プログラミングⅠ

REPORT#2

提出日	:	2012年5月24日
所属	:	琉球大学工学部情報工学科
学籍番号	:	125722G
氏名	:	知念 辰明

ii.1234 円の買い物をして 1 万円札を出したときの、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。

a.scanf()関数を用いて、価格と支払い金額を入力せよ。

a-1:scanf()関数による標準入力と、お釣りの枚数の計算。

高水準コードの一部▼　※コード冒頭、コメントアウトによるファイル説明などの行は省略。

```
8   #include <stdio.h>
9   int main(){
10
11      int price=1234,pay=10000;
12      int balance, amount;
13      int note, coin, total;
14      printf( "finished definition.\n");
15      /************************** scanf ***/
16      printf( "Price?   => "); scanf( "%d",&price);
17      printf( "Payment? => "); scanf( "%d",&pay);
18      printf( "----\n");
19      /************************** balance */
20      balance=pay-price;
21      printf( "price=%d, ",price);
22      printf( "payment=%d, balance=%d\n",pay,balance);
23      printf( "----\n" );
24      /************************** 5000-yen*/
25      amount = balance / 5000;
26      balance = balance % 5000;
27      printf( "5000-yen note = %d\n",amount);
28      note = amount;
29      /************************** 2000-yen*/
30      amount = balance / 2000;
31      balance = balance % 2000;
32      printf( "2000-yen note = %d\n",amount);
33      note = note + amount;
34      /************************** 1000-yen*/
35      amount = balance / 1000;
36      balance = balance % 1000;
37      printf( "1000-yen note = %d\n",amount);
38      note = note + amount;
39      /************************** 500-yen*/
40      amount = balance / 500;
41      balance = balance % 500;
42      printf( "500 -yen coin = %d\n",amount);
43      coin = amount;
44      /************************** 100-yen*/
45      amount = balance / 100;
46      balance = balance % 100;
47      printf( "100 -yen coin = %d\n",amount);
48      coin = coin + amount;
49      /************************** 50-yen*/
50      amount = balance / 50;
51      balance = balance % 50;
52      printf( "50  -yen coin = %d\n",amount);
53      coin = coin + amount;
54      /************************** 10-yen*/
55      amount = balance / 10;
56      balance = balance % 10;
57      printf( "10  -yen coin = %d\n",amount);
58      coin = coin + amount;
59      /************************** 5-yen*/
60      amount = balance / 5;
61      balance = balance % 5;
62      printf( "5   -yen coin = %d\n",amount);
63      coin = coin + amount;
64      /************************** 1-yen*/
65      printf( "1   -yen coin = %d\n",balance);
66      coin = coin + balance;
67      printf( "----\n" );
68      /************************** total*/
69      printf( "note = %d ,coin = %d\n",note,coin);
70      total = note + coin;
71      printf( "total = %d\n",total);
72      return(0);
73
74  }
```

実行結果▼　※変数"price"、"pay"はそれぞれ初期値である"1234"と"10000"を入力した。

```
finished definition.
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
```

考察▼

・scanf()関数を用いて、一万円のお釣りが出ないような入力に対応するプログラムができた。
・お釣りの札と硬貨の枚数を表す為に、新たに変数"note"、"coin"、"total"を宣言した。
・変数宣言が終わったことを分かりやすくする為に、変数宣言を終えた直後に
printf()関数を用いて"finished definition."というメッセージを表示させている。
・仮にscanf()関数に対して半角数字以外の文字が入力された場合、"price"には"1234"が、
"pay"には"10000"がそれぞれ代入される。
・"price"に半角数字を、"pay"に半角数字以外を入力すると、"pay"だけが初期値"10000"で扱われる。
・あまりにも大きすぎる数字（2200000000など）を入力すると正常に計算されない。
（"2100000000"やその直前後の整数値を"price"や"pay"に入力すると正常に計算されることを確認した。）

a-2:scanf()関数において変換指定子を使った入力。

高水準コードの一部▼　※コード冒頭、コメントアウトによるファイル説明などの行は省略。

```
8   #include <stdio.h>
9   int main(){
10
11      int abc , def;
12      printf( "変数宣言が終了しました。\n" );
13      scanf( "%x%d",&abc,&def );
14      printf( "abc Dec: = %d \n    Hex: = %x\n",abc,abc);
15      printf( "def Dec: = %d \n    Hex: = %x\n",def,def);
16      return(0);
17
18  }
```

実行結果▼　※scanf()関数にはいずれも"10"を入力した。

```
変数宣言が終了しました。
10
10
abc Dec: = 16
    Hex: = 10
def Dec: = 10
    Hex: = a
```

