ソフトゼミＡ　第４回 解答と解説

■ 練習問題

1. 3 の倍数が入力されるまでループするプログラムを作れ。

```
#include <stdio.h>
int main( void ){
   int n;
   while( 1 ){
      scanf( "%d", &n );
      if( n % 3 == 0 ){ break; }
   }
   return 0;
}
```

while(1)で無限ループを作ることができます。「無限」とはいいつつも、「break」でループを抜けることができてしまいます。ちなみに、do-while を使うと以下のようになります。

```
#include <stdio.h>
int main( void ){
   int n;
   do{
      scanf( "%d", &n );
   } while( n % 3 != 0 );
   return 0;
}
```

どちらでもお好みの方でどうぞ。

2. 二重ループを使って九九を出力するプログラムを作れ 。

ここでは、九九の表を出力するプログラムを解説します。

```
#include <stdio.h>
int main( void ){
   int i, j;
```

(次ページへ続く)

```
  for( i = 1; i <= 9; i++ ){
    for( j = 1; j <= 9; j++ ){
      printf( "%3d", i * j );   /*この書き方については解説参照*/
    }
    printf("¥n");
  }
  return 0;
}
```

printf に「%3d」という書式指定文字を使いました。これは、整数を「長さ 3 で表示する」。ここでは「**整数を可能な限り 3 ケタで表示する**」という具合の書式指定文字です。3 ケタ以上の整数はそのまま出力され、2 ケタ以下の数字は 3 ケタになるように**左に半角スペースが詰められます**。

九九は最大(9*9=)81 ですから、2 ケタあれば答えを表示できます。しかし、隣のマスの数字とくっついてしまうのも見にくいですから、半角スペースを一つ入れるように 3 ケタで表示させるように指定してあります。

ちなみに、%3d を指定しない場合は printf("%3d", i * j);の部分を、以下のようにすると同じような出力結果が得られます。

※main 文の頭で「int ans;」を宣言してください。

```
ans = i * j;
if( ans < 10 ){
  printf("   %d", ans);     /*半角スペース 2 個と 1 桁の数字*/
}else{
  printf(" %d", ans);       /*半角スペース 1 個と 2 桁の数字*/
}
```

Output

```
 1  2  3  4  5  6  7  8  9
 2  4  6  8 10 12 14 16 18
 3  6  9 12 15 18 21 24 27
 4  8 12 16 20 24 28 32 36
 5 10 15 20 25 30 35 40 45
 6 12 18 24 30 36 42 48 54
 7 14 21 28 35 42 49 56 63
 8 16 24 32 40 48 56 64 72
 9 18 27 36 45 54 63 72 81
```

■ 追加練習問題

1. 1 から 100 までの整数を出力せよ。ただし、以下の条件を満たすこと。
   - 数字は 1 行に 1 つ出力せよ。
   - 3 の倍数の場合は、数字の代わりに＼（＾ｏ＾）／と出力せよ。
   - 数のどこかの桁に「3」がある場合は、数字の代わりに／（＾ｏ＾）＼と出力せよ。
   - 3 の倍数であり、かつ、数のどこかの桁に「3」がある場合は、(＾ｏ＾ ≡ ＾ｏ＾)と出力せよ。

```
#include <stdio.h>
int main( void ){
  int i;
  for( i = 1; i <= 100; i++ ){
      if(
          ( i % 10 == 3 ) ||
          ( i / 10 % 10 == 3 ) ||
          ( i / 100 % 100 == 3 )
      ) {
        if( i % 3 == 0 ){
          printf( "( ^o^ 三 ^o^ )¥n");
        }else{
          printf( "／(^o^)＼¥n");
        }
      }else if( i % 3 == 0 ){
        printf( "＼(^o^)／¥n" );
      }else{
        printf( "%d¥n", i );
      }
  }
  return 0;
}
```

最初の if 文の条件式があまりにも見にくくなってしまったので 3 行に分けました。このように、条件式の途中で改行することもできます。ちなみに、「i / 100 % 100 == 3」は i の 100 の位が 3 であるかどうかを判定するだけなので、あっても無くても構いません。(むしろ無い方が速いかも?)

