プログラミング1

Report#02

提出日: 2009 年 5 月 21 日(木)

所属: 工学部情報工学科

学籍番号: 095736E

氏名: 玉城　翔

1. scanf()関数による標準入力と基本演算子

i.例題 balance.c

ソースプログラム

```
/*
  Program    : balance.c
  Student-ID : 095736E
  Author     : TAMASHIRO,Kakeru
  UpDate     : 2009/05/10(Sun)
  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("payment = %d, balance = %d\n",pay,balance);
  printf("----\n");

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

例題の出力結果

```
Price?   => 1234
Payment? => 10000
----
price = 1234, payment = 10000, balance = 8766
----
5000-yen note = 1
```

解説

最初の/*(スラッシュ・アスタリスク)~*/で囲まれた部分は、標題コメントと呼ばれ、このプログラムに関してプログラマに説明すべきあらゆる事項を記述するところです。

次に、int main()内で、int 型(整数型)の表示方法で変数を指定して、それぞれ price,pay,balance, amount と変数名をいれます。また、price と pay の変数には、あらかじめ1234、10000のデータを入れときます。

次に、/**scanf*/内で、printf 関数を使って price と Payment の変数にデータを入力するように促すメッセージを作り、直後の scanf 関数を使って、入力されるデータをそれぞれの変数に代入するプログラムを作る。

次に、/**balance*/内で、pay から price を引いたデータの値を balance に代入する演算をつくり、printf 関数により、それぞれのデータを出力します。

最後に、/**5000-yen*/内で、balance から 5000 のデータを割った商を、amount に代入する演算と、balance から 5000 を割った時の余りを再度、balance に代入する演算を作り、printf 関数により、amount のデータを出力します。

考察

scanf 関数を使うことにより、出力画面に入力したデータをそれぞれの変数に代入するのは便利な機能ではあるが、何も分からずにこのプログラムを使用した時などは、printf 関数に作ったメッセージだけでは入力に戸惑ってしまったり、入力をするのかどうかも分からない場合(自分がそうだった)があるので、そこんところは配慮が必要である。

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。

例題のところでの結果が、お釣りのお札の枚数を求めるプログラムであることを示しているので、別のデータを入力してみようと思う。

a.出力結果

```
--------------------------------------------------------------------------------------------------------------
Price?   => 1498
Payment? => 10000
----
price = 1498, payment = 10000, balance = 8502
----
5000-yen note = 1
--------------------------------------------------------------------------------------------------------------
```

解説

別のデータを入力しても、プログラムは正常に作動しました。

考察

解説の様に、scanf 関数はどんな値でも大丈夫だと思い、試しに小数点を付けて入力したところ、データは小数点が付く前までのデータを受付、後のデータは受け付けなかった。これは、変数を int 型で指定しているためで、改めて int 型の意味を理解できた。

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.ソースプログラム

```
----------------------------------------------------------------------------------------------------------------
*/
 Program    : balance.c
 Student-ID : 095736E
 Author     : TAMASHIRO,Kakeru
 UpDate     : 2009/05/10(Sun)
 Comments   : Payment & Balance
*/

#include <stdio.h>

int main()
{
 int kakaku = 1234, shiharaigaku = 10000;
 int zangaku, maisu;

 /****************************** scanf */
 printf("kakaku?   => "); scanf("%d",&kakaku);
 printf("shiharaigaku? => "); scanf("%d",&shiharaigaku);
 printf("----\n");

 /****************************** zangaku */
 zangaku = shiharaigaku - kakaku;
 printf("kakaku = %d, ",kakaku);
 printf("shiharaigaku = %d, zangaku = %d\n",shiharaigaku,zangaku);
 printf("----\n");

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

 /****************************** 2000-yen */
 maisu  = zangaku / 2000;
 zangaku = zangaku % 2000;
 printf("2000-yen note = %d\n",maisu);

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

 /****************************** 500-yen */
 maisu  = zangaku / 500;
 zangaku = zangaku % 500;
 printf("500-yen note = %d\n",maisu);
```

