プログラミングⅠ

Report#2

提出日：５月１６日（木）

所属：琉球大学工学部情報工学科

学籍番号：１３５７１３B

氏名：天願寛之

目次

C 言語の基本演算

1. scanf()関数による標準入力と基本演算子

ⅱ.1234 円の買い物をして 1 万円札を出した時の、お釣りの札と硬貨の枚数を求めるプログラムを作成せよ。

C 言語の基本演算（+,-,*,/,%）

scanf()関数による標準入力と基本演算子

ⅱ．1234 円の買い物をして 1 万円札を出した時の、お釣りの札と硬貨の枚数を求めるプログラムを作成せよ。

1. scanf()関数を用いて、価格と支払い金額を入力せよ。

1-a 高水準コードの一部

```
01 #include <stdio.h>
02
03 int main()
04 {
05   int price = 1234, pay = 10000;
06   int balance, amount;
07   int note, coin, total;
08   /****************************** scanf */
09   printf("Price?   => "); scanf("%d",&price);
10   printf("Payment? => "); scanf("%d",&pay);
11   printf("----¥n");
12
13   /****************************** balance */
14   balance = pay - price;
15   printf("price = %d, ",price);
16   printf("payment = %d, balance = %d¥n",pay,balance);
17   printf("----¥n");
18   /****************************** 5000-yen */
19   amount  = balance / 5000;
20   balance = balance % 5000;
21   printf("5000-yen note = %d¥n",amount);
22   note = amount;
23
24   /****************************** 2000-yen */
25   amount  = balance / 2000;
26   balance = balance % 2000;
27   printf("2000-yen note = %d¥n",amount);
28   note = note + amount;
29
30   /****************************** 1000-yen */
31   amount  = balance / 1000;
32   balance = balance % 1000;
33   printf("1000-yen note = %d¥n",amount);
34   note = note + amount;
35
36   /****************************** 500-yen */
37   amount  = balance / 500;
38   balance = balance % 500;
39   printf("500-yen coin = %d¥n",amount);
40   coin = amount;
41
42   /****************************** 100-yen */
43   amount  = balance / 100;
44   balance = balance % 100;
45   printf("100-yen coin = %d¥n",amount);
46   coin = coin + amount;
47
48   /****************************** 50-yen */
49   amount  = balance / 50;
50   balance = balance % 50;
51   printf("50-yen coin = %d¥n",amount);
```

```
52   coin = coin + amount;
53
54   /****************************** 10-yen */
55   amount  = balance / 10;
56   balance = balance % 10;
57   printf("10-yen coin = %d¥n",amount);
58   coin = coin + amount;
59
60   /****************************** 5-yen */
61   amount  = balance / 5;
62   balance = balance % 5;
63   printf("5-yen coin = %d¥n",amount);
64   coin = coin + amount;
65
66   /****************************** 1-yen */
67   printf("1 -yen coin = %d¥n" ,balance);
68   coin = coin + balance;
69
70   /****************************** total */
71   printf("note = %d ,coin = %d¥n" ,note,coin);
72   total = note + coin;
73   printf("total = %d¥n" ,total);
74
75   return(0);
76 }
```

1-b 実行結果

```
Price?   => 1234
Payment? => 10000
----
price = 1234, payment = 10000, balance = 8766
----
5000-yen note = 1
2000-yen note = 1
1000-yen note = 1
500-yen coin = 1
100-yen coin = 2
50-yen coin = 1
10-yen coin = 1
5-yen coin = 1
1 -yen coin = 1
note = 3 ,coin = 7
total = 10
```

1-c 考察

イ. 1234 円の買い物をして 1 万円札を出した時の、お釣りの札と硬貨の枚数を求めることが出来た。

ロ. 07 行目、枚数を明らかにするため変数名 ‘note’ ,coin’,’total’ を宣言した。

ハ. ‘price’ に ’10000’, ‘payment’ に ’1000000000000’ を代入すると ’price’ には 10000 が代入されたが、’payment’にはなぜか’-727379968’が代入された。

