C 言語の基本演算子（＋、－、＊、／、％）

１．scanf()関数による標準入力と基本演算子

1. 例題 balance.c

2. １２３４円の買い物をして１万円札を出したときの、おつりの札と硬貨の
の枚数を求めるプログラムを作成せよ。

2.1. scan()関数を用いて、価格と支払い金額を入力せよ。

＊ソースコード＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊

```
/*
  Program    : balance.c
  Student-ID : 075739A
  Author     : TSUHAKO,Masaki
  UpDate     : 2005/05/16(Wed)
  Comments   : Payment & Balance
*/

#include <stdio.h>

int main()
{
  int price = 1234, pay = 10000;
  int balance, amount;

  /****************************** scanf */
  printf("Price?   => "); scanf("%d",&price);
  printf("Payment? => "); scanf("%d",&pay);
  printf("----\n");

  /****************************** balance */
  balance = pay - price;
  printf("price = %d, ",price);
  printf("pay = %d, balance = %d\n",pay,balance);
  printf("----\n");
```

```
/****************************** 10000-yen */
amount  = balance / 10000;
balance = balance % 10000;
printf("10000-yen note = %d\n",amount);

/****************************** 5000-yen */
amount  = balance / 5000;
balance = balance % 5000;
printf("5000-yen note = %d\n",amount);

/****************************** 1000-yen */
amount  = balance / 1000;
balance = balance % 1000;
printf("1000-yen note = %d\n",amount);

/****************************** 500-yen */
amount  = balance / 500;
balance = balance % 500;
printf("500-yen coin = %d\n",amount);

/****************************** 100-yen */
amount  = balance / 100;
balance = balance % 100;
printf("100-yen coin = %d\n",amount);

/****************************** 50-yen */
amount  = balance / 50;
balance = balance % 50;
printf("50-yen coin = %d\n",amount);

/****************************** 10-yen */
amount  = balance / 10;
balance = balance % 10;
```

```
  printf("10-yen coin = %d\n",amount);

  /****************************** 5-yen */
  amount  = balance / 5;
  balance = balance % 5;
  printf("5-yen coin = %d\n",amount);

  /****************************** 1-yen */
  amount  = balance / 1;
  balance = balance % 1;
  printf("1-yen coin = %d\n",amount);

  return(0);
}
```

＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊ソースコード終了＊＊＊＊＊＊＊

出力結果：

```
Price?   => 1234
Payment? => 10000
----
price = 1234, pay = 10000, balance = 8766
----
10000-yen note = 0
5000-yen note = 1
1000-yen note = 3
500-yen coin = 1
100-yen coin = 2
50-yen coin = 1
10-yen coin = 1
5-yen coin = 1
1-yen coin = 1
```

2.2. 例題の変数名を変え、自分自身で変えた変数名にせよ。

現実的な名前にしてみる。

```
#include <stdio.h>

int main()
{
  int price = 1234, pay = 10000;
  int balance, amount;

  /****************************** scanf */
  printf("Your cost of living?   => "); scanf("%d",&price);
  printf("Your salary? => "); scanf("%d",&pay);
  printf("----\n");

  /****************************** balance */
  balance = pay - price;
  printf("Your cost of living? = %d, ",price);
  printf("salary = %d, saving money = %d\n",pay,balance);
  printf("----\n");

  /****************************** 10000-yen */
  amount  = balance / 10000;
  balance = balance % 10000;
  printf("10000-yen note = %d\n",amount);

  /****************************** 5000-yen */
  amount  = balance / 5000;
  balance = balance % 5000;
  printf("5000-yen note = %d\n",amount);

  /****************************** 1000-yen */
  amount  = balance / 1000;
  balance = balance % 1000;
```

```
printf("1000-yen note = %d\n",amount);

/****************************** 500-yen */
amount  = balance / 500;
balance = balance % 500;
printf("500-yen coin = %d\n",amount);

/****************************** 100-yen */
amount  = balance / 100;
balance = balance % 100;
printf("100-yen coin = %d\n",amount);

/****************************** 50-yen */
amount  = balance / 50;
balance = balance % 50;
printf("50-yen coin = %d\n",amount);

/****************************** 10-yen */
amount  = balance / 10;
balance = balance % 10;
printf("10-yen coin = %d\n",amount);

/****************************** 5-yen */
amount  = balance / 5;
balance = balance % 5;
printf("5-yen coin = %d\n",amount);

/****************************** 1-yen */
amount  = balance / 1;
balance = balance % 1;
printf("1-yen coin = %d\n",amount);

return(0);
```

```
}
```