```
/****************************** 100-yen */
  maisu  = zangaku / 100;
  zangaku = zangaku % 100;
  printf("100-yen note = %d\n",maisu);

  /****************************** 50-yen */
  maisu  = zangaku / 50;
  zangaku = zangaku % 50;
  printf("50-yen note = %d\n",maisu);

  /****************************** 10-yen */
  maisu  = zangaku / 10;
  zangaku = zangaku % 10;
  printf("10-yen note = %d\n",maisu);

/****************************** 5-yen */
  maisu  = zangaku / 5;
  zangaku = zangaku % 5;
  printf("5-yen note = %d\n",maisu);

  /****************************** 1-yen */
  maisu  = zangaku / 1;
  printf("1-yen note = %d\n",maisu);
  return(0);
}
```

---

b.出力結果

```
kakaku?   => 1357
shiharaigaku? => 10000
----
kakaku = 1357, shiharaigaku = 10000, zangaku = 8643
----
5000-yen note = 1
2000-yen note = 1
1000-yen note = 1
500-yen note = 1
100-yen note = 1
50-yen note = 0
10-yen note = 4
5-yen note = 0
1-yen note = 3
```

解説

変数名を変えるということで、4つの変数名をローマ字読みで自分なりの言葉に変えてみました。

“price”=”kakaku”（価格）、 ”pay”=”shiharaigaku”（支払い額）、

”balance”=”zangaku”（残額）、 ”amount”=”maisu”（枚数）

　また、硬貨の枚数を求めるプログラムも作るということで、5000円札以外お札と硬貨の枚数を求めるためのプログラムを追加しました。基本的に出力結果は、例題や問題ａの出力結果の動作と何ら変わりはありません。

考察

　5000円以下のお金の演算方法は、/**5000-yen*/内で求めた余りが、ここでの変数残額に再度代入されます。これはどういうことかというと、最初に求めたお釣りから、5000円を割ったときの商を先に計算し枚数を求め、その後の計算で余りを出すことにより、次の/**1000-yen*/内での変数残額のデータは余りの値なので、この値から1000円を割れば、簡単に枚数が求められるのです。この作業を繰り返すことにより、最小単位の1円までの枚数が求めることができるようになるのです。また、/**1-yen*/内では余りを出す必要がないので余りの演算は必要ありません。

c.工夫…!

c.ソースプログラム

```
/*
  Program    : balance.c
  Student-ID : 095736E
  Author     : TAMASHIRO,Kakeru
  UpDate     : 2009/05/10(Sun)
  Comments   : Payment & Balance
*/

#include <stdio.h>

int main()
{
  int kakaku = 1234, shiharaigaku = 10000;
  int zangaku, maisu;

  /******************************** scanf */
  printf("↓ 下記の文章の後にデータを入力せよ↓ \n");
  printf("いくらですか?   => "); scanf("%d",&kakaku);
  printf("支払い額は? => "); scanf("%d",&shiharaigaku);
  printf("----\n");

  /******************************** zangaku */
  zangaku = shiharaigaku - kakaku;
  printf("価格 = %d 円 ",kakaku);
  printf("支払い額 = %d 円 残額 = %d 円\n",shiharaigaku,zangaku);
  printf("----\n");

  /******************************** 五千円札の枚数 */
  maisu  = zangaku / 5000;
  zangaku = zangaku % 5000;
  printf("伍千円札 = %d 枚\n",maisu);
```

```
 /****************************** 二千円札の枚数 */
 maisu  = zangaku / 2000;
 zangaku = zangaku % 2000;
 printf("弐千円札 = %d 枚\n",maisu);

 /****************************** 千円札の枚数 */
 maisu  = zangaku / 1000;
 zangaku = zangaku % 1000;
 printf("千円札 = %d 枚\n",maisu);

 /****************************** 五百円玉の枚数 */
 maisu  = zangaku / 500;
 zangaku = zangaku % 500;
 printf("伍百円硬貨 = %d 枚\n",maisu);