考察▼

・scanf()関数によって、10進数や16進数で値を取り込み、それを10進数や16進数に変換して出力する
プログラムを作った。
・a-1項でprintf()関数やscanf()関数に使われている変換指定子"%d"は、
Wikiサイト『Open Lecture』（URLは後述）によれば10進数を表す変換指定子である。
・これが16進数を表す"%x"になった場合を観察する。
・一回目のscanf()関数によって入力された値を出力すると、10進数では"16"、16進数では"16"を表す"10"の出力が得られた。
・二回目のscanf()関数によって入力された値を出力すると、10進数では"10"、16進数では"10"を示す"a"の出力が得られた。
・よって、確かに変換指定子"%x"による入力は16進数として入力され、"%d"による入力は10進数として入力されることが確認できた。

b.例題の変数名を変え、自分自身で考えた変数名にせよ。

b-1:半角英数での変数宣言。

高水準コードの一部▼

```
8   #include <stdio.h>
9   int main(){
10
11      int az12 = 1234, 12az = 10000;
12      int Remnant, 1942;
13
14      /************************** scanf ***/
15      printf( "Price?   => "); scanf( "%d",&az12);
16      printf( "Payment? => "); scanf( "%d",&12az);
17      printf( "----\n");
18      /************************** Remnant */
19      Remnant=12az-az12;
20      printf( "az12=%d, ",az12);
21      printf( "12az=%d, Remnant=%d\n",12az,Remnant);
22      printf( "----\n" );
23      /************************** 5000-yen*/
24      1942 = Remnant / 5000;
25      Remnant = Remnant % 5000;
26      printf( "5000-yen note = %d\n",1942);
27      return(0);
28
29  }
```

コンパイルする際のエラー▼

```
ypb2.c:11:22: error: invalid suffix "az" on integer constant
ypb2.c: In function ‘main’:
ypb2.c:11: error: expected identifier or ‘(’ before numeric constant
ypb2.c:12: error: expected identifier or ‘(’ before numeric constant
ypb2.c:16:43: error: invalid suffix "az" on integer constant
ypb2.c:19:13: error: invalid suffix "az" on integer constant
ypb2.c:21:37: error: invalid suffix "az" on integer constant
ypb2.c:24: error: lvalue required as left operand of assignment
```

エラーの考察▼

・11 行目：22 列目、int 定数に余計な文字列が付いている。
・11 行目：数字の前に(があるかを確認して下さい。
・16 行目：43 列目、int 定数に余計な文字列が付いている。
・24 行目：割り当てられる変数を左辺に置いて下さい。

考察▼

・数字を先頭にした変数ではエラーが発生している。
・数字が先頭でない変数ではエラーが発生していない。
・改善する為に、数字を先頭に含む変数をそうでない変数名へ書き換える。

修正した高水準コードの一部▼

```
8   #include <stdio.h>
9   int main(){
10
11      int az12 = 1234, ZA12 = 10000;
12      int Remnant, VWXYZ1942;
13
14      /************************** scanf ***/
15      printf( "Price?   => "); scanf( "%d",&az12);
16      printf( "Payment? => "); scanf( "%d",&ZA12);
17      printf( "----\n");
18      /************************** Remnant */
19      Remnant = ZA12 - az12;
20      printf( "az12=%d, ",az12);
21      printf( "ZA12=%d, Remnant=%d\n",ZA12,Remnant);
22      printf( "----\n" );
23      /************************** 5000-yen*/
24      VWXYZ1942 = Remnant / 5000;
25      Remnant = Remnant % 5000;
26      printf( "5000-yen note = %d\n",VWXYZ1942);
27      return(0);
28
29  }
```

実行結果▼　※変数"az12"、"ZA12"にはそれぞれ"980"と"10000"を入力した。

```
Price?   => 980
Payment? => 10000
----
az12=980, ZA12=10000, Remnant=9020
----
5000-yen note = 1
```

考察▼

・数字を先頭にしなければ、変数として宣言することができた。

・次に、大文字と小文字の区別がされるかを試行する。

高水準コードの一部▼

```
8   #include <stdio.h>
9   int main(){
10
11      int za12 = 1234, ZA12 = 10000;
12      int REMNANT, remnant;
13      /************************** scanf ***/
14      printf( "Price?   => "); scanf( "%d",&za12);
15      printf( "Payment? => "); scanf( "%d",&ZA12);
16      printf( "----\n");
17      /************************** Remnant */
18      REMNANT = ZA12 - za12;
19      printf( "za12=%d, ",za12);
20      printf( "ZA12=%d, REMNANT=%d\n",ZA12,REMNANT);
21      printf( "----\n" );
22      /************************** 5000-yen*/
23      remnant = REMNANT / 5000;    REMNANT = REMNANT % 5000;
24      printf( "remnant = %d ,REMNANT = %d\n",remnant,REMNANT);
25      printf( "remnantとREMNANTはprintf関数で同時に出力されています。\n" );
26      printf( "5000-yen note = %d\n",remnant);
27      return(0);
28
29  }
```

実行結果▼　※変数"za12"、"ZA12"にはそれぞれ"980"と"10000"を入力した。

```
Price?   => 980
Payment? => 10000
----
za12=980, ZA12=10000, REMNANT=9020
----
remnant = 1 ,REMNANT = 4020
remnantとREMNANTはprintf関数で同時に出力されています。
5000-yen note = 1
```

考察▼

・実行結果から分かる通り、大文字と小文字は区別され、別々の値を代入することができた。

b-3:全角文字や"%"、"\"などの記号による変数名。

考察▼

・この項についての試行や考察は、エラーについての考察で取り扱う。

b-4:変数宣言できる文字数の限界。

高水準コードの一部▼

```
1   #include <stdio.h>
2   int main(){
3
4       int o123456789;
5       o123456789 = 1 + 25;
6       printf( "int = %d\n",o123456789);
7       return(0);
8
9   }
```

