オブジェクト指向とゲームプログラミング

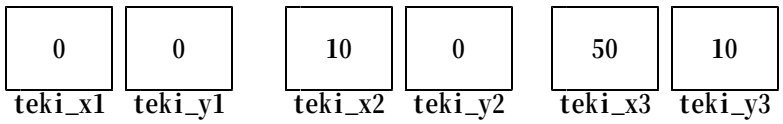
● 基礎編 － 第８回　配列 ●

配列

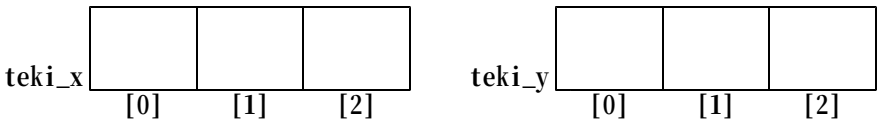
プログラムでは、データを変数に入れて扱います。１つのデータにつき１つの変数を宣言する必要があります。しかし、この方法ではプログラム作成上、困難になる場合があります。
たとえば、敵キャラクターの座標が、teki_x1, teki_y1, teki_x2, teki_y2…というint型の変数に格納されているとしましょう。

```
int   teki_x1 =  0, teki_y1 =  0;   // 敵キャラNo.1
int   teki_x2 = 10, teki_y2 =  0;   // 敵キャラNo.2
int   teki_x3 = 50, teki_y3 = 10;   // 敵キャラNo.3
```

これは、以下のように個々の変数を用意していることになります。



それでは、敵キャラクターの数が多くなったらどうでしょう。たとえば、敵キャラクターの数1000になったとしたら、teki_x1000とteki_y1000まで２０００個もの変数を定義しなければいけないとしたら、それはかなり面倒です。
ここで、配列の出番です。配列は、「個々の変数をひとかたまりにしたもの」といえます。いままでは、プログラムの中で値を保持するためには、必要な分だけ変数を作ってきました。
配列を使うと、同じ属性のデータをまとめて保持できるようになり、プログラムの煩雑さから解放されます。配列を図で表現すると、以下のように変数が連続しているイメージになります。上の図との違いは、変数が連続している点にあります。



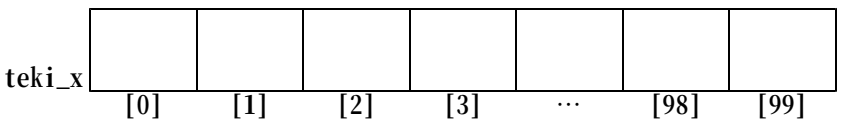
配列とは、まとめて箱を確保し、位置(先頭からの距離)を指定してアクセスできる変数なのです。

配列の定義

変数を使用するには定義が必要であったのと同じように、配列を使うためにも定義が必要です。

```
int   teki_x[100];
```

これは、teki_xという名前のint型の配列を定義しています。変数名の後に角括弧[]でくくって必要な個数を記述することで、配列の数(これを要素数といいます)を定義することができます。この例ではint型の１００個のデータが格納できる配列を定義しています。
配列を定義すると、メモリ上には以下の図のような変数の格納領域ができあがります。ちょうど、teki_x[0], teki_x[1], …, teki_x[99]という名前の１００個の変数が並んだ形になります。先頭がteki_x[1]ではなく、teki_x[0]になっていることに注意してください。



配列の定義の最初のintは、この配列がint型の変数を格納することを意味しています。もちろん、double型の配列やchar型の配列、そしてポインタや構造体などさまざまな変数を格納する配列を定義できます。
ただし、１つの配列を構成する要素は、すべて同じ型である必要があります。たとえば、１個目から３個目まではint型で、４個目から７個目まではdouble型にしたい、というような定義はできません。

