要素数を省略することができる。

```
        int a[] = {56, 67, 99, 87, 63, 89, 90, 49, 77, 64};
```

マクロ定義（#define 文）

#define 文は、ある文字列を別の文字列で置き換えたいときに用いる。これを「マクロ定義」という。書式は以下のようにして用いる。

書式

```
#define 文字列1 文字列2
```

文字列１にあたるものを「記号定数」という。この記号定数名は区別しやすいように大文字で書く習慣がある。#define 文以降に出てきた文字列１は全て文字列２に置き換わる。

プログラム例３

```
#include <stdio.h>

#define NINZU 10

int main(void)
{
        int i;
        int a[NINZU];

        for(i=0; i<NINZU; i++)
        {
                printf("値を入力してください:");
                scanf("%d", &a[i]);
        }

        printf("入力された値は以下の通りです¥n");
        for(i=0; i<NINZU; i++)
        {
                printf("%d番目:%d¥n", i+1, a[i]);
        }
```

```
        return 0;
}
```

このプログラム例３はプログラム例１について#define 文で配列の要素数を宣言したものである。プログラム例１はこの要素数を他の入力や出力の処理にも使っているので、対応する部分を全て記号定数に変更している。このようにすると、データ数を変更する必要が出た場合（通常のソフトウェア作成時には良くある）には、#define 文のところで数値を変えるだけで正しく動作するプログラムに変更可能である。

※現在の例題や課題程度の長さのプログラムでは変更があってもたいした手間ではない。ある程度の長さのプログラムを作成する際には#define 文を使用しないと変更部分を探し出すだけでも大変になる。その際には#define 文を利用するのが良い。

演習

プログラム例１～３を作成し、実行しなさい。プログラム例３については#define 文の数値を変更し、正しく実行できるか確認すること。

課題 4

kadai4-1（30 点）

キーボードから配列に整数型のデータ 10 個を入力する。その後、入力した値の最大値、最小値、平均値を求めて、出力するプログラムを作成しなさい。ただし、平均値は実数で正しく表示されなくてはならない。作成できたら、#define 文を利用したプログラムに変更しなさい。変更したら、データ数を 5 個にして動作の確認をすること。小数点以下の値も正しく表示されているか確認すること.

kadai4-2（30 点）

サイコロを振ったときに、それぞれのサイコロの目に対して次の表のような得点が与えられるゲームを考える。

サイコロの目	1	2	3	4	5	6
得点	30	10	15	2	8	28

このゲームを２回繰り返したときの合計得点の表を、配列と繰り返し処理を利用して求め、表示するプログラムを作成しなさい。

以下に作成の手順を示す。

・サイコロの目に対する得点を整数型の１次元配列に格納する。
・for 文の 2 重ループを使って 2 つのサイコロの目の組み合わせとその合計点を表示する。
・出力例を以下に示す。

```
ーーーサイコロの得点表ーーー
[1 回目] [2 回目] [得点]
[  1  ] [  1  ]  [60]
[  1  ] [  2  ]  [40]
        :
```

kadai4-3（40 点）

ここまでに習ったものを利用し、自分で使用例を考えてオリジナルの処理を行うプログラムを作成しなさい。時間に余裕がある人は複数作成して提出してもよい。ただし、少なくとも一つは作成して提出すること。

このプログラムについては、どのような動作を行うものかを簡単でよいので説明を付けること。

kadai4-3 は習った内容を理解し、作成していれば 25 点。
以下は 40 点の例です。
（1）プログラムにある程度の行数がある。
（2）とても良く応用できている。
（3）複数のプログラムを作成している。