考察▼

・上記のようなプログラムで、変数として宣言できる文字数の限界を確認しようとした。
・4行目、5行目、6行目で扱われる変数名に、"0123456789"という文字列を次々に足していくという手段をとった。
・変数として宣言できる文字列の文字数に制限があるのかを試行した結果、少なくとも185630文字
　までは正常に認識されることが分かった。（それ以降については確認していない）

c.工夫...!

c:端数合わせ計算や、printf()関数を使っての日本語表示など。

高水準コードの一部▼

```
8   #include <stdio.h>
9   int main(){
10
11      int price=1234,pay=10000;
12      int balance, amount;
13      printf( "変数宣言が終わりました。\n" );
14      /************************** scanf ***/
15      printf( "値段はいくらですか?   => "); scanf( "%d",&price);
16      printf( "いくら払いますか?     => "); scanf( "%d",&pay);
17      printf( "----\n");
18
19      /************************** balance */
20      balance=pay-price;
21      printf( "値段=%d円, 支払額=%d円, お釣り=%d円\n",price,pay,balance);
22      printf( "----\n" );
23      amount = balance % 10;
24      price = 10 - amount + pay;
25      printf("%d円払えば、10円未満のお釣りが出ません。\n",price);
26      printf("----\n");
27      /************************** 5000-yen*/
28      amount = balance / 5000;
29      balance = balance % 5000;
30      printf( "5000円札 = %d枚, ",amount);
31      price = amount;
32      /************************** 2000-yen*/
33      amount = balance / 2000;
34      balance = balance % 2000;
35      printf( "2000円札 = %d枚, ",amount);
36      price = price + amount;
37      /************************** 1000-yen*/
38      amount = balance / 1000;
39      balance = balance % 1000;
40      printf( "1000円札 = %d枚\n",amount);
41      price = price + amount;
42      /************************** 500-yen*/
43      amount = balance / 500;
44      balance = balance % 500;
45      printf( "500円 玉 = %d枚, " ,amount);
46      pay = amount;
47      /************************** 100-yen*/
48      amount = balance / 100;
49      balance = balance % 100;
50      printf( "100円 玉 = %d枚\n",amount);
51      pay  = pay + amount;
52      /************************** 50-yen*/
53      amount = balance / 50;
54      balance = balance % 50;
55      printf( "50 円 玉 = %d枚, ",amount);
56      pay = pay + amount;
57      /************************** 10-yen*/
58      amount = balance / 10;
59      balance = balance % 10;
60      printf( "10 円 玉 = %d枚, ",amount);
61      pay = pay + amount;
62      /************************** 5-yen*/
63      amount = balance / 5;
64      balance = balance % 5;
65      printf( "5 円 玉  = %d枚, ",amount);
66      pay = pay + amount;
67      /************************** 1-yen*/
68      printf( "1 円 玉  = %d枚\n",balance);
69      pay = pay + balance;
70      printf( "----\n" );
71      /************************** total*/
72      printf( "紙幣 = %d枚, 硬貨 = %d枚\n",price,pay);
73      price = price + pay;
74      printf( "紙幣と硬貨の合計枚数 = %d枚\n",price);
75      return(0);
76
77  }
```

実行結果▼　※変数**"price"**、**"pay"**はそれぞれ初期値**"1234"**と**"10000"**を入力した。

```
変数宣言が終わりました。
値段はいくらですか?   => 1234
いくら払いますか?     => 10000
----
値段=1234円, 支払額=10000円, お釣り=8766円
----
10004円払えば、10円未満のお釣りが出ません。
----
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
----
紙幣 = 3枚, 硬貨 = 7枚
紙幣と硬貨の合計枚数 = 10枚
```

考察▼
・printf による出力を日本語にした。
・また、端数合わせの計算を組み込み、10 円未満のお釣りが出ない支払い方を提案する機能を追加した。
・scanf()関数により"pay"に対して端数合わせを考えた後の値を入力すると、10 円高い金額を提案されてしまう。
・なるべく変数宣言をしないように変数を取り扱った。
（a-1 項において int 型変数を 7 つ宣言しているのに対し、本項のコードでは 4 つの宣言で対応している）
・a-1 項でのプログラムと同じく、10000 円のお釣りには対応せず、scanf()関数による入力に半角数字以外を入力すると初期値"1234"、"10000"で計算が行われる。

iii.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。

a.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。

a-1:int 型整数の入力と出力について、基数 10 で入力し基数 10 や基数 16 で出力するプログラムを作成する。

高水準コードの一部▼ ※コード冒頭、コメントアウトによるファイル説明などの行は省略。

```
8   #include <stdio.h>
9   int main(){
10
11      int limit;
12      printf( "基数10や16で表記したい値を入力して下さい。\n" );
13      scanf("%d",&limit);
14      printf( "基数10による表記。 limit = %010d\n",limit);
15      printf( "基数16による表記。 limit = 0x%08x\n",limit);
16      return(0);
17
18  }
```

実行結果▼ ※scanf()関数によって、変数"limit"には"10000"が入力されている。

```
基数10や16で表記したい値を入力して下さい。
10000
基数10による表記。 limit = 0000010000
基数16による表記。 limit = 0x00002710
```

