プログラミング　第2回レポート

202111609　仲村和士

2022年6月8日

1　はじめに

1.1　はじめにについて

今回のレポートには、前回と異なり、はじめにの”節”が必要らしい。よって書く内容が特にあるわけではないが、この節を設けた。

1.2　確認

今回ははじめにの節を設けることが要求されている。その中で書く内容については特に要求はない。よって題意を満たす。

2　設問 (1)

2.1　課題概要と方針

本課題は定義されたgetint関数を利用してa, b, cの値を取得するという課題である。その他の条件はプロンプトを半角文字で表示させることである。これらの実装はgetint関数とprintf関数を順次並べることで実現できる。

2.2　実装

早速実装を提示する。以下の実装の処理の流れを説明すると、前述の2つの関数を、main関数でまずプロンプトの表示、次に標準入力となるように交互に呼び出しているだけである。そして最後に標準出力をして終了である。

Listing 1: keisan01.c

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

#define SSIZE 100

int getint(){
  char s[SSIZE];

  return atoi(fgets(s, SSIZE, stdin));
}
int main(void){
    int a,b,c;

    printf("input an interger number(a): ");
    a = getint();
    printf("input an interger number(b): ");
    b = getint();
    printf("input an interger number(c): ");
    c = getint();
    printf("value (a) is %d\n",a);
    printf("value (b) is %d\n",b);
    printf("value (c) is %d\n",c);
}
```

2.3 確認

このプログラムは分岐などはないので一度実行して予期される動作が見られれば十分である。なお、このプログラムの入力として想定されるのは int 型数値である。次に実行結果を示す。

```
$ cc -std=c89 keisan01.c
$ ./a.out
input an interger number(a): 3
input an interger number(b): 5
input an interger number(c): 17
value (a) is 3
value (b) is 5
value (c) is 17
```

2.4 難しかった点など

特になかった。

3　設問 (2)

3.1　課題内容と方針

この課題は (1) で受け取る値を整数 2-9 に制限して、条件に合致する値が得られるまで無限に入力を繰り返す関数 mygetint を作成するという内容である。全体の方針として、C 言語はオブジェクト指向言語ではないので、より扱いやすいコードを作成するには、処理を関数にまとめていくことが最もよいと思われる。よって以降の課題でも積極的に関数を作成する。この設問を解くには、while(1) を用いて無限ループを作り出し、if 文で条件に合致する入力が得られた際にループを抜ければ良い。範囲外のメッセージは else 部に記述する。

3.2　実装

実装を提示する。11 行目から mygetint を定義している。引数に文字列を取っているのは、変数名が必要な入力のプロンプトもまとめてこの関数内で表示するように実装しているからである。他の例を見てもそのほうが自然であると考えられる (例: ログイン認証など[1])。13 行目からは前述の通りの while(1) によるループであり、内部の if-else 文でループを抜けるか、ループを抜けずにメッセージを表示するかを判定する。main 関数は getint + printf を用いた実装だった部分をまるまる mygetint に置き換えている。

Listing 2: keisan02.c

```
1  #include<stdio.h>
2  #include <stdlib.h>
3
4  #define SSIZE 100
5
6  int getint(){
7    char s[SSIZE];
8
9    return atoi(fgets(s, SSIZE, stdin));
10 }
11 int mygetint(char var_name[]){
12   int num;
13   while(1){
14     printf("input an interger number(%s): ",var_name);
15     num = getint();
16     if(2 <= num && num <= 9){
17       return num;
18     }else{
```

[1] ログイン認証やパスワードによる ssh 接続では、ミスがあった際にその旨のメッセージを表示した後、もとのプロンプトを表示する。

```
19        printf("Invalid value. Retype an integer in range
              2-9.\n");
20      }
21    }
22  }
23  int main(void){
24      int a,b,c;
25      a = mygetint("a");
26      b = mygetint("b");
27      c = mygetint("c");
28      printf("value (a) is %d\n",a);
29      printf("value (b) is %d\n",b);
30      printf("value (c) is %d\n",c);
31  }
```

3.3 確認

このプログラムは while ループ部で分岐が発生するので、不正な数値が入力されたときに正しくループをするかの確認が必要である。以下の実行例から、正しく実行できていることが確認できる。このプログラムは設問条件 (2a)-(2c) を満たしている。

