基礎プログラミング演習 II 教材　(#14)

■ 文法落ち穂拾い：変数名と予約語

変数名として利用可能なのは以下のようなものです。

・最初の一文字は英字（大文字、小文字のいずれでも）ではじまること。
（なお C 言語では「 _ 」（アンダースコア）も英字として扱われる。high_score なども可。）
・二文字目以降は英字または数字が使える。
・変数名も大文字と小文字は区別される。
・予約語以外の名前でなければならない。

予約語とは int や while、return といった、C 言語の文法上、既に使われている単語のことです。以下に予約語の一覧を示します。

```
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
```

なお、C 言語では「 _ 」（下線、アンダースコア）は英字とされています。長い変数名をわかりやすくする場合などに使われます。工夫の例としては `leftbutton` とせず `left_button` あるいは `leftButton` などがあります。

注意：予約語ではないが衝突を避けたい名前
printf( ) などは C 言語の予約語ではなく「一般的な関数」として標準的に UNIX システムに用意されているものです。この名前と衝突するのも良くありませんので、注意しましょう。
こうした標準的なライブラリ関数の名前は多すぎてここには出しません。
変数名を決める際に、同じ名前のライブラリ関数があるかもしれない、と不安になった場合は man コマンドで確認すると良いでしょう。たとえば man 3 printf とすると printf( ) 関数の詳細なマニュアルが表示されるでしょう。

■ 配列の範囲

★教科書 p.104 6.1.4 「配列を用いる場合の注意点」（特に図 6.3 を見よ）

・添字の範囲には注意が必要
・用意した配列の要素数範囲に収める
（int array[5]; なら添字は 0〜4 まで）
・範囲を超える場合でもエラーは出ない
・コンパイルエラーは出ない
・実行時エラーは出る、かも、知れない
・偶然正常に動作してしまうかも知れない

```
  int x1, x2, a[5], y1, y2;

  x1=100; x2=101;
  y1=200; y2=201;
  a[0]=1;
  a[1]=2;
  a[2]=3;
  a[3]=4;
  a[4]=5;

  printf("x1=%d, x2=%d, y1=%d y2=%d\n",
x1, x2, y1, y2);
```

上のプログラムは正常に動作するはずです。しかし例えば a[5] に値を代入した場合、何が起きるか試してみると良いでしょう。添字を巨大な値にした場合はどうでしょう。マイナスにした場合何が起きるでしょう。
つまり C 言語ではこれはプログラマの責任範囲なのです。

■ 配列を利用する例

前回は「複数のボールを処理する」ために「普通の変数を複数並べた」ものとして配列変数を使い、その添字をループによって変化させることで「一つのボールを移動させる」処理で「複数のボールを移動」させました。
「プログラムは一度しか書かれていないが、ループによって複数回実行され、その処理対象を配列変数によって毎回変更した」といった構図が握めているでしょうか？

今回は配列を用いて良く行われる例を幾つか学びます。

□ 課題 1. ：要素の中を調べる

サンプルプログラム（下左、前回の「簡単な利用例」と同じ）を修正して、要素の内容をすべてプリントするのではなく、その中で最大、最小のものだけを表示させてください。

考え方（最大の値を見つける）：
・最大値を保持するための変数を用意し、その最初の値（初期値）をあり得る最小の値にする
・配列の全要素の値を順繰りにチェック
　・もし最大値変数の値より大きなものがあれば、それで最大値変数を置き換える
・全要素をチェックし終わった時の最大値変数の値が、全要素中最大の値である
・最小のものも同様。

なお、rand( ) 関数は 0〜2147483647 までの値を返します。rand( ) を 1000 で割った余りを使う、ということは 0〜999 の乱数を使う事を意味します。

□ #define による定数定義

下の二つのプログラムは同じ動作をするものです。右側のプログラムは頻出する重要な定数、10 を TIMES という文字で置き換えて記述したものです。`#define` は主に定数を定義するために利用します。`#define TIMES 10` とすることで、それ以降に登場する TIMES という単語（識別子）はそのままそっくり 10 と置き換えてコンパイルされます。（教科書 p.72 も参照）

```
#include <stdio.h>


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

  printf("random seed = ");
  scanf("%d",&seed);
  srand(seed);

  /* 各要素に 1000 未満の乱数を代入 */
  for(i=0; i<10 ; i++) {
    a[i]=rand()%1000;
  }

  /* 各要素をプリント */
  for(i=0; i<10; i++) {
    printf("a[%d] = %d\n", i, a[i]);
  }

  return 0;
}
```

```
#include <stdio.h>
#define TIMES 10

int main() {
  int i, a[TIMES], seed;

  printf("random seed = ");
  scanf("%d",&seed);
  srand(seed);

  /* 各要素に 1000 未満の乱数を代入 */
  for(i=0; i<TIMES; i++) {
    a[i]=rand()%1000;
  }

  /* 各要素をプリント */
  for(i=0; i<TIMES; i++) {
    printf("a[%d] = %d\n", i, a[i]);
  }

  return 0;
}
```

□ 課題 2. ：エラトステネスのふるい

★教科書 p.111 参照（特に図 6.7）
（アルゴリズムとコードの対応を、手作業でシミュレートしながら納得する）

押さえて欲しいポイント：
・アルゴリズム
・配列をマークするために使う、というアイディア（数を納めるのではなくマークを記録する）
・#define による定数の記述
・マークの初期化（初期状態を設定し、処理し、最後に状態を確認する、という処理モデル）
・（少しずつ範囲が狭くなる）二重ループ

アルゴリズムとデータ構造：「エラトステネスのふるい」は有名な「アルゴリズム」の一つです。その問題を解くための明確に定式化された手順、解法のことを指します。
これをコンピュータ上で実現するために必要な情報（データ、変数）をデータ構造と一般に呼び、つまり「アルゴリズムとデータ構造」を思いつくことができれば、それに沿ってプログラムは書けるはずであり、これがプログラミングの基本です。
課題に取り組む時に、アルゴリズムとデータ構造を意識できていますか？

□ 課題 3.

課題 1. のプログラムについて、前頁に示したサンプルのように #define の機能を使って試行する数を変更しやすいように修正してください。例えばプログラムの先頭に

```
#define TIMES 10
```

と書き、それ以降の必要な箇所をこの TIMES を利用して書き直して同じように動作することを確認してください。次にこれを 50 に変更してコンパイル・実行するだけで 50 回試行した結果の最大・最小値が得られることを確認してください。

□ 課題 4.

教科書 p.114 の演習問題 6.4 のプログラム（100 以下の数で入力した二数 x, y のいずれでも割り切れなかった数を表示せよ）を作成してください。
単純にループさせるだけでもできますが、配列を用意して、x や y で割れた数にマークをするようにして処理してください。

どのようなデータにどのようにマークすることで処理を実現するか、そのアルゴリズムとデータ構造は幾つも思い浮かぶと思います。