考察▼
・基数 10 や基数 16 で表したい数値を scanf()関数で入力し、printf()関数で出力するプログラムを作成した。
・テキスト『C 実践プログラミング 第三版』P409 によれば、int 型整数の bit 数は 32bit であり上限を"2147483647"、下限を"-2147483648"として値域を取るらしい。
・プログラミング I の授業でたびたび扱われる wiki サイト『Open Lecture』内の、printf()関数を説明する pdf を参考に基数 16 で出力するための変換指定子"%x"を入力している。（URL を後述する）
・また、2 進数 32bit は 16 進数の 8 桁で表されるはずであるから、変換指定子に 8 桁で表示するための"8"と、未使用の桁に"0"を詰める"0"を入力した。
・基数 10 の場合、テキストの上限値を参考に 10 桁で表すための"10"と未使用の桁に"0"を詰める"0"を入力した。
・int 型整数の上下限を確かめるために、その前後の適当な数値を scanf()関数を用いて上記のプログラムに入力し試行する。

a-2:int 型整数の上限の値について、上限値とその前後の数値を入力して観察する。

高水準コードの内容は上記プログラムと同一であるため省略する。
実行結果▼ ※scanf()関数によって、変数"limit"には"2147483648"が入力されている。

```
基数10や16で表記したい値を入力して下さい。
2147483648
基数10による表記。 limit = -2147483648
基数16による表記。 limit = 0x80000000
```

考察▼
・基数 10 における表記では、出力された値は入力した値が逆符号となった"-2147483648"となった。これは下限であるはずの値と同値である。
・よって、int 型整数には"2147483648"が入力した通りには格納されないことが分かった。
・基数 16 における表記は正常に出力されているようだ。
・同様に、上限値であるはずの"2147483647"や上限内である"2147483646"を入力する。

実行結果▼

```
基数10や16で表記したい値を入力して下さい。
2147483647
基数10による表記。 limit = 2147483647
基数16による表記。 limit = 0x7fffffff
```

```
基数10や16で表記したい値を入力して下さい。
2147483648
基数10による表記。 limit = 2147483646
基数16による表記。 limit = 0x7ffffffe
```

考察▼
・基数を問わず、値域に含まれる値については正常に入力され、出力される。
・よって、入力した通りに認識される上限値は"2147483637"である。
・次に下限値である"-2147483648"や、前後の値である"-2147483649"や"-2147483647"を入力する。

a-3:int 型整数の下限の値について、下限値とその前後の数値を入力して観察する。

実行結果▼　※左上、右上、左下の順に"-2147483648"、"-2147483649"、"-2147483647"を入力した。

```
基数10や16で表記したい値を入力して下さい。
-2147483648
基数10による表記。 limit = -2147483648
基数16による表記。 limit = 0x80000000
```

```
基数10や16で表記したい値を入力して下さい。
-2147483649
基数10による表記。 limit = 2147483647
基数16による表記。 limit = 0x7fffffff
```

```
基数10や16で表記したい値を入力して下さい。
-2147483647
基数10による表記。 limit = -2147483647
基数16による表記。 limit = 0x80000001
```

考察▼

・基数を問わず、下限値である"-2147483648"や値域に含まれる"-2147483647"は正常に入力され、出力された。
・よって、入力したとおりに認識される下限値は"-2147483648"である。
・値"-2147483649"を入力した試行では、基底 10 では上限値であるはずの"2147483647"が出力され、
基数 16 では a-1 項で"2147483647"を入力した際に出力された"7fffffff"が出力されている。