 /****************************** 百円玉の枚数 */
 maisu  = zangaku / 100;
 zangaku = zangaku % 100;
 printf("百円硬貨 = %d 枚\n",maisu);

 /****************************** 五十円玉の枚数 */
 maisu  = zangaku / 50;
 zangaku = zangaku % 50;
 printf("伍十円硬貨 = %d 枚\n",maisu);

 /****************************** 十円玉の枚数 */
 maisu  = zangaku / 10;
 zangaku = zangaku % 10;
 printf("十円硬貨 = %d 枚\n",maisu);

 /****************************** 五円玉の枚数 */
 maisu  = zangaku / 5;
 zangaku = zangaku % 5;
 printf("伍円硬貨 = %d 枚\n",maisu);

 /****************************** 一円玉の枚数 */
 maisu  = zangaku / 1;
 zangaku = zangaku % 1;
 printf("壱円硬貨 = %d 枚\n",maisu);

 return(0);
}
```

---

c.出力結果

↓ 下記の文章の後にデータ(整数)を入力せよ↓

いくらですか? => 1352

```
支払い額は?  => 10000
----
価格 = 1352 円 支払い額 = 10000 円 残額 = 8648 円
----
伍千円札 = 1 枚
弐千円札 = 1 枚
千円札 = 1 枚
伍百円硬貨 = 1 枚
百円硬貨 = 1 枚
伍十円硬貨 = 0 枚
十円硬貨 = 4 枚
伍円硬貨 = 1 枚
壱円硬貨 = 3 枚
```

解説

例題の考察をもとに下線部分を追加した。また、ローマ字表記の部分や、数字、英語の部分を、日本語表記に全部変え、数字の後に数の単位を付け足しました。

考察

どのような配慮が必要かを考え、直接的な命令で誘導することで、戸惑わないのではと考えた。やはり、私たち日本人には漢字の方が読みやすいことが多いので、日本語表記に直しました。数の単位を付けることにより、より分かりやすい結果になったと思います。

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 ソースプログラム

```
*/
 Program    : numvalue-4.c
 Student-ID : 095736E
 Author     : TAMASHIRO,Kakeru
 UpDate     : 2009/05/14(Thu)
 Comments   : int 型整数の下限・上限
*/

#include <stdio.h>

int main(){

 int i ;

 puts("\"i\"に16進数データを入力せよ");
 printf("i = "); scanf("%x",&i);
 printf("(Hex) i = %09x\n",i);
```

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

  return(0);
}
```

---

a.2 ソースプログラム

```
/*
  Program    : numvalue-5.c
  Student-ID : 095736E
  Author     : TAMASHIRO,Kakeru
  UpDate     : 2009/05/14(Thu)
  Comments   : int 型整数の下限・上限
*/

#include <stdio.h>

int main(){

  int j ;

  puts("\"j\"に10進数データを入力せよ");
  printf("j = "); scanf("%d",&j);
  printf("(Hex) j = %09x\n",j);
  printf("(Dec) j = %09d\n",j);

  return(0);
}
```

---

a.出力画面

```
"i"に16進数データを入力せよ
i = ffff
(Hex) i = 00000ffff
(Dec) i = 000065535
"i"に16進数データを入力せよ
i = ffffff
(Hex) i = 000ffffff
(Dec) i = 016777215
"i"に16進数データを入力せよ
i = ffffffff
(Hex) i = 0ffffffff
(Dec) i = -00000001
"j"に10進数データを入力せよ
j = 2147483647
(Hex) j = 07fffffff
```

(Dec) j = 2147483647
"i"に16進数データを入力せよ
i = 80000000
(Hex) i = 080000000
(Dec) i = -2147483648

解説

　数値の上限と下限を調べる為に、16進数を入力して、そのデータを10進数に変換して、その値が正常に表示されなくなるまでを調べる。
scanf 関数により、16進数のデータを入力するよう促します。
　次の行で、16進数としての値にそのまま変換するプログラムを作り、次の行で、10進数に変換するプログラムを作ります。
　また、10進数を16進数に変換するプログラマも後々使うので作ります。