最初の if 文で、100 の位, 10 の位, 1 の位のいずれかに 3 が含まれているかどうかを判定しています。それでいて、かつ 3 の倍数であれば「(^o^ 三 ^o^)」、3 の倍数でなければ

「／(^o^)＼」となります。100 の位, 10 の位, 1 の位のいずれにも 3 が含まれていない場合は、3 の倍数であれば「＼(^o^)／」、そうでない場合は普通の数字となります。

Output

1

2

(^o^ 三 ^o^)

4

5

(中略)

95

＼(^o^)／

97

98

＼(^o^)／

100

2. でかい X を出力せよ。(問題の詳細は追加問題資料を参照のこと)

```
#include <stdio.h>

int main( void ){
   int i, j, n;
   while( 1 ){
      scanf( "%d", &n );
      if( n <= 0 ){ break; }
      else if( n == 1 ){ printf("X¥n"); }
      else{
         for( i = 0; i < n; i++ ){
            for( j = 0; j < n; j++ ){
               if( i == j || i + j == n - 1 ){
                  printf( "■" );
               }else{
                  printf( "　" );
               }
            }
```

```
            printf("¥n");
        }
      }
    }
    return 0;
}
```

i と j で二重の for 文を回します。■になっている部分は、n = 5 の時、以下のようになります。(数字は順に i, j)

0,0	0,1	0,2	0,3	0,4
1,0	1,1	1,2	1,3	1,4
2,0	2,1	2,2	2,3	2,4
3,0	3,1	3,2	3,3	3,4
4,0	4,1	4,2	4,3	4,4

左上から右下に降りる対角線は「i = j」左下から右上に上がる対角線は「i + j = 4」を満たしています。一般にに、左下から右上に上がる対角線は「i + j = n - 1」を満たしています。

Sample Input

3 1 4 1 5

Sample Output

■ ■
  ■
■ ■
X
■ ■
  ■■
  ■■
■ ■
X
■ ■
  ■ ■
    ■
  ■ ■
■ ■

3. 整数 n を読み取り、1+2+ … + n を求めよ。

```
#include <stdio.h>
int main( void ){
   int n;
   scanf( "%d", &n );
   printf( "%d¥n", ( 1 + n ) * n / 2 );
   return 0;
}
```

解説は第 1 回の追加問題とほぼ同じなので、そちらを参照してください。for 文の練習のために、あえて等差数列の和の公式を使わずに全部足しても構いません。

for 文を使うと以下のようになります。

```
#include <stdio.h>
int main( void ){
   int sum = 0;
   int n, i;
   scanf( "%d", &n );
   for( i = 1; i <= n; i++ ){
      sum += i;
   }
   printf( "%d¥n", sum );
   return 0;
}
```

Sample Input

9999

Sample Output

49995000

4. 100000 までの素数をすべて表示せよ。

```
#include <stdio.h>
int main( void ){
   int i, j;
   for( i = 2; i < 100000; i++ ){
      for( j = 2; j < i; j++ ){
         if( i % j == 0 ){ break; }
```

(次ページへ続く)

```
    }
      if( i == j ){ printf("%d¥n", i ); }
  }
  return 0;
}
```

外側のループ(変数が i の方)で、2 から 100000 までの全ての整数に対して素数判定を行うようにループさせています。内側のループでは、その整数(i)が 2 以上 i 未満のいかなる整数でも割りきれないということを検証しています。最後に、すべての値のチェックが済んだ場合(その数が素数と確定した時)にのみ数字が出力されるようになっています。

Output

2

3

5

(中略)

99989

99991

5. 以下の値を求めよ。

$$1+\frac{1}{1!}+\frac{1}{2!}+\frac{1}{3!}+\cdots+\frac{1}{10!}$$

```
#include <stdio.h>
int main( void ){
  int i, j, fi;
  double inv;
  double sum = 1;
  for( i = 1; i <= 10; i++ ){
    fi = 1;
    for( j = 1; j <= i; j++ ){
      fi *= j;
    }
    inv = 1.0 / fi;   /*  注意！  */
    sum += inv;
```