a-4:int 型整数の適当な値について、数値を入力して観察する。

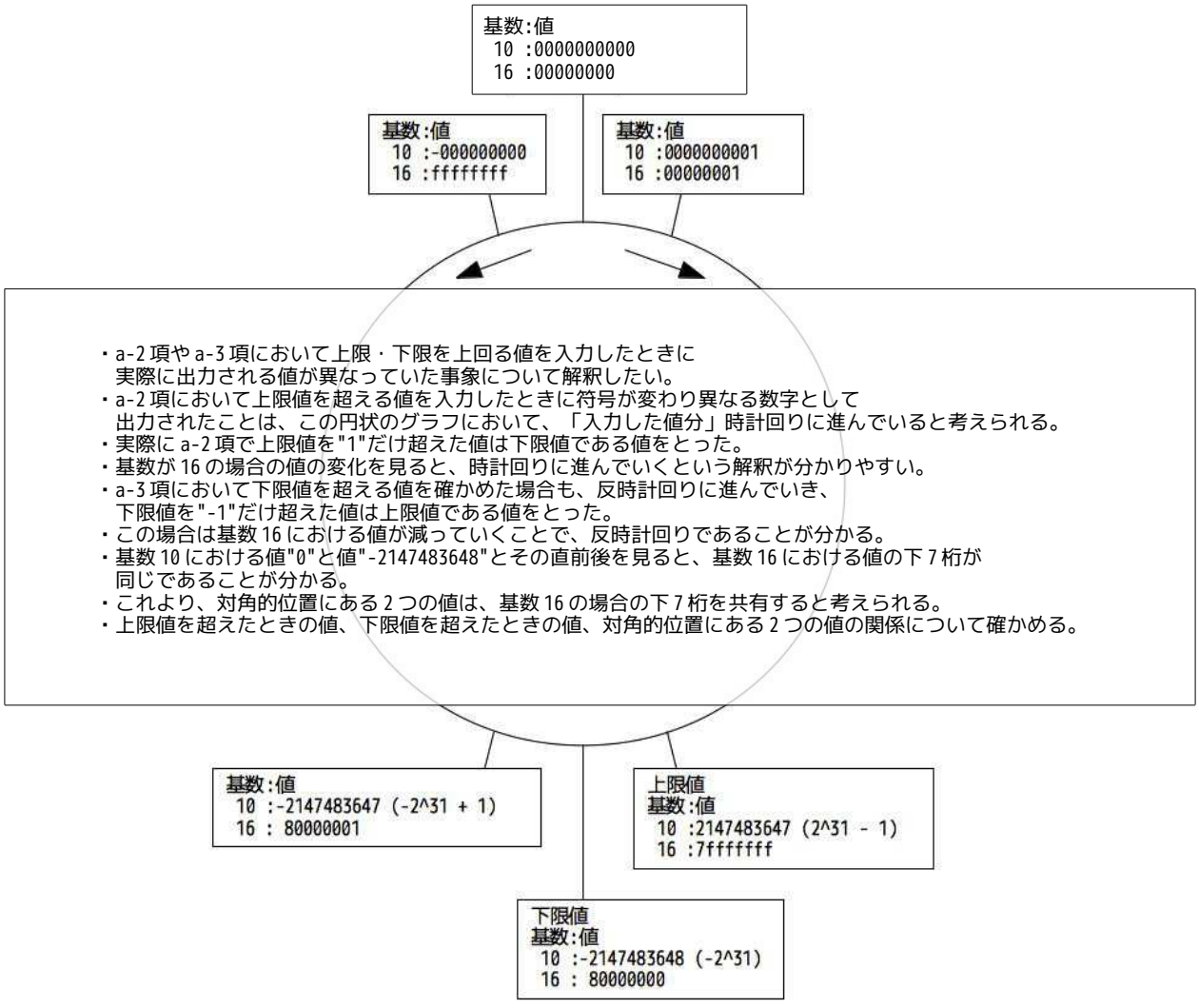
実行結果▼　※左上、右上、左下の順に"-2147483648"、"-2147483649"、"-2147483647"を入力した。

```
基数10や16で表記したい値を入力して下さい。
1
基数10による表記。 limit = 0000000001
基数16による表記。 limit = 0x00000001
```

```
基数10や16で表記したい値を入力して下さい。
-1
基数10による表記。 limit = -000000001
基数16による表記。 limit = 0x7fffffff
```

考察▼

・基数を問わず、下限値である"-2147483648"や値域に含まれる"-2147483647"は正常に入力され、出力された。
・よって、入力したとおりに認識される下限値は"-2147483648"である。
・値"-2147483649"を入力した試行では、基底 10 では上限値であるはずの"2147483647"が出力され、
基数 16 では a-1 項で"2147483647"を入力した際に出力された"7fffffff"が出力されている。
・『Open Lecture』内の「数値のコンピュータの内部表現」に示されているような円図形を使うことで、
一連の結果をまとめたい。



a-5:a-4 項の考察における仮説について検証する。

実行結果▼　※左上、右上、左下、右下の順に"-2"、"2"、"2147483650"、"-2147483650"を入力した。

```
基数10や16で表記したい値を入力して下さい。
-2
基数10による表記。 limit = -000000002
基数16による表記。 limit = 0xfffffffe
```

```
基数10や16で表記したい値を入力して下さい。
2
基数10による表記。 limit = 0000000002
基数16による表記。 limit = 0x00000002
```

```
基数10や16で表記したい値を入力して下さい。
2147483650
基数10による表記。 limit = -2147483646
基数16による表記。 limit = 0x80000002
```

```
基数10や16で表記したい値を入力して下さい。
-2147483650
基数10による表記。 limit = 2147483646
基数16による表記。 limit = 0x7ffffffe
```

考察▼

・上下限値を超えた下 2 つの試行では、入力した値が正なら時計回りに、負なら反時計回りに
仮説通り基底 16 の値が描く円を進んでいることが確認できた。
・また、左上と右下の結果が、右上と左下の結果が対応するように
円において対角的な位置にある、基数 16 における値の下 7 桁が共通していることが確認できた。

iv.エラーについて考察せよ。

a-1.全角文字や特殊な記号による変数宣言。

```
8   #include <stdio.h>
9   int main(){
10
11      int ZA\12 = 1234, ZA/12 = 10000;
12      int レム%ナント, アイウエオ;
13
14      /************************** scanf ***/
15      printf( "Price?   => "); scanf( "%d",&ZA\12);
16      printf( "Payment? => "); scanf( "%d",&ZA/12);
17      printf( "----\n");
18      /************************** Remnant */
19      レム%ナント = ZA/12 - ZA\12;
20      printf( "ZA\12=%d, ",ZA\12);
21      printf( "ZA/12=%d, Remnant=%d\n",ZA/12,レム%ナント);
22      printf( "----\n" );
23      /************************** 5000-yen*/
24      アイウエオ = レム%ナント / 5000;
25      レム%ナント = レム%ナント % 5000;
26      printf( "5000-yen note = %d\n",アイウエオ);
27      return(0);
28
29  }
```

