【課題】

★ exercise-06-1 次の空欄を埋めなさい.

ユークリッドの互除法を用いて $a = 40920$, $b = 24140$ の最大公約数を求めよう.

$$\boxed{40920} = \boxed{24140} \times \square + \square$$
$$\square = \square \times \square + \square$$
$$\square = \square \times \square + \square$$
$$\square = \square \times \square + \square$$
$$\square = \square \times \square + \square$$
$$\square = \square \times \square + \square$$
$$\square = \square \times \square + \square$$
$$\square = \square \times \square + \square$$
$$\square = \square \times \square + \square$$

したがって $\gcd(40920, 24140) = \square$ である.

★ exercise-06-2 ２つの正の整数の最大公約数を求めるプログラムを「ユークリッドの互除法」を用いて書きなさい.

★ exercise-06-3 次の空欄を埋めなさい.

拡張されたユークリッドの互除法を用いて $a = 40920$, $b = 24140$ に対して,

$$ax + by = \gcd(a, b)$$

をみたす (x, y) を一組求めよう.

$$(\boxed{40920}, \boxed{1}, \boxed{0}) = \square \times (\boxed{24140}, \boxed{0}, \boxed{1}) - (\square, \square, \square)$$
$$(\square, \square, \square) = \square \times (\square, \square, \square) - (\square, \square, \square)$$
$$(\square, \square, \square) = \square \times (\square, \square, \square) - (\square, \square, \square)$$
$$(\square, \square, \square) = \square \times (\square, \square, \square) - (\square, \square, \square)$$
$$(\square, \square, \square) = \square \times (\square, \square, \square) - (\square, \square, \square)$$
$$(\square, \square, \square) = \square \times (\square, \square, \square) - (\square, \square, \square)$$
$$(\square, \square, \square) = \square \times (\square, \square, \square) - (\square, \square, \square)$$
$$(\square, \square, \square) = \square \times (\square, \square, \square) - (\square, \square, \square)$$
$$(\square, \square, \square) = \square \times (\square, \square, \square) - (\square, \square, \square)$$

したがって

$$40920 \times \square + 24140 \times \square = \gcd(40920, 24140)$$

が成り立つ. また, 他の (x, y) の組は

$$40920 \times \square + 24140 \times \square = 0$$

であることより,

$$x = \square + n \times \square$$
$$y = \square + n \times \square$$

であらわすことができる.

★ exercise-06-4 ２つの正の整数 a, b に対して, 拡張されたユークリッドの互除法を用いて

$$ax + by = \gcd(a, b)$$

をみたす (x, y) を一組求めるプログラムを書きなさい. ただし, $xy \neq 0$ をみたすこと.

★ exercise-06-5 次の空欄を埋めなさい.

２進互除法を用いて $a = 40920$, $b = 24140$ の最大公約数を求めよう.

ステップ１ $a = 40920$, $b = 24140$ のどちらか一方が奇数になるまで両方を 2 で割ると, $k = \square$ 回割れて, $a = \square$, $b = \square$ となる.

ステップ２ 必要なら a, b がともに奇数になるまで 2 で割り, a, b の大きい方から小さい方を引くことを繰り返すと,

$$\square - \square = \square$$
$$\square - \square = \square$$
$$\square - \square = \square$$
$$\square - \square = \square$$
$$\square - \square = \square$$
$$\square - \square = \square$$
$$\square - \square = \square$$
$$\square - \square = \square$$

となる.

ステップ３ したがって $\gcd(40920, 24140) = \square \times \square = \square$ である.

★ exercise-06-6 **（やや難？）** ２つの正の整数の最大公約数を求めるプログラムを「２進互除法」を用いて書きなさい.

★ exercise-06-7 **（難）** ２つの正の整数 a, b に対して

$$ax + by = \gcd(a, b)$$

をみたす (x, y) を一組求めるプログラムを「２進互叙法」を利用して書きなさい. ただし, $xy \neq 0$ をみたすこと.

★ exercise-06-8 **（難）** 上の問題 (exercies-06-7) で用いたアルゴリズムの正当性を証明しなさい.

★ exercise-05-9 ２つの正の整数の最小公倍数を求めるプログラムを書きなさい.