ニ. 結局、後の考察で分かったが、表記できる数値には限りがあり、それを超えると正常に表記されないことが分かった。

ホ. ‘price’ や’payment’に半角数字や全角数字を代入したらそれぞれの初期値が

代入された。

ヘ. 小数点がつくと、小数点以下の数値以外が代入された。

1-d 試み 1.scanf 関数・10 進数、16 進数による数値入力の入力形式変換

1-d-1 高水準コードの一部

```
01 #include <stdio.h>
02 int main()
03 {
04   int try, chal;
05   scanf("%d%x",&try,&chal);
06   printf("try 10base: = %d¥n    16base: = %x¥n" ,try,try);
07   printf("chal 10base: = %d¥n   16base: = %x¥n" ,chal,chal);
08
09   return(0);
10 }
```

1-d-2 実行結果（数値 15 を入力）

```
15
15
try 10base: = 15
    16base: = f
chal 10base: = 21
   16base: = 15
```

1-d-3 考察

イ. 10 進数の'try'、16 進数の'chal'それぞれに数値を入力したらいずれも変換できた。

ロ. 10 進数で数値 15 は 16 進数で f に変換された。

ハ. 16 進数で数値 15 は 10 進数で 21 に変換された。

1-e 試み 2.浮動小数点型による数値入力の入力形式変換

'試み 1'でのプログラムの'%x'と'%f'をそのまま交換する。

1-e-1 高水準コードの一部

```
01 #include <stdio.h>
02 int main()
03 {
04   int try, att;
05   scanf("%d%f",&try,&att);
06   printf("try 10base: = %d¥n    16base: = %f¥n" ,try,try);
07   printf("att 10base: = %d¥n   16base: = %f¥n" ,att,att);
08
09   return(0);
10 }
```

1-e-2 実行結果

```
report#2-2.c:5:21: warning: format specifies type 'float *' but the argument has type 'int *'
[-Wformat]
  scanf("%d%f",&try,&att);
          ~~       ^~~~
          %d
report#2-2.c:6:54: warning: format specifies type 'double' but the argument has  type 'int'
[-Wformat]
  printf("try 10base: = %d¥n    16base: = %f¥n" ,try,try);
                                      ~~         ^~~
                                      %d
report#2-2.c:7:53: warning: format specifies type 'double' but the argument has type 'int' [-Wformat]
  printf("att 10base: = %d¥n   16base: = %f¥n" ,att,att);
                                     ~~         ^~~
                                     %d
3 warnings generated.
```

1-e-3 エラーの意味

イ. 5 行 21 桁：変換書式型'float*'ではなく主張が'int*'である。

ロ. 6 行 54 桁、7 行 53 桁：変換書式型'double'ではなく主張が'int'である。int 型ではなく float 型で試みたところ逆のエラーが出た事から'%f'だけを float 型で再度試みる。

1-e-4 修正後

```
01 #include <stdio.h>
02 int main()
03 {
04   float att;
05   scanf("%f",&att);
06   printf("att 0.00~: = %f¥n" ,att);
07
08   return(0);
09 }
```