出力結果：

```
Your cost of living?   => 1234
Your salary? => 10000
----
Your cost of living? = 1234, salary = 10000, saving money = 8766
----
10000-yen note = 0
5000-yen note = 1
1000-yen note = 3
500-yen coin = 1
100-yen coin = 2
50-yen coin = 1
10-yen coin = 1
5-yen coin = 1
1-yen coin = 1
```

変更完了。福沢諭吉が一人もいないのはさびしい。

2.3 工夫!

見やすいように（プログラムが長すぎるので）いらない部分を消去する。

```
#include <stdio.h>

int main()
{
  int price = 1234, pay = 10000;
  int balance, amount;
  printf("Your cost of living?   => "); scanf("%d",&price);
  printf("Your salary? => "); scanf("%d",&pay);
  balance = pay - price;
  printf("Your cost of living? = %d, ",price);
  printf("salary = %d, saving money = %d\n",pay,balance);
  amount  = balance / 10000;balance = balance % 10000;
  printf("10000-yen note = %d\n",amount);
```

```
  amount  = balance / 5000;balance = balance % 5000;
  printf("5000-yen note = %d\n",amount);
  amount  = balance / 1000;balance = balance % 1000;
  printf("1000-yen note = %d\n",amount);
  amount  = balance / 500;balance = balance % 500;
  printf("500-yen coin = %d\n",amount);
  amount  = balance / 100;balance = balance % 100;
  printf("100-yen note = %d\n",amount);
  amount  = balance / 50;balance = balance % 50;
  printf("50-yen coin = %d\n",amount);
  amount  = balance / 10;balance = balance % 10;
  printf("10-yen coin = %d\n",amount);
  amount  = balance / 5;balance = balance % 5;
  printf("5-yen coin = %d\n",amount);
  amount  = balance / 1;balance = balance % 1;
  printf("1-yen coin = %d\n",amount);
  return(0);
}
```

出力結果：

```
Your cost of living?   => 1234
Your salary? => 10000
Your cost of living? = 1234, salary = 10000, saving money = 8766
10000-yen note = 0
5000-yen note = 1
1000-yen note = 3
500-yen coin = 1
100-yen note = 2
50-yen coin = 1
10-yen coin = 1
5-yen coin = 1
1-yen coin = 1
```

問題なく作動。しかし、短くはなったものの、コメントがなくなったことで何がなんの出力に関係しているのかが、非常に分かりにくい。よってこの実験から分かったことは、『プログラミ

ングは分かりやすさが一番。』ということ。コメントはまめにいれたほうがいい。

3.int 型整数の下限・上限の値について、簡単なプログラムと実行結果を示し考察せよ。
（テキスト PP.68 基数 16 の表記法を用いたプログラムを考えること。）

考察：

int 型整数の上限と下限について今から調べるのだが、その前に、資料からコンピュータはデータを「0」と「1」の数字の組み合わせで記憶したり通信していることがわかった。文字もこの 0 と 1 の組み合わせ（2 進数）で表されている。ということは、コンピュータが 10 進数の数字を 2 進数で表せられなくなった時に int 型整数の上限・下限がわかるはずである。

次の事を考える。

a. 0 と 1 しかないので 1 の次はくりあがる。要するに我々が生活の中で使っている 10 進数の 1→2→3→4 という順は、2 進数では 00→01→10→11 という順になる。
b. 2 進数と同様に、16 進数というものがある。9→10→11（10 進数表記）という順は 9→a→b（16 進数表記）となる。
c. コンピュータで扱うデータの最小単位を『ビット（bit)』という。
d. 1 ビットで 0 または 1 という情報を表すことができる。2 ビットでは 2 の 2 乗の情報量を表すことができる。つまり、こういうことである。