```
$ cc -std=c89 keisan02.c
$ ./a.out
input an interger number(a): 3
input an interger number(b): 12
Invalid value. Retype an integer in range 2-9.
input an interger number(b): 1
Invalid value. Retype an integer in range 2-9.
input an interger number(b): 10
Invalid value. Retype an integer in range 2-9.
input an interger number(b): 2
input an interger number(c): 0
Invalid value. Retype an integer in range 2-9.
input an interger number(c): a
Invalid value. Retype an integer in range 2-9.
input an interger number(c): 9
value (a) is 3
value (b) is 2
value (c) is 9
```

3.4　難しかった点など

配列の宣言時に Java の配列の宣言の書式で書いてしまうミスをした程度で、それ以外は特になかった。

4　設問 (3)

4.1　課題内容と方針

本課題の要件は、定義された関数 myrand を用いて式 (A) の範囲の乱数を生成し、n に代入すること、n を用いて式 (A) を表示すること、これまで通り a, b, c を標準入力から得ること、入力された a, b, c を用いて式 (A) が成り立つか判定すること、である。

次に方針を考える。まず myrand(n) は 1 以上 n 以下の数を出力する関数であるが、まだ取り回しが不便である。より扱いやすい関数として関数内にシード値の初期化機能を持ち、乱数の最小値と最大値を指定できる新しい関数 randint(a,b) を定義しよう。これは a 以上 b 以下の乱数を生成する関数であり、myrand を引数に注意して平行移動する形で実装する。残りの部分は main 関数に順次実装する。n を用いた式の表示は printf で行い、a, b, c を用いた式の判定は if-else で行う。

4.2　実装

以下に実装例を提示する。26 行目から新しい関数 randint を定義している。まずシード値の初期化を行い、次の分岐は 0 割りの除外である[2]0 割りが起こらないことが確認できたら、引数に注意して myrand を平行移動している。乱数を生成する関数はデバッグが難しいのでここで論理的に正しいのかを確認しよう。まず、myrand が最小値を取るとすると、myrand から 1 が返る。よって全体の返り値は a となり正しい。次に myrand が最大値を取ると仮定する。そのとき、myrand からは b-a+1 が返り、全体の返り値は b となる。よって intrand は a 以上 b 以下の値を返すことが確認できる。

この関数を用いて main を実装する。まず式 (A) の値域を考える。最小値は $a = b = c = 2$ のとき 6、最大値は $a = b = c = 9$ のとき 90 である。よって n にはこの範囲の乱数を randint で与える。次に printf で式 (A) を表示している。n の部分はフォーマットで値を挿入している。その後の標準入力はこれまで通りである。最後に if-else で式が成り立つか判定して、それに応じたメッセージ good!/bad!を表示している。

[2]もっとも、0 割りが発生する呼び出しが起こったとすれば、それは乱数を生成する状況でないので無意味であるが。

Listing 3: keisan03.c

```c
#include<stdio.h>
#include <stdlib.h>
#include<time.h>

#define SSIZE 100

int getint(void){
  char s[SSIZE];
  return atoi(fgets(s, SSIZE, stdin));
}
int mygetint(char var_name[]){
  int num;
  while(1){
    printf("input an interger number(%s): ",var_name);
    num = getint();
    if(2 <= num && num <= 9){
      return num;
    }else{
      printf("Invalid value. Retype an integer in range
          2-9.\n");
    }
  }
}
int myrand(int n){
  return rand()%n +1;
}
int randint(int a, int b){
  srand(time(NULL));
  if(a == b){
    return a;
  }else{
    return a-1 + myrand(b-a+1);
  }
}
int main(void){
    int a,b,c;
    int n;
    n = randint(6,90);
    printf("a * b + c = %d\n", n);
    a = mygetint("a");
    b = mygetint("b");
    c = mygetint("c");

    if(a * b + c == n){
      printf("good!\n");
    }else{
```

```
46          printf("bad!\n");
47      }
48  }
```

4.3 確認

randint の正当性はすでに確認したので、あとは main における if の分岐が正しく実行できることを確認すればよい。以下の実行結果から、if が正しく機能していることと、乱数と思われる数が生成されていることが確認できる。よって設問要件を満たしている。