1-e-5 実行結果（3.1415 を代入）

```
3.1415
att 0.00~: = 3.141500
```

1-e-6 考察

イ. 浮動小数点数の形式変換をすることが出来た。

ロ. 10 進数と 16 進数の形式変換には int 型、浮動小数点数の形式変換には float 型が有効であることが分かった。

ハ. 代入した数値のほかに 0 が小数点第 6 位まで出力された。

ニ. 小数点第 7 位まで代入してみる。

1-e-7 実行結果(3.1415926 を代入する)

```
3.1415926
att 0.00~: = 3.141593
```

float 型による浮動小数点数の形式変換では小数点第 6 位まで出力でき、それ以下は四捨五入される事が分かった。

2. **例題の変数名を変え、自分自身で考えた変数名にせよ。**

半角文字による変数名変更は成立しているので、異なる文字種で試みる。

2-a 高水準コードの一部

```
01 #include <stdio.h>
02 int main()
03 {
04   int 値段 = 1234, 支払い = 10000;
05   int お釣り, 量;
06   /****************************** scanf */
07   printf("値段?   => "); scanf("%d",&値段);
08   printf("支払い? => "); scanf("%d",&支払い);
09   printf("----¥n");
10
11   /****************************** balance */
12   お釣り = 支払い - 値段;
13   printf("値段 = %d, ",値段);
14   printf("支払い = %d, お釣り = %d¥n",支払い,お釣り);
15   printf("----¥n");
16
17   /****************************** 5000-yen */
18   量  = お釣り / 5000;
19   お釣り = お釣り % 5000;
20   printf("----¥n");
21
22   return(0);
23 }
```

2-b 実行結果

```
report#2-2.c:5:7: error: expected identifier or '('
  int お釣り, 量;
      ^
report#2-2.c:7:40: error: expected expression
  printf("値段?   => "); scanf("%d",&値段);
                                    ^
report#2-2.c:8:41: error: expected expression
  printf("支払い? => "); scanf("%d",&支払い);
                                    ^
report#2-2.c:12:3: error: expected expression
  お釣り = 支払い - 値段;
  ^
report#2-2.c:13:26: error: expected expression
  printf("値段 = %d, ",値段);
                       ^
report#2-2.c:14:45: error: expected expression
  printf("支払い = %d, お釣り = %d¥n",支払い,お釣り);
                                    ^
report#2-2.c:18:3: error: expected expression
  量  = お釣り / 5000;
  ^
report#2-2.c:19:3: error: expected expression
  お釣り = お釣り % 5000;
  ^
9 errors generated.
```

2-c エラーの意味

イ. 4 行 7 桁、5 行 7 桁：識別詞か'('が入ると予期される。

ロ. その他：表現が予期される。

考察

イ. 全角文字ではエラーの原因となってしまうと分かった。

ロ. 半角英数字の大文字や記号、数字で試みる。

2-d 高水準コードの一部(半角英数字の大文字)

```
01 #include <stdio.h>
02 int main()
03 {
04   int PRICE = 1234, PAY = 10000;
05   int BALANCE, AMOUNT;
06   /****************************** scanf */
07   printf("PRICE?   => "); scanf("%d",&PRICE);
08   printf("PAYMENT? => "); scanf("%d",&PAY);
09   printf("----¥n");
10
11   /****************************** balance */
12   BALANCE = PAY - PRICE;
13   printf("PRICE = %d, ",PRICE);
14   printf("PAYMENT = %d, BALANCE = %d¥n",PAY,BALANCE);
15   printf("----¥n");
16
17   /****************************** 5000-yen */
18   AMOUNT  = BALANCE / 5000;
19   BALANCE = BALANCE % 5000;
20   printf("5000-yen note = %d¥n",AMOUNT);
21
22   return(0);
23 }
```

2-e 実行結果

```
PRICE?   => 1234
PAYMENT? => 10000
----
PRICE = 1234, PAYMENT = 10000, BALANCE = 8766
----
5000-yen note = 1
```

2-f 考察

イ．半角英数字の大文字では出力することが出来た。

2-g 高水準コードの一部(半角数字と記号)

```
01 #include <stdio.h>
02 int main()
03 {
04   int  1# = 1234, *1 = 10000;
05   int 3¥, 4%;
06   /****************************** scanf */
07   printf("price?   => "); scanf("%d",&1#);
08   printf("amount? => "); scanf("%d",&*1);
09   printf("----¥n");
10
11   /****************************** balance */
12   3¥ = *1 - 1#;
13   printf("PRICE = %d, ",1#);
14   printf("PAYMENT = %d, BALANCE = %d¥n",*1,3¥);
15   printf("----¥n");
16
17   /****************************** 5000-yen */
18   3¥  = 3¥ / 5000;
19   3¥ = 3¥ % 5000;
20   printf("5000-yen note = %d¥n",4%);
21
22   return(0);
23 }
```