(次ページへ続く)

```
  }
  printf( "%f¥n", sum );
  return 0;
}
```

内側のループ(変数 j)では階乗計算(n! = n * (n-1) * … * 2 * 1)を行っています。fi には i!の値が入っています。次に、1 / fi の値を計算するのですが、ここで**注意！**ここで「1/fi」とすると、int 型÷int 型とみなされて、inv には小数点以下が切り捨てられた値が入ってしまいます。これを回避するために、1.0 / fi とすると double 型÷int 型とみなされて、inv には正しい double 型の値が代入されます。

Output

2.718282

あれ、この値見たことありません？実は、この数式は e^x のマクローリン展開の x = 1 の場合、すなわち自然対数の底 e の近似値だったのですよ。1/10! ともなればものすごく微小な値ですから、かなり正確に求まってしまいます。

6. 素数判定アルゴリズムをより早く終わるように改良せよ。
   完成したら n = 100000 の場合で試してみよ。

```
#include <stdio.h>

int main( void ){
  int i, j;
  int isPrime;
  printf( "2¥n");
  for( i = 3; i < 100000; i += 2 ){
    isPrime = 1;
    for( j = 2; j * j <= i; j++ ){ /*重要*/
      if( i % j == 0 ){
        isPrime = 0;
        break;
      }
    }
    if( isPrime ){ printf( "%d¥n", i ); }
  }
  return 0;
}
```

あくまでも上の例は一例に過ぎません。いろいろな方法があります。

重要なのは、j は i まで調べなくてもよく、√iまで調べれば十分素数と判定できるということです。この例では for 文の条件を j * j <= i としています。

なぜこれで良いのか。整数 i を i = x * y という 2 つの数の積にしてみた時に、$i = \sqrt{i} * \sqrt{i}$ 以降は x と y が逆転するだけで、同じ組み合わせを 2 回見に行くことになります。例えば、97 を素数かどうか判定する時に、以下の作業の内、灰色で塗った部分は無駄なものとなります。

（以降、循環小数や無理数は随時値を切り捨てています。）

97 を 2 で割り切れるかどうか -> 割り切れない 97 = 2 * 47.500
97 を 3 で割り切れるかどうか -> 割り切れない 97 = 3 * 32.333
97 を 4 で割り切れるかどうか -> 割り切れない 97 = 4 * 24.250
97 を 5 で割り切れるかどうか -> 割り切れない 97 = 5 * 19.400
97 を 6 で割り切れるかどうか -> 割り切れない 97 = 6 * 16.167
97 を 7 で割り切れるかどうか -> 割り切れない 97 = 7 * 13.857
97 を 8 で割り切れるかどうか -> 割り切れない 97 = 8 * 12.125
97 を 9 で割り切れるかどうか -> 割り切れない 97 = 9 * 10.778
97 を 10 で割り切れるかどうか-> 割り切れない 97 = 10 * 9.700
97 を 11 で割り切れるかどうか -> 割り切れない 97 = 11 * 8.818
…
97 を 96 で割り切れるかどうか -> 割り切れない 97 = 96 * 1.010

$$\sqrt{97} = 9.849$$

i= x * y の **x と y どちらも整数**になるなら i は x で割り切れ、また、i は y で割り切れ、この時点で**素数でない**ことは確定します。x = 2〜9 の時点でそれを満たす y は存在しないということは、x が 10 以上になった時は、x = 2〜9 で相棒の y がいなかったのと同じで、いかなる整数値の y と巡りあうことはできないのです。もし、素数でないなら、√iまでの間で、両方整数値となる x と y が存在して、また、√i から i までの間に x と y が逆になって再登場します。

※数学的に厳密な証明は数学科の方に聞いてください。

最後に、解答と解説の掲載が遅れましたこと、ならびにプリンタの不調により資料の印刷が不十分であったことをお詫び申し上げます。