   1 ビット：0000,0001　　（2 の 1 乗）
   2 ビット：0000,0001,0010,0011　　（2 の 2 乗）
   3 ビット：0000,0001,0010,0011,0100,0101,0110,0111　（2 の 3 乗）
e. 8 ビットで 1 バイトである。

以上のことを考えると int 型整数の上限を調べる方法として次の方法が考えられる。

〝コンピュータで扱うことのできるぎりぎりのビット数（2 の n 乗）と、できないビット数（2 の n＋1 乗）を調べる。扱うことのできないビット数から－1 だけすると、コンピュータで扱えるぎりぎりのビット数で表すことができる最後の数字（2 の n＋1 乗－1）がわかる〟

＊ソースコード 2＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊

```
/*
  Program    : numvalue-2.c
  Student-ID :075739A
  Authar     :TSUHAKO,Masaki
  UpDate     :2005/05/22(Tue)
```

```
  Comments   : zyougenn,kagenn
*/
#include <stdio.h>

int main(){

  int i, j;

  i=1; j=0x1;
  puts("i=1; j=0x1;");
  printf("(Hex) i = %09x, j = %09x\n",i,j);
  printf("(Dec) i = %09d, j = %09d\n",i,j);

  i=10; j=0x12;
  puts("i=10; j=012;");
  printf("(Hex) i = %09x, j = %09x\n",i,j);
  printf("(Dec) i = %09d, j = %09d\n",i,j);

  i=15; j=0x14;
  puts("i=15; j=0x14;");
  printf("(Hex) i = %09x, j = %09x\n",i,j);
  printf("(Dec) i = %09d, j = %09d\n",i,j);

  return(0);
}
```

＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊ソースコード終了＊＊＊＊＊＊＊＊＊＊

出力結果：

```
i=1; j=0x1;
(Hex) i = 000000001, j = 000000001
(Dec) i = 000000001, j = 000000001
i=10; j=012;
(Hex) i = 00000000a, j = 000000012
(Dec) i = 000000010, j = 000000018
```

i=15; j=0x14;

(Hex) i = 00000000f, j = 000000014

(Dec) i = 000000015, j = 000000020

上の出力結果より、

i=A,j=B の時、

(Hex) i = A の基数 16 表記, j = B の基数 16 表記

(Dec) i = A の基数 10 表示, j = B の基数 10 表記

というふうにこのプログラムが出力していることがわかる。

int 型整数の上限を調べる。

実験 1：

```
#include <stdio.h>

int main(){

  int i, j;

  i=1000000000; j=0x1;
  puts("i=1000000000; j=0x1;");
  printf("(Hex) i = %09x, j = %09x\n",i,j);
  printf("(Dec) i = %09d, j = %09d\n",i,j);

  return(0);
}
```

出力結果：

i=1000000000; j=0x1;

(Hex) i = 03b9aca00, j = 000000001

(Dec) i = 1000000000, j = 000000001

10 億は出力できた。

実験 2：

```
#include <stdio.h>
```

```
int main(){

  int i, j;

  i=3000000000; j=0x1;
  puts("i=3000000000; j=0x1;");
  printf("(Hex) i = %09x, j = %09x\n",i,j);
  printf("(Dec) i = %09d, j = %09d\n",i,j);

  return(0);
}
```

出力結果：

```
 [Masaki:~] e075739% cc prgrmng4.c
prgrmng4.c: In function ‘main’:
prgrmng4.c:14: warning: this decimal constant is unsigned only in ISO C90
```