2-h 実行結果

```
report#2-2.c:4:8: error: expected identifier or '('
  int  1# = 1234, *1 = 10000;
       ^
report#2-2.c:5:7: error: expected identifier or '('
  int 3¥, 4%;
      ^
report#2-2.c:7:38: error: address expression must be an lvalue or a function
      designator
  printf("price?   => "); scanf("%d",&1#);
                                    ^~
report#2-2.c:8:38: error: indirection requires pointer operand ('int' invalid)
  printf("amount? => "); scanf("%d",&*1);
                                     ^~
report#2-2.c:12:4: error: expected ';' after expression
  3¥ = *1 - 1#;
   ^
   ;
report#2-2.c:12:4: error: expected expression
report#2-2.c:13:26: error: expected ')'
  printf("PRICE = %d, ",1#);
                         ^
report#2-2.c:13:9: note: to match this '('
  printf("PRICE = %d, ",1#);
        ^
report#2-2.c:14:41: error: indirection requires pointer operand ('int' invalid)
  printf("PAYMENT = %d, BALANCE = %d¥n",*1,3¥);
                                       ^~
report#2-2.c:18:4: error: expected ';' after expression
  3¥  = 3¥ / 5000;
   ^
   ;
report#2-2.c:18:4: error: expected expression
report#2-2.c:19:4: error: expected ';' after expression
  3¥ = 3¥ % 5000;
   ^
   ;
report#2-2.c:19:4: error: expected expression
report#2-2.c:20:35: error: expected expression
  printf("5000-yen note = %d¥n",4%);
                                  ^
report#2-2.c:12:3: warning: expression result unused [-Wunused-value]
  3¥ = *1 - 1#;
  ^
report#2-2.c:18:3: warning: expression result unused [-Wunused-value]
  3¥  = 3¥ / 5000;
  ^
report#2-2.c:19:3: warning: expression result unused [-Wunused-value]
  3¥ = 3¥ % 5000;
  ^
3 warnings and 13 errors generated.
```

2-i エラーの意味（多いので要約する。）

イ. ‘¥’の前の数字は結果使われていない。

ロ. 数字の後ろには’;’が付くと予測される。

ハ. 処理対象である’*1’に対して’int’が無効になっている。

ニ. 変数である 1 と 3 の位置に識別詞が’(‘つくと予測される。

2-j 考察

イ. ¥'は printf 関数と同様にエラーの原因になると思われるので削除する。

ロ. '¥'のかわりに';'を加えてみる。

ハ. 識別詞はデータの区別をするものであり数字が前にあるものに反応しているので逆にしてみる。

2-k エラーを改善し試行を繰り返した結果

イ. 半角でもローマ字と数字しか有効に使えず、それ以外はエラーを起こしてしまった。

ロ. 数字で始まる変数ではデータの識別が必要になってしまった。

3. 工夫...!

結果をわかりやすく示す。

3-a 高水準コードの一部

```
 1
 2     #include <stdio.h>
 3
 4     int main()
 5     {
 6       int price = 1234, pay = 10000;
 7       int balance, amount;
 8       int note, coin, total;
 9       /****************************** scanf */
10       printf("値段?   => "); scanf("%d",&price);
11       printf("支払い? => "); scanf("%d",&pay);
12       printf("----¥n");
13
14       /****************************** balance */
15       balance = pay - price;
16       printf("値段 = %d 円, ",price);
17       printf("支払い = %d 円, お釣り = %d 円¥n",pay,balance);
18       printf("----¥n");
19
20       /****************************** 5000-yen */
21       amount  = balance / 5000;
22       balance = balance % 5000;
23       printf("5000-円札 = %d 枚¥n",amount);
24       note = amount;
25
26       /****************************** 2000-yen */
27       amount  = balance / 2000;
28       balance = balance % 2000;
29       printf("2000-円札 = %d 枚¥n",amount);
30       note = note + amount;
31
32       /****************************** 1000-yen */
33       amount  = balance / 1000;
34       balance = balance % 1000;
35       printf("1000-円札 = %d 枚¥n",amount);
```

```
    note = note + amount;

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

    /****************************** 100-yen */
    amount  = balance / 100;
    balance = balance % 100;
    printf("100-円玉 = %d 枚¥n",amount);
    coin = coin + amount;

    /****************************** 50-yen */
    amount  = balance / 50;
    balance = balance % 50;
    printf("50-円玉 = %d 枚¥n",amount);
    coin = coin + amount;

    /****************************** 10-yen */
    amount  = balance / 10;
    balance = balance % 10;
    printf("10-円玉 = %d 枚¥n",amount);
    coin = coin + amount;

    /****************************** 5-yen */
    amount  = balance / 5;
    balance = balance % 5;
    printf("5-円玉 = %d 枚¥n",amount);
    coin = coin + amount;

    /****************************** 1-yen */
    printf("1 -円玉 = %d 枚¥n" ,balance);
    coin = coin + balance;

    /****************************** total */
    printf("お札 = %d 枚 ,硬貨 = %d 枚¥n" ,note,coin);
    total = note + coin;
    printf("お札と硬貨の総枚数 = %d 枚¥n" ,total);

    return(0);
}
```