配列を定義する際の括弧[]で囲まれた配列の要素数は、正の整数の定数式でなければなりません。小数や変数を指定することはできません。

```
int    array1[0.5];        // 小数はエラー

int    i = 100;
int    array2[i];          // iは変数なのでエラー

int    array3[89 / 3];     // 定数式なのでＯＫ
int    array4[0];          // 要素数が０の配列はエラー
```

配列の要素数は、プログラム実行中に変更することはできません。コンパイル時に決定されるため、プログラムの設計時によく考えておくことが必要です。

配列の添字

配列の個々の変数(これを配列の要素と呼びます)は、括弧[]を使ってアクセスすることができます。たとえば、前節の配列teki_xでは、teki_x[0]やteki_x[1]とすることで各要素にアクセスできます。この括弧でくくられた数字の部分を配列の添字と呼びます。Ｃ言語で配列の要素を扱う場合、次のことを注意してください。

配列の最初の要素は添字が０であり、最後の添字は「配列の要素数－１」である

配列の最初の要素には、添字を0にして(たとえば、teki_x[0])としてアクセスします。その次が1でアクセスします。0から始まることに注意してください。
１００個の要素数の場合、最後の１００番目の要素にアクセスするためには、添字を99とします。100にしてはいけません。0から始まるので、そこから100数えると最後は99になるわけです。また、添字には変数を使用することもとできます。たとえば、

```
int    array[100];
int    i = 9;
int    j = array[i];
```

というように変数を使ってアクセスすることもできます。

配列の値の参照と代入

配列の要素は、添字が必要ということを除けば、通常の変数と同じように使用することができます。たとえば、「int array[100]」として定義した場合、array[添字]はint型の変数として扱われます。array[50]に0を代入したければ、

```
array[50] = 0;
```

とします。配列は要素ごとにしかアクセスできません。=演算子である配列を別の配列にコピーしたり、+=演算子で「配列＋配列」などとすることはできません。

```
int    a[5], b[5];

a[0] = 1;
a[1] = 2;
a[2] = 3;
a[3] = 4;
a[4] = 5;

b  = a;       // 配列のコピー？→エラー
b += a;       // 配列の加算？  →エラー
```

このような場合、要素１つずつ処理を行う必要があります。

```
b[0] = a[0];
b[1] = a[1];
```

```
b[2] = a[2];
b[3] = a[3];
b[4] = a[4];
```

配列をまとめて処理する場合は、ループを使うと効率よいプログラムを作成することができます。また、配列のコピー程度の操作であれば、メモリ操作関数でも行うことができます。

ところで、前述の配列a，bをまちがえて

```
b[10] = a[10];
```

とした場合はどうなるのでしょうか。配列a，bともに要素数が５であり、10は存在しないのですが、C言語では配列の境界チェックを行わないため、コンパイルエラーにはならないのです。このようなプログラムでは、メモリを破壊しながら処理を続けてしまいます。まったく関係のない変数やプログラムの一部が壊されてしまうのです。
配列を使用するときは、範囲外にならないように、プログラム側で充分考慮する必要があります。