（訳『注意：この 10 進法の定数は ISO C90 で無署名でしかない』）

コンパイラができなかった。（30 億は出力できなかった）

実験 1、2 より、10 億～30 億の間に int 型整数の上限があると考えられ、

2 の 30 乗＝1073741824　（約 10 億）

2 の 31 乗＝2147483648　（約 20 億）

2 の 32 乗＝4294967296　（約 40 億）

30 億は 2 の 31 乗＋852516352。なので 2 の 31 乗はコンピュータで扱うことができないビット数であることがわかった。ということは、2 の 31 乗－1 つまり 2147483647 が上限であると予想することができる。試してみる。

検証：2147483647 バージョン

```
#include <stdio.h>

int main(){

  int i, j;
```

```
  i=2147483647; j=0x1;
  puts("i=2147483647; j=0x1;");
  printf("(Hex) i = %09x, j = %09x\n",i,j);
  printf("(Dec) i = %09d, j = %09d\n",i,j);

  return(0);
}
```

出力結果：

```
i=2147483647; j=0x1;
(Hex) i = 07fffffff, j = 000000001
(Dec) i = 2147483647, j = 000000001
```

出力できた。

検証：2147483648 バージョン

```
#include <stdio.h>

int main(){

  int i, j;

  i=2147483648; j=0x1;
  puts("i=2147483648; j=0x1;");
  printf("(Hex) i = %09x, j = %09x\n",i,j);
  printf("(Dec) i = %09d, j = %09d\n",i,j);

  return(0);
}
```

出力結果：

```
[Masaki:~] e075739% cc prgrmng4.c
prgrmng4.c: In function ‘main’:
prgrmng4.c:14: warning: this decimal constant is unsigned only in ISO C90
```

コンパイラができなかった。

検証結果より、int 型整数の上限は『2147483647』ということがわかった。

次は int 型整数の下限を調べる。

10 進数の 2147483648 はコンパイラできなかった。では 2147483648 を 16 進数で表した値、『0x80000000』はコンパイラできるのだろうか。

実験：

```
#include <stdio.h>

int main(){

  int i, j;

  i=2147483647; j=0x80000000;
  puts("i=2147483647; j=0x80000000;");
  printf("(Hex) i = %09x, j = %09x\n",i,j);
  printf("(Dec) i = %09d, j = %09d\n",i,j);

  return(0);
}
```

出力結果：

```
i=2147483647; j=0x80000000;
(Hex) i = 07fffffff, j = 080000000
(Dec) i = 2147483647, j = -2147483648
```

コンパイラできた。

しかも 16 進数 0x80000000 の値がマイナスになっている。また実験をやってみる。

実験２：

```
#include <stdio.h>

int main(){

  int i, j;
```

```
  i=2147483647; j=0x80000001;
  puts("i=2147483647; j=0x80000001;");
  printf("(Hex) i = %09x, j = %09x\n",i,j);
  printf("(Dec) i = %09d, j = %09d\n",i,j);

  i=2147483647; j=0x80000002;
  puts("i=2147483647; j=0x80000002;");
  printf("(Hex) i = %09x, j = %09x\n",i,j);
  printf("(Dec) i = %09d, j = %09d\n",i,j);

  i=2147483647; j=0xffffffff;
  puts("i=2147483647; j=0xffffffff;");
  printf("(Hex) i = %09x, j = %09x\n",i,j);
  printf("(Dec) i = %09d, j = %09d\n",i,j);
  return(0);
}
```

出力結果：

```
i=2147483647; j=0x80000001;
(Hex) i = 07fffffff, j = 080000001
(Dec) i = 2147483647, j = -2147483647
i=2147483647; j=0x80000002;
(Hex) i = 07fffffff, j = 080000002
(Dec) i = 2147483647, j = -2147483646
i=2147483647; j=0xffffffff;
(Hex) i = 07fffffff, j = 0ffffffff
(Dec) i = 2147483647, j = -00000001
[Masaki:~] e075739%
```