3-b 実行結果

```
値段?   => 1234
支払い? => 10000
----
値段 = 1234 円，支払い = 10000 円，お釣り = 8766 円
----
5000-円札 = 1 枚
2000-円札 = 1 枚
1000-円札 = 1 枚
500-円玉 = 1 枚
100-円玉 = 2 枚
50-円玉 = 1 枚
10-円玉 = 1 枚
5-円玉 = 1 枚
1 -円玉 = 1 枚
お札 = 3 枚 ,硬貨 = 7 枚
お札と硬貨の総枚数 = 10 枚
```

3-c 考察

イ. 日本人になら一番わかりやすいと思う工夫で実行結果を出せた。

4. 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。

4-a 高水準コードの一部

```
01 #include <stdio.h>
02
03 int main(){
04
05   int i;
06
07   printf("i = ");scanf("%d",&i);
08   printf("16進数表記 i = ox%08x¥n",i);
09   printf("10進数表記 i = %011d¥n",i);
10
11   return(0);
12 }
```

4-b 実行結果（i には 1000 を代入した）

```
i = 1000
16進数表記 i = 0x000003e8
10進数表記 i = 00000001000
```

4-c 考察

イ. i に適当な数字を代入したら、16 進数と 10 進数で表記されるプログラムを作成した。

ロ. 未使用の桁を埋めるため、'0'で埋められるようにした。

ハ. 教科書より int 型で 10 進数の最限値については負の符号をふまえると 11 桁と分かったため 11 桁分の 0 を用意した。

ニ. int 型の最大ビット数は 32 であり、16 進数は 4 ビットで 1 桁を表記することから最大表記は 8 桁と分かっているので、8 桁分の 0 を用意した。

ホ. '0x'は 16 進数での表現に使用する。

4-d 予想

2 進数 4 ビットで表せる数は 2^4 個である。符号付きの整数型数値では最大 7、最小-8 であり、これは正と負の個数を 2^(4-1)で分けた時に正の符号に 0 を加えた結果である。そのため n ビットで表せる数は 2^n 個であり、その範囲は-2^(n-1)~2^(n-1)-1 となる。

よって int 型は 32 ビットであるため表せる数は 2^32 個＝4294967296 個であり、その範囲は-2^31~2^31-1=-2147483648~2147483647 と分かる。

4 ビットで表せる数の最大値は 7 となるが、7 の 2 進数表記は 0111 である。コンピュータの数値の内部表現は 0 と 1 で表されることをふまえて、この 0111 に 1 を足すと 2 進数表記で 1000 である。4 ビットでは 10 進数で表せる数値の最大は 7 であるため 8 という数字は出ないはずである。実際、1000 を 10 進数表記すると-8 になる。つまり 4 ビットで 10 進数表記の最大値に 2 進数表記での 1 を足すと 10 進数で最小値をとることになる。

1000(-8)~1111(-1)、10 進数ではこれらのとおり最上位の数値が 1 のとき負の値をとることになるが、これを 1 の補数表現という。

従って、これを 32 ビットで考えると上限値に 1 を足すと下限値になり、1 を引くと-1 されると考えられ、下限値に 1 を足すと-1 され、1 を引くと上限値になるはずである。