コンパイルする際のエラー▼　※変数"Need"、"COST"にはそれぞれ"298"と"10000"を入力した。

```
ypb4.c: In function ‘main’:
ypb4.c:11: error: stray ‘\’ in program
ypb4.c:11: error: nested functions are disabled, use -fnested-functions to re-enable
ypb4.c:11: error: expected ‘=’, ‘,’, ‘;’, ‘asm’ or ‘__attribute__’ before numeric constant
ypb4.c:11: error: lvalue required as left operand of assignment
ypb4.c:11: error: ‘ZA’ undeclared (first use in this function)
ypb4.c:11: error: (Each undeclared identifier is reported only once
ypb4.c:11: error: for each function it appears in.)
ypb4.c:12: error: expected identifier or ‘(’ before ‘%’ token
ypb4.c:15: error: stray ‘\’ in program
ypb4.c:15: error: expected ‘)’ before numeric constant
ypb4.c:19: error: expected expression before ‘%’ token
ypb4.c:19: error: stray ‘\’ in program
ypb4.c:20: error: stray ‘\’ in program
ypb4.c:20: error: expected ‘)’ before numeric constant
ypb4.c:21: error: expected expression before ‘%’ token
ypb4.c:24: error: expected expression before ‘=’ token
ypb4.c:25: error: expected expression before ‘%’ token
ypb4.c:26: error: expected expression before ‘)’ token
```

エラーの考察▼

・11 行目：プログラム中に余計な"\"があります。
・11 行目：関数のネスト化はできません。 "-fnested"関数を使って下さい。
・11 行目：定数の前に"="、","、";"、"asm"、または"__attribute__"が含まれています。
・11 行目：割り当てられる変数を左辺に置いて下さい。
・11 行目："ZA"は宣言されていません。（初めて使われた関数です）
・12 行目："%"の前の"("がないと予想されます。
・15 行目：定数の前に")"があると予想されます。
・19 行目："%"の前に適切な表現がありません。
・20 行目：定数の前に")"があると予想されます。
・21 行目："%"の前に適切な表現がありません。
・24 行目："="の前に適切な表現がありません。
・25 行目："%"の前に適切な表現がありません。
・26 行目：")"の前に適切な表現がありません。
・ほか、下記のようなエラーが大量に検出された。

同種のエラーと考えられるエラー群の一部▼ ※大量に検出されたため、12行目に対するエラーコードのみ。

```
ypb4.c:12: error: stray ‘\343’ in program
ypb4.c:12: error: stray ‘\203’ in program
ypb4.c:12: error: stray ‘\254’ in program
ypb4.c:12: error: stray ‘\343’ in program
ypb4.c:12: error: stray ‘\203’ in program
ypb4.c:12: error: stray ‘\240’ in program
ypb4.c:12: error: stray ‘\343’ in program
ypb4.c:12: error: stray ‘\203’ in program
ypb4.c:12: error: stray ‘\212’ in program
ypb4.c:12: error: stray ‘\343’ in program
ypb4.c:12: error: stray ‘\203’ in program
ypb4.c:12: error: stray ‘\263’ in program
ypb4.c:12: error: stray ‘\343’ in program
ypb4.c:12: error: stray ‘\203’ in program
ypb4.c:12: error: stray ‘\210’ in program
ypb4.c:12: error: stray ‘\343’ in program
ypb4.c:12: error: stray ‘\202’ in program
ypb4.c:12: error: stray ‘\242’ in program
ypb4.c:12: error: stray ‘\343’ in program
ypb4.c:12: error: stray ‘\202’ in program
ypb4.c:12: error: stray ‘\244’ in program
ypb4.c:12: error: stray ‘\343’ in program
ypb4.c:12: error: stray ‘\202’ in program
ypb4.c:12: error: stray ‘\246’ in program
ypb4.c:12: error: stray ‘\343’ in program
ypb4.c:12: error: stray ‘\202’ in program
ypb4.c:12: error: stray ‘\250’ in program
ypb4.c:12: error: stray ‘\343’ in program
ypb4.c:12: error: stray ‘\202’ in program
ypb4.c:12: error: stray ‘\252’ in program
```

エラーの考察▼
・12行目：プログラム中に余計な"\343"があります。
・12行目：プログラム中に余計な"\203"があります。
・12行目：プログラム中に余計な"\254"があります。

考察▼
・大量に検出された"\xxx"のエラーについては、検出位置から全角文字に対するエラーと判断し
変数名から全角を消去する。
・次に、特殊な記号が変数の先頭や末尾にあった場合を、下記の簡単なプログラムで観察する。

a-2.特殊な記号の位置を変えての観察。

```
8   #include <stdio.h>
9   int main(){
10
11      int \ZA12 = 1234;
12      printf( "変数宣言が終了しました。\n" );
13      printf( "%d\n",\ZA12);
14      return(0);
15
16  }
```