上の出力結果から次ことがわかった。

a. 16 進数：0x7fffffff→0x80000000 を 10 進数に変換すると
   10 進数：2147483647→-2147483648 になる。

b. 16 進数 0x80000000 から値をどんどん増やしていくと 10 進数に変換した時、どんどん『0』に近づいていく。

例

16 進数：0x80000000→0x80000001→0x80000000→……→0xffffffff→0x00000000

10 進数：-2147483648→-2147483647→-2147483646→……→-1→0

c. int 型整数の下限は『-2147483648』である。

d. int 型整数は循環している。

よって

int 型整数の上限は『2147483647』である。下限は『-2147483648』である。

iv.エラーについて考察せよ。

1.

```
#include <stdio.h>

int main(){

  int i, j;

  i=2147483647; j=0x80000001;
  puts("i=2147483647; j=0x80000001;");
  printf("(Hex) i = %09x, j = %09x¥n",i,j);
```

```
    printf("(Dec) i = %09d, j = %09d¥n",i,j);

    i=2147483647; j=0x80000002;
    puts("i=2147483647; j=0x80000002;");
    printf("(Hex) i = %09x, j = %09x¥n",i,j);
    printf("(Dec) i = %09d, j = %09d¥n",i,j);

    i=2147483647; j=0xffffffff;
    puts("i=2147483647; j=0xffffffff;");
    printf("(Hex) i = %09x, j = %09x¥n",i,j);
    printf("(Dec) i = %09d, j = %09d¥n",i,j);
    return("2147483647");
}
```

出力結果：

```
prgrmng4.c: In function ‘main’:
prgrmng4.c:28: warning: return makes integer from pointer without a cast
```

コンパイルできなかった。理由は return **の（）の中の数字にある。調べてみると** return(0)**というものは、オペレーティングシステムにプログラムが正常に終了したことを通知するために使用する、とあった。なので** return **の(　)の中身は『**0**』にしなければいけない。**

2.

```
#include <stdio.h>

int main(){

    int i, j;

    i=2147483648; j=0x80000000;
    puts("i=2147483648; j=0x80000000;");
    printf("(Hex) i = %10x, j = %09x¥n",i,j);
```

```
   printf("(Dec) i = %10d, j = %09d¥n",i,j);

   return(0);
}
```

コンパイル実行

```
prgrmng2.c: In function ‘main’:
prgrmng2.c:14: warning: this decimal constant is unsigned only in ISO C90
```

コンパイルできなかった。しかし、次のような場合はコンパイルできた。

```
*/
#include <stdio.h>

int main(){

   int i, j;

   i=2147483648; j=0x80000000;
   puts("i=2147483648; j=0x80000000;");
   printf("(Hex) i = %10x, j = %10x¥n",i,j);
   printf("(Dec) i = %10d, j = %10d¥n",i,j);

   return(0);
}
```

出力結果：

```
i=2147483648; j=0x80000000;
(Hex) i =      80000000, j =      80000000
(Dec) i = -2147483648, j = -2147483648
```

整数が 2147483648 のときはコンパイルできなかったが、

```
printf("(Hex) i = %09x, j = %10x¥n",i,j);
printf("(Dec) i = %09d, j = %10d¥n",i,j);
```

から

```
printf("(Hex) i = %10x, j = %10x¥n",i,j);
printf("(Dec) i = %10d, j = %10d¥n",i,j);
```

に変えたところ、コンパイルができた。コンパイルされたので、上限の 2147483647 のをこえ

て 2147483648 が出力されると思ったが、出力されたのは－2147483648 であった。この事から上限はやはり 2147483647 ということがわかった。

参考文献：

スティーブン・オウアルライン(谷口功訳)（実践プログラミング第 3 版　オーム社 2006/12/11）

インターネット：

http://www.asahi-net.or.jp/~ax2s-kmtn/ref/bdh.html(2 進数、16 進数と 10 進数)

文字コード：Osaka レギュラー等幅　文字サイズ 10

感想：

2 週間あれば余裕だろ！！とか思ってたけど人間思っているほどことは進まないということを実感させられた。一度宿題をやりはじめ、途中で中断すると前回やったところがわからなくなって困った。また 2 進数、16 進数もちょっとしか分からなかったので、ほぼゼロからのスタートだった。

バランスのプログラムは面白かった。

コンピュータにも数字の限界があるのは意外だった。