練習問題

１　int型の数値を１００個格納する配列arrayを作成しましょう。

２　配列arrayの最初の要素に１，最後の要素に１００を代入するプログラムを作成しましょう。

３　配列arrayのすべての要素に０を代入するプログラムを作成しましょう。

４　配列arrayの先頭から順に１，２，３…９８，９９，１００を代入するプログラムを作成しましょう。

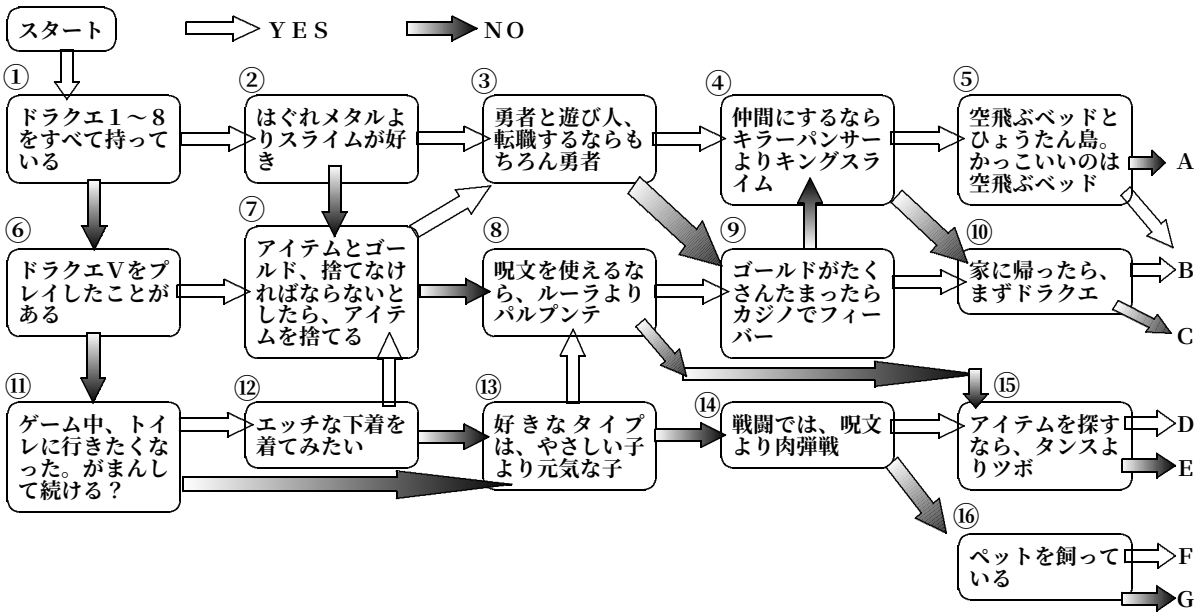
５　配列arrayのすべての要素の合計を求めるプログラムを作成しましょう。

６　配列arrayを配列array2(要素数１００)にコピーするプログラムを作成しましょう。

７　配列arrayのすべての要素に１～１００のランダムな数値を代入するプログラムを作成しましょう。

８　数値を入力させ、その数値が配列arrayに格納されているかどうかを検索するプログラムを作成しましょう。

9　メッセージボックスを用いて、以下のような相性診断を行うプログラムを作成しましょう。



A…主人公	主人公にたどり着いたあたなは、行動力大。ひとりでがんばれ
B…バーバラ	元気で明るいバーバラが、気弱なあなたををひっぱってくれるでしょう
C…チャモロ	いつもまじめなあなたは、チャモロと一緒に呪文を研究しましょう
D…ハッサン	自由気ままなハッサンが、あなたの力強い仲間になってくれるでしょう
E…テリー	クールなテリーと良い友達になれるでしょう
F…ミレーユ	やさしいお姉さんのミレーユに甘えてみたい、それが本心のようです
G…ファルシオン	決して戦わない、白馬のファルシオンと心が通じそう

ヒント

```
// 質問
char*   QUESTION[] = {"相性診断を始めます",                                                    // 質問0
                      "ドラクエ１～８をすべて持っている",                                      // 質問1
                      "はぐれメタルよりスライムが好き",                                        // 質問2
                      "勇者と遊び人、転職するならもちろん勇者",                                // 質問3
                      "仲間にするならキラーパンサーよりキングスライム",                        // 質問4
                      "空飛ぶベッドとひょうたん島。かっこいいのは空飛ぶベッド",                // 質問5
                      "ドラクエⅤをプレイしたことがある",                                      // 質問6
                      "アイテムとゴールド、捨てなければならないとしたら、アイテムを捨てる",  // 質問7
                      "呪文を使えるなら、ルーラよりパルプンテ",                                // 質問8
                      "ゴールドがたくさんたまったらカジノでフィーバー",                        // 質問9
                      "家に帰ったら、まずドラクエ",                                            // 質問10
                      "ゲーム中、トイレに行きたくなった。がまんして続ける？",                  // 質問11
                      "エッチな下着を着てみたい",                                              // 質問12
                      "好きなタイプは、やさしい子より元気な子",                                // 質問13
                      "戦闘では、呪文より肉弾戦",                                              // 質問14
                      "アイテムを探すなら、タンスよりツボ",                                    // 質問15
                      "ペットを飼っている"                                                     // 質問16
                     };
```