コンパイルする際のエラー▼

```
ypb41.c: In function ‘main’:
ypb41.c:11: error: stray ‘\’ in program
ypb41.c:13: error: stray ‘\’ in program
```

エラーの考察▼
・プログラム中に余計な"\"が含まれています。
考察▼
・上記のプログラムでは変数の先頭に"\"が付いているが、実行後に書き換え、変数末尾に"\"を入力した場合も
試行した。
・"\"については、位置に限らず変数名に含めることはできず、エラー内容についても変化はなかった。
・"%"や"/"についても試行した結果、以下の様なエラーが得られた。

コンパイルする際のエラー▼ ※これは"%"をそれぞれ変数の先頭、末尾に入力した際のエラーである。

```
ypb41.c: In function ‘main’:
ypb41.c:11: error: expected identifier or ‘(’ before ‘%’ token
ypb41.c:13: error: expected expression before ‘%’ token
```

```
ypb42.c: In function ‘main’:
ypb42.c:11: error: nested functions are disabled, use -fnested-functions to re-enable
ypb42.c:11: error: expected ‘=’, ‘,’, ‘;’, ‘asm’ or ‘__attribute__’ before ‘%’ token
ypb42.c:11: error: expected expression before ‘%’ token
ypb42.c:13: error: ‘ZA12’ undeclared (first use in this function)
ypb42.c:13: error: (Each undeclared identifier is reported only once
ypb42.c:13: error: for each function it appears in.)
ypb42.c:13: error: expected expression before ‘)’ token
```

エラーの考察▼
・上段 11 行目："%"の前に"("がないと予想されます。
・上段 13 行目："%"の前に適切な表現がありません。
・下段 11 行目：関数のネスト化はできません。"-fnested"関数を使って下さい。
・下段 11 行目："="、","、";"、"asm"または"__attribute__"が"%"の前にありません。
・下段 11 行目："%"の前に適切な表現がありません。
・下段 13 行目："ZA12"は宣言された変数ではありません。
・下段 13 行目：")"の前に適切な表現がありません。

エラーの考察▼
・"%"や"/"、"&"を変数名の前後に入力した場合も、正常にコンパイルされなかった。
・エラーが提言している通り、変数宣言時に"("、")"を入力してみる。

```
8   #include <stdio.h>
9   int main(){
10
11      int (ZA12%) = 1234;
12      printf( "変数宣言が終了しました。\n" );
13      printf( "%d\n",(ZA12%));
14      return(0);
15
16  }
```

コンパイルする際のエラー▼

```
ypb42.c: In function ‘main’:
ypb42.c:11: error: expected ‘)’ before ‘%’ token
ypb42.c:13: error: ‘ZA12’ undeclared (first use in this function)
ypb42.c:13: error: (Each undeclared identifier is reported only once
ypb42.c:13: error: for each function it appears in.)
ypb42.c:13: error: expected expression before ‘)’ token
```

考察▼
・"("、")"を入力しても、"%"を変数名に含めることはできなかった。
・"/"や"&"に置き換えても、変数名に含めることはできなかった。
・このことから、"\"や"/"、"&"、"%"などの演算子は変数に含めることができないことが分かった。

a-3.全角文字に対するエラーについて考察する。

同種のエラーと考えられるエラー群の一部▼ ※大量に検出されたため、12 行目に対するエラーコードのみ。

```
ypb4.c:12: error: stray ‘\343’ in program
ypb4.c:12: error: stray ‘\203’ in program
ypb4.c:12: error: stray ‘\254’ in program
ypb4.c:12: error: stray ‘\343’ in program
ypb4.c:12: error: stray ‘\203’ in program
ypb4.c:12: error: stray ‘\240’ in program
ypb4.c:12: error: stray ‘\343’ in program
ypb4.c:12: error: stray ‘\203’ in program
ypb4.c:12: error: stray ‘\212’ in program
ypb4.c:12: error: stray ‘\343’ in program
ypb4.c:12: error: stray ‘\203’ in program
ypb4.c:12: error: stray ‘\263’ in program
ypb4.c:12: error: stray ‘\343’ in program
ypb4.c:12: error: stray ‘\203’ in program
ypb4.c:12: error: stray ‘\210’ in program
ypb4.c:12: error: stray ‘\343’ in program
ypb4.c:12: error: stray ‘\202’ in program
ypb4.c:12: error: stray ‘\242’ in program
ypb4.c:12: error: stray ‘\343’ in program
ypb4.c:12: error: stray ‘\202’ in program
ypb4.c:12: error: stray ‘\244’ in program
ypb4.c:12: error: stray ‘\343’ in program
ypb4.c:12: error: stray ‘\202’ in program
ypb4.c:12: error: stray ‘\246’ in program
ypb4.c:12: error: stray ‘\343’ in program
ypb4.c:12: error: stray ‘\202’ in program
ypb4.c:12: error: stray ‘\250’ in program
ypb4.c:12: error: stray ‘\343’ in program
ypb4.c:12: error: stray ‘\202’ in program
ypb4.c:12: error: stray ‘\252’ in program
```