考察

　出力結果で、16進数の ffff を入力したところ、65535に変換されました。この ffff は16ビット（2バイト）のデータ容量を有し、2^16=65536 のデータを扱えます。コンピュータでは0もデータとして扱い、0から数値は始まって行きますので、2^16-1=65535 が16ビットの数値で扱える最上位数である（下限を0とした時）と捉えることができるので、このプログラムが正常に作動していることも伺える。
　次に、24ビット（3バイト）の ffffff を入力したところ、16777215に変換されました。つまり、24ビットは16777215+1=2^24 のデータを扱えることが分かる。
　それでは、32ビット（4 バイト）の ffffffff を入力するとどうなるか。結果は、-1 が出力された。これはどういうことか。
　整数型（int）の16進数での10進数数値は、0から始まりデータの限界で-1 を表示します。上限と下限は、n バイトのデータ容量を/（割る）2した値-（引く）1を上限とし、/2 した値を負の数（マイナスを付ける）とした値を下限としています。つまり、2^n-1-1 が上限、-(2^n-1)が下限と表すことができる。例として、8ビット（1バイト）の整数型は0から始まり、上限を256/2-1=127 とし、下限を-(256/2)=-128 として-1でおわります。(0,1,2,・・・126,127,-128,-127,・・・-3,-2,-1)
　以上のことを踏まえると、ffffffff の結果は32ビット以上のデータは扱えないということが分かる。それでは、上限と下限はいくらになるのか。
　前述の式を使って、2^31-1 (n-1=32-1)=2147483647 が上限、-(2^31)=-2147483648 が下限ということになる。実際に試して見ると、まず、2147483647の16進数データを調べる為に、10進数を16進数に変換するプログラムで16進数データを調べます。これで、16進数データが分かりました。
　それでは、上限であると予想される16進数 7fffffff の次の 80000000 を入力することにより、下限が出力されるのかどうかを試します。すると、実際に-2147483648 が出力されました。これにより、上限と下限の数値が分かりました。

ⅳ.エラーについて考察せよ。

ソースプログラム

```
/*
  Program    : numvalue-5.c
  Student-ID : 095736E
  Author     : TAMASHIRO,Kakeru
  UpDate     : 2009/05/14(Thu)
  Comments   : int 型整数の下限・上限
*/
```

```
#include <stdio.h>

int main(){

  int j ;

  puts("\"j\"に10進数データを入力せよ");
  printf("j = "); scanf("%d",&i);
  printf("(Hex) j = %09x\n",i);
  printf("(Dec) j = %09d\n",i);

  return(0);
}
```

出力画面

```
numvalue5.c: In function ‘main’:
numvalue5.c:16: error: ‘i’ undeclared (first use in this function)
numvalue5.c:16: error: (Each undeclared identifier is reported only once
numvalue5.c:16: error: for each function it appears in.)
```

解説

変数を”j”で指定したのに、scanf 関数・printf 関数での変数が”i”で指定されたので、プログラムをコンパイルするときに出力画面の様なエラーが表示された。

出力画面では、”i”が変数として指定されておらず、変数を”j”で指定しているので互いが全く機能していないことを表している。

考察

このエラーは、似た様なプログラムを作ろうとして、コピー&ペーストを使って入力し、変数名などを少し変えただけだったので、エラーが出たときには何が何だか分からなかった。コピー&ペーストを使ったあとは、ソースプログラムを隅々まで確認しないと、ちょっとしたことでもエラーが起きてしまう。こういう所を注意しないといけない。

2.反省・感想

お金の計算をするプログラム作りでは、四則演算に関するプログラミングがとても重要な部分であったと思います。特に、%の剰余をうまく利用してあまりをきれいに次の処理に使うという所は、理解するのに時間が掛かってしまった。

数値の上限・下限の調査は、16進数を10進数に変換することによってデータ容量を調べられたり、上限・下限を求めることが出来たりするまでの道のりが、とても険しかったです。