● 前回の講義のキーポイント及び補足

- 負の整数の表現方法として２の補数による表現を用いると, 加算回路を用いて減算を行うことが可能となる.
- 計算機内部の「実数」は「(２進) 浮動小数点数」であらわさる. C言語においては double 型（または float 型）であらわされる.
- 浮動小数点数の「仮数部」の桁数は有限であるため, （たとえば）１０進数の 0.1 は有限桁の２進浮動小数点型では正確な値をあらわすことはできなない. したがって（たとえば） 0.1 を 10 回加算しても 1.0 と等しいとは限らない.
- 整数の（int 型の） 1 と浮動小数点数の（double 型の） 1.0 とは, メモリ内部における表現が全く異なる.
- 算術演算においては次の事実が成り立つ.
  - 両方の被演算数が同じ整数型であるときには, 結果の型もその整数型となる.
  - 両方の被演算数が同じ浮動小数点型であるときには, 結果の型もその浮動小数点型となる.
  - 片方の被演算数が int 型であり, もう片方の被演算数が double 型であるとき, 演算が行われる前に int 型の数値が double 型に変換される. これを「算術変換」と呼ぶ. （算術変換の詳しい挙動は K&R A6 を参照）
- 論理 AND 演算子, 論理 OR 演算子は左オペランドを先に評価する. （他の演算子のオペランドの評価順序は不定である）論理 AND 演算子, 論理 OR 演算子は左オペランドを評価した時点で結果の値が確定するときには, 右オペランドの評価は行わない.

  論理 AND/OR とビット AND/OR の挙動の違いを示すプログラム例は次のようなものである. （これは久保氏に教えてもらった）

```
/* $Id: loginal_bit_and.c,v 1.2 2004-05-15 15:41:42+09 naito Exp $ */
#include <stdio.h>
int     i, j, count ;
int main(int argc, char **argv)
{
  i = 0 ; j = 0 ; count = 0 ;
  while(i++<10 |  j++<10) count += 1 ;
  printf("i = %d, j = %d, count = %d\n", i,j,count) ;
  i = 0 ; j = 0 ; count = 0 ;
  while(i++<10 || j++<10) count += 1 ;
  printf("i = %d, j = %d, count = %d\n", i,j,count) ;
  i = 0 ; j = 0 ; count = 0 ;
  while(i++<10 &  j++<10) count += 1 ;
  printf("i = %d, j = %d, count = %d\n", i,j,count) ;
  i = 0 ; j = 0 ; count = 0 ;
  while(i++<10 && j++<10) count += 1 ;
  printf("i = %d, j = %d, count = %d\n", i,j,count) ;
  return 0 ;
}
```


● 前回の課題の解説

★ exercise-05-1 西暦 1900 年から西暦 2004 年までの閏年をすべて出力するプログラムを書きなさい.

```
/* $Id: exercise-05-1.c,v 1.2 2004-05-12 16:22:22+09 naito Exp $ */

#include <stdio.h>

#define START   1900
#define END     2004

int main(int argc, char **argv)
{
  int   year ;

  for(year = START; year <= END; year++) {
    if (((year%4 == 0)&&(year%100 != 0))||(year%400 == 0))
      printf("%d\n", year) ;
  }
  return 0 ;
}
```

- 閏年の条件は「西暦が４の倍数である中で, １００の倍数にはならない年. ただし, ４００の倍数の時には閏年とする」というものである. 条件が「独立」ではないため, 種々の書き方が考えられる.
- 上のプログラム例は, 閏年になる条件を「素直」に書いたものである. わかりやすくするため「括弧」が少々冗長になっている.
- 西暦を表す変数は n, i などの単純な識別子名ではなく, year などの「意味のはっきりした」識別子名を使うことが望ましい.
- 他の条件判断例には以下のようなものがある.

  ▼ 誤りではないもの

  - ((year%100 != 0)&&(year%4 == 0))||(year%400 == 0)
  - (year%400 == 0)||((year%100 != 0)&&(year%4 == 0))
  - (year%400 == 0)||((year%4 == 0)&&(year%100 != 0))

  これらは間違ってはいない. しかし, プログラムを書く場合には主に次の２つの考え方がある.

  1. プログラムをわかりやすく書く.
  2. 高速に動作するプログラムを書く.

  この２つの考え方は相反する場合があり, （極端な言い方をすれば）閏年の判定もその一種である.

  1. 「プログラムをわかりやすく書く」立場にたてば, 「閏年」の条件をそのまま条件文に記述するのが望ましい. プログラム例の
     ((year%4 == 0)&&(year%100 != 0))||(year%400 == 0)

は「条件」をそのまま記述したものである. 一方, 他の例は条件の記述順序が異なっている. この程度の単純な（３つ程度の）条件の時にはどれでもすぐに意味をとることができるが, 条件が複雑になった場合には, 「条件を簡潔に, かつわかりやすく」書くことが必要となる.