考察▼
・"\xxx"に関するエラーについて考察したい。
・このエラーは、全角文字を入力している部分に表れている。
・前回の printf()関数についてのレポートにおいて同種のエラーが表れたとき、
"%"を表すと考えられる"\045"についてのエラーであった。
・よって、このエラー群中の"\xxx"も何らかの表現方法によって"全角文字"を表すものであると考えられる。
・ここでエラーコードを見ると、3 行ごとに"\343"という表記が表れている。
・3 行 1 組としてみなすと、12 行目に対して 10 組のエラー群が出ている。
・高水準コード 12 行目には全角文字がちょうど 10 文字含まれており、1 組が 1 文字に対応する形であるようだ。
・次のようなプログラムで、この仮説を確かめる。
・エラー先頭から 10 組を抜き出して printf()関数で出力する。
・仮説が正しければ、高水準コード 12 行目にある全角文字、先頭から 5 文字分である"レムナント"と
末尾から 5 文字分である"アイウエオ"が出力されるはずだ。

高水準コードの一部▼　※コード冒頭、コメントアウトによるファイル説明などの行は省略。

```
8   #include <stdio.h>
9   int main(){
10
11      printf( "\343\203\254\343\203\240\343\203\212\343\203\263\343\203\210\n" );
12      printf( "\343\202\242\343\202\244\343\202\246\343\202\250\343\202\252\n" );
13      return(0);
14
15  }
```

実行結果▼

```
レムナント
アイウエオ
```

考察▼

・予測通りの結果が得られた。
・printf()関数において、"\343"の直後に対応する 3 桁の数値を"\"に続く形で 2 組入力することによって
　全角文字を出力できることが分かった。
・"アイウエオ"と出力したときの数値を見ると、3 つ目の"\"に続く数値は、全角一文字ごとに"2"増え、
　かつ"8"を扱っていないようだ。
・この法則が全ての全角文字に適応されるなら、数値から意図する全角文字を出力できるはずだ。
・"レムナント"と出力された部分の"ト"に対応する数値を見ると、"\343\203\210"となっている。
・よって、3 つ目の"\"に続く値が"276"を超えると、2 つ目の"\"に続く値が"1"増えるようだ。
・これらの観察をもとに、次のプログラムで"タチツテト"と出力することで正誤を確認したい。

高水準コードの一部▼　※コード冒頭、コメントアウトによるファイル説明などの行は省略。

```
8   #include <stdio.h>
9   int main(){
10
11      printf( "\343\203\200\343\203\202\343\203\204\343\203\206\343\203\210\n" );
12      return(0);
13
14  }
```

実行結果▼

```
ダヂツテト
```

考察▼

・予想とは異なる結果が得られた。
・これまでの値で"8"が表れていないことから、この 3 桁の値が 8 進数であると考えられる。
・ツテトについては予想通りであるため、"ダ"、"ヂ"を表している値の±2 を出力して確認する。

高水準コードの一部▼　※コード冒頭、コメントアウトによるファイル説明などの行は省略。

```
8   #include <stdio.h>
9   int main(){
10
11      printf( "\343\203\200\343\203\202\343\203\204\343\203\206\343\203\210\n" );
12      printf( "\\343\\202\\276 = \343\202\276, " );
13      printf( "\\343\\202\\277 = \343\202\277\n" );
14      printf( "\\343\\203\\200 = \343\203\200, " );
15      printf( "\\343\\203\\201 = \343\203\201\n" );
16      printf( "\\343\\203\\202 = \343\203\202, " );
17      printf( "\\343\\203\\203 = \343\203\203\n" );
18      printf( "\\343\\203\\204 = \343\203\204\n" );
19      return(0);
20
21  }
```

実行結果▼

```
ダヂツテト
\343\202\276 = ゾ, \343\202\277 = タ
\343\203\200 = ダ, \343\203\201 = チ
\343\203\202 = ヂ, \343\203\203 = ッ
\343\203\204 = ツ
```

考察▼

・得られた結果から、全角文字は"2"ずつではなく、8 進数の値 1 つにつき 1 つの全角文字が対応していることが分かる。
・"2"ずつ増えているように見えた理由は、濁音や小文字などの文字も含まれているためであった。
・最後に、"タチツテト"と入力、出力してエラーの考察を終了したい。

高水準コードの一部▼　※コード冒頭、コメントアウトによるファイル説明などの行は省略。

```
8   #include <stdio.h>
9   int main(){
10
11      printf( "\343\202\277\343\203\201\343\203\204\343\203\206\343\203\210\n" );
12      return(0);
13
14  }
```

実行結果▼

```
タチツテト
```

XXX.あとがき。（反省・感想・その他）

a-1.参考サイト・文献。

・『C 実践プログラミング 第三版』（オーム社）
・『Open Lecture』
 http://www.osn.u-ryukyu.ac.jp/lecture/wiki/index.php?Open%20Lecture

a-2.反省・感想。

今回は今回でびっくりするぐらいページ数が少なくなってしまってびびっています。
ただ、実際に活用価値のあるようなプログラムが書けたことはとても楽しかったです。
出力するだけでなくプログラムに入力することもできるようになり、できる事が増えていくことが嬉しいです。

エラーの考察で、予想通りに"\nnn"で全角文字を出力できたことで、
物凄くテンションが上がってしまいました。
発見をして、それを実行していくことが楽しいということが改めて分かったレポートでした。

最後までご覧いただきありがとうございました。