4-e .int 型整数の上限と下限の値についての観察

int 型整数の上限値は'2147483647'、下限値は'-2147483648'であると分かっているため、その周辺での動きを観察する。

前述で作ったプログラムを使う

（2147483648 を代入）上限値+1

```
i = 2147483648
16 進数表記 i = 0x80000000
10 進数表記 i = -2147483648
```

2 進数での表記、（1000,0000,0000,0000,0000,0000,0000,0000）

（2147483646 を代入）上限値-1

```
i = 2147483646
16 進数表記 i = 0x7ffffffe
10 進数表記 i = 2147483646
```

2 進数での表記、（0111,1111,1111,1111,1111,1111,1111,1110）

（-2147483647 を代入）下限値+1

```
i = -2147483647
16 進数表記 i = 0x80000001
10 進数表記 i = -2147483647
```

2 進数での表記、（1000,0000,0000,0000,0000,0000,0000,0001）

（-2147483649 を代入）下限値-1

```
i = -2147483649
16 進数表記 i = 0x7fffffff
10 進数表記 i = 2147483647
```

2 進数での表記、（0111,1111,1111,1111,1111,1111,1111,1111）

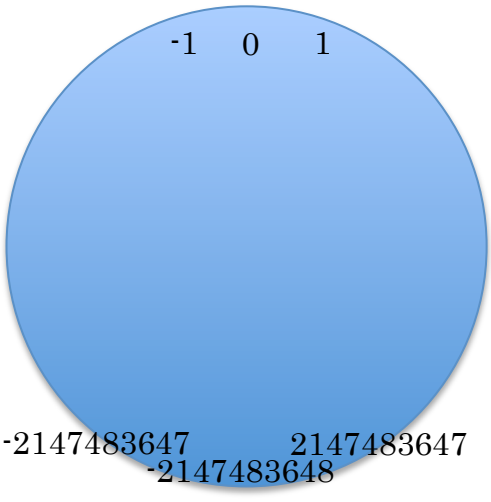
4-f 考察

イ. 10 進数については予想通りの結果が得られた。

ロ. 2 進数=16 進数の対応は 1111=f,1110=e,1000=8,0111=7,である。

ハ. これらのことより、上限値に 1 を加えると下限値になり、そのまま 1 を加え続けるとそのまま値が 1 ずつ大きくなっていくことが分かり、最終的に 2 進数で 32 ビット全てが 1 の時、また 16 進数で f が 8 個並んだ時、10 進数での値は-1 になる理由が分かる。

コンピュータの内部表現(10 進数数値)



5．エラーについて考察せよ。

イ. 4-a で変数名を変えたが、半角英数字の英字から始まるもの以外(記号も含む)は識別詞が必要になったり、’)’や表現が入ると予期された。

ロ．半角英数字の英字から始まるもの以外(記号も含む)をコンピュータの警告通り’(‘と’)’で囲ってもうまくはいかなかった。

ハ．識別詞や表現については改善の仕様が分からなかった。

ニ．浮動小数点数の表記を試みた結果 int 型では整数部分すら表記出来なかった。

ホ．float 型では浮動小数点数の表記は成功し、入力したところ小数点は 6 桁まで表せて、7 桁は四捨五入されることが分かった。

ヘ. float 型で整数を代入すると小数点以下も 0 で出力された。

6. 反省・感想

課題の意味についてあまり理解が出来ていなかったのでずいぶんと手こずりました。上限下限の話は最初になにをしたらよいのか検討もつきませんでしたが、こういうことなのかなと落ち着いた次第です。しかし、今回の買い物のプログラムは実用性があって、頭で計算せずにおつりや、お札や硬貨の枚数を割り出せた事については感動しました。もっとうまく作れば他にも出来るのかなと思いましたが、ひらめきはきませんでした。そういう才能が欲しいです。

今回は手書きじゃなくて本当によかった。