2. 「高速に動作するプログラムを書く」立場にたってみよう. この時は, それぞれの条件文に対して除算が何回実行されるかを調べる必要がある. ここでは 1601 から 2000 までに対して除算の合計回数を調べてみよう.

	year	出現数	(a)	(b)	(c)	(d)	(e)	(f)
1	4 の倍数でない	1200	2	3	3	2	2	2
2	4 の倍数で 100 の倍数でない	396	2	2	3	3	3	2
3	100 の倍数で 400 の倍数でない	3	3	2	2	3	3	3
4	400 の倍数	1	3	2	1	1	1	3
合計		1600	3207	4402	4797	3601	3601	3207

(a) `((year%4 == 0)&&(year%100 != 0))||(year%400 == 0)`

(b) `((year%100 != 0)&&(year%4 == 0))||(year%400 == 0)`

(c) `(year%400 == 0)||((year%100 != 0)&&(year%4 == 0))`

(d) `(year%400 == 0)||((year%4 == 0)&&(year%100 != 0))`

(e) `if (year%400 == 0)...; else if ((year%4 == 0)&&(year%100 != 0))...`

(f) `if ((year%4 == 0)&&(year%100 != 0))...; else if (year%400 == 0)...`

▼ 誤りだと思われるもの

- `((year%4 == 0)&(year%100 != 0))|(year%400 == 0)`
  論理 AND/OR ではなくビット AND/OR を使うと, 条件判断を行うための式の値としては等価となるが, 計算に無駄が多いことがわかる. 実際, year = 1601 から 2000 に対して $1600 \times 4 = 6400$ 回の除算が行われる.
- `(((is_leap = year%4) == 0)&&(is_leap%25 != 0))||(is_leap%100 == 0)`
  これは, 一見「格好よく」見えるのだが, “25”, “100” という数字の意味が一見してはわからない. これは「プログラムをわかりやすく書く」ことに反している. 閏年の判定の場合, year は高々（１０進）４桁であり, 100 で割っても 400 で割ってもその計算速度に大きな違いはない. しかし, 極めて長い桁の数値に対する演算の場合には, このような工夫は必要かもしれない.

▼ 何とも言えないけど「ちょっとねぇ」と思うもの

`if (year%400 == 0)...; else if ((year%4 == 0)&&(year%100 != 0))...`
`if ((year%4 == 0)&&(year%100 != 0))...; else if (year%400 == 0)...`
閏年の判定を（後に講義で出てくる）関数として実現し, 関数内部でこれを書くのであれば, 関数内部は一種の「ブラックボックス」と思えるので, 悪くはないのだと思うが, いきなりプログラム中にこのような条件があらわれると, 一体何を判定しているのかよくわからない. 少なくとも, この if 文の主旨がわかりやすいとは言えない. また, ここの条件判断における除算回数は上を参照のこと.

★ exercise-05-2 ２つの int 型の整数の和を計算するプログラムを書きなさい.

```
/* $Id: exercise-05-2.c,v 1.9 2004-05-15 14:53:47+09 naito Exp $ */
#include <stdio.h>

int             a, b, c, sum ;
unsigned int    n ;

int main(int argc, char **argv)
{
  a = 0xab123 ; b = 0x00ffffff ;
  sum = c = 0 ;

  for(n=1;n;n<<=1) {
    sum |= (a^b^c)&n ;
    c = (((a&b)|(b&c)|(c&a))&n)<<1 ;
  }
  printf("%d + %d = %d", a,b,sum) ;
  return 0 ;
}
```

「インクリメント」を使うことを認めれば, 次のような書き方もある.

```
/* $Id: exercise-05-2-1.c,v 1.4 2004-05-15 14:54:02+09 naito Exp $ */
#include <stdio.h>
#define INT_BITS        32

int             a, b, c, sum, n ;

int main(int argc, char **argv)
{
  a = 0xab123 ; b = 0x00ffffff ;
  sum = c = 0 ;

  for(n=0;n<INT_BITS;n++) {
    sum |= (a^b^c)&(1<<n) ;
    c = (((a&b)|(b&c)|(c&a))&(1<<n))<<1 ;
  }
  printf("%d + %d = %d\n", a,b,sum) ;
  return 0 ;
}
```