応用問題

１　配列arrayの要素を昇順(小さい方から大きい方への順)にソートするプログラムを作成しましょう。

ソート(整列)とは、データ群をある項目(キー)について、その昇順(小さい順)もしくは降順(大きい順)に並べ替えることです。基本的なソート法には、以下の３つがあります。

単純選択法
単純交換法(バブルソート)
単純挿入法

単純選択法は、非常に単純なアイディアに基づいています。数列全体を調べ、最小(または最大)の値を取り出し、先頭の数値と交換します。次に、先頭を除いた数列全体から最小の数値を取り出し、それを先頭の隣の数値と交換します。これを数列の最後まで続けるというものです。

単純交換法は、バブルソートとも呼ばれるソート法です。これも非常に単純なアイディアに基づいています。数列を先頭から隣どうしを比較し、逆順ならば交換します。数列の最後までこれを繰り返したら、また先頭にもどって同じことを続けます。交換するところがなくなったら終了です。要素が泡のように浮かび上がっていくことから、バブルソートと呼ばれています。

単純挿入法は、挿入ソートとも呼ばれるソート法です。ほぼ整列している数列に対しては非常に速いので、クイックソートでおおまかにソートして挿入ソートで整列を完成する方法がよく使われます。
挿入法では、比較範囲を限定して整列していきます。比較範囲を少しずつ増やしながら、全体を並べていきます。まず先頭の２つの要素を整列します。次に、範囲を１つ増やし、先頭の３つの要素を整列します。次はさらに範囲を１つ増やし、先頭の４つの要素を整列します。これを数列の最後まで続けるというものです。ここで重要なのは、すでに比較した範囲の大小関係は保証されているということです。整列済みの数列に要素を１つ挿入するように整列することから、挿入法と呼ばれています。

２　数値を入力させ、その数値が配列arrayに格納されているかどうかを効率よく検索するプログラムを作成しましょう。

検索とは、配列や表から、特定のキー(数値や文字列)をもつデータを探し出すことをいいます。探索には、

逐次探索法(シーケンシャルサーチ)
２分探索法(バイナリサーチ)

逐次探索法は、特に工夫のない探索法で、検索したい値が数列中に存在するかどうかを、数列の先頭から１つずつ比較しながら探していくというものです。

２分探索法は、ソートされた数列から高速に探索する方法です。まず、中央の要素と検索したい値を比較します。もし、検索したい値の方が大きければ、それは数列の右半分にあるはずです。また、もし検索したい値の方が小さければ、数列の左半分にあるはずです。このようにして、１回比較で候補の数が半分に減ります。以下同様にして、候補の数を次々に半分にしていけば、逐次探索法より高速に探索ができます。

３　１～１００を１回使ってできるランダムな順列を配列arrayに格納するプログラムを作成しましょう。

１～６のランダムな順列とは、２，４，１，６，５，３のような不規則に並んだ順列のことを言います。プログラムで再現するには、トランプをシャッフルするイメージになります。購入したばかりのトランプを上から順にめくると、当然順番にカードが登場します。しかし、適当にシャッフルすれば、上から順にめくっても、ランダムな順でカードが登場します。
プログラムでも同様の作業になります。配列に１から順番に数値を格納します。これが購入したばかりのトランプのイメージになります。これを先頭から取り出すと、当然１から順に数値が取り出されます。しかし、適当にシャッフルすれば、配列の先頭から順に数値を取り出しても、ランダムな数値が取り出されます。一般的に配列のシャッフルをする場合、先頭の要素とランダムな場所の要素を交換し、先頭の隣の要素とランダムな場所の要素を交換、これを配列の最後まで行うことで代用します。