```
$ cc -std=c89 keisan03.c
$ ./a.out
a * b + c = 39
input an interger number(a): 6
input an interger number(b): 6
input an interger number(c): 3
good!
$ ./a.out
a * b + c = 48
input an interger number(a): 4
input an interger number(b): 4
input an interger number(c): 4
bad!
$ ./a.out
a * b + c = 29
input an interger number(a): 5
input an interger number(b): 5
input an interger number(c): 4
good!
```

4.4 難しかった点など

//でコメントを書いたら、C++タイプのコメントは C89 では使えないとコンパイラに怒られ、やるせない気持ちで修正したという事案が発生した程度である。なお、コード挿入に利用している listings パッケージでは日本語のコメントはうまく表示できないようなのでコメントはレポートに挿入する時点で削除した。

5　設問 (4)

5.1　課題内容と方針

本課題は全問のプログラムを改造して、最後の式判定の際に成り立つ答えが得られるまで繰り返すという内容である。

方針としては入力、判定の部分を while(1) でループさせて式判定で good!を出力後に break する。また、本設問には直接関係ないが、ここまでの処理を play() という関数にまとめて取り回しやすくする。

5.2　実装

以下に実装例を提示する。まず、ここまで main に書かれてきた処理内容はすべて 33 行目からの play 関数に移した。そのコードに対し、38 から 49 行目の while ループを追加し、if 文の then 部に break を書いた。main 関数では play 関数を呼び出して終了である。

Listing 4: keisan04.c

```
1  #include<stdio.h>
2  #include <stdlib.h>
3
4  #define SSIZE 100
5
6  int getint(void){
7    char s[SSIZE];
8    return atoi(fgets(s, SSIZE, stdin));
9  }
10 int mygetint(char var_name[]){
11   int num;
12   while(1){
13     printf("input an interger number(%s): ",var_name);
14     num = getint();
15     if(2 <= num && num <= 9){
16       return num;
17     }else{
18       printf("Invalid value. Retype an integer in range
           2-9.\n");
19     }
20   }
21 }
22 int myrand(int n){
23   return rand()%n +1;
24 }
25 int randint(int a, int b){
```

```
26    srand(time(NULL));
27    if(a == b){
28      return a;
29    }else{
30      return a-1 + myrand(b-a+1);
31    }
32  }
33  void play(void){
34      int a, b, c;
35      int n;
36      n = randint(6,90);
37      printf("a * b + c = %d\n",n);
38      while(1){
39        a = mygetint("a");
40        b = mygetint("b");
41        c = mygetint("c");
42
43        if(a * b + c == n){
44          printf("good!\n");
45          break;
46        }else{
47          printf("bad!\n");
48        }
49      }
50  }
51  int main(void){
52      play();
53  }
```

5.3 確認

誤答をしたときに繰り返しをして、正答で即座に終了することを確認すればよい。以下の実行結果は要件に合致する。

```
$ cc -std=c89 keisan04.c
$ ./a.out
a * b + c = 12
input an interger number(a): 2
input an interger number(b): 4
input an interger number(c): 4
good!
$ ./a.out
a * b + c = 47
input an interger number(a): 3
```

```
input an interger number(b): 3
input an interger number(c): 3
bad!
input an interger number(a): 4
input an interger number(b): 4
input an interger number(c): 4
bad!
input an interger number(a): 5
input an interger number(b): 5
input an interger number(c): 5
bad!
input an interger number(a): 5
input an interger number(b): 9
input an interger number(c): 4
bad!
input an interger number(a): 5
input an interger number(b): 8
input an interger number(c): 7
good!
```

5.4 難しかった点など

特になかった。

6 設問 (5)

6.1 課題内容と方針

本課題の要件は、今までは式 (A) を対象にしていたが、式 (B) を対象にしたプログラムを作成することである。

方針としては、全問の play をまず複製して play_B とする。それに伴い、play は play_A とする。そして、式に依存している部分として、n の範囲と式の printf, 正誤判定の式を変更する。

6.2 実装

以下に実装例を提示する。変更したのは 60 行目の n の値の範囲、61 行目の printf、70 行目の条件式である。まず、60 行目の n の値の範囲を考える。最小値は問題文

から 0 である。最大値は $a = b = 9, c = 2$ のとき 79 である。これを randint の引数とする。次に 61 行目の printf は文字列を置き換えるだけであり、70 行目の条件式も式を置き換えるだけである。main 関数では今度は play_B 関数を呼び出して終了である

Listing 5: keisan05.c

```
1  #include<stdio.h>
2  #include <stdlib.h>
3
4  #define SSIZE 100
5
6  int getint(void){
7    char s[SSIZE];
8    return atoi(fgets(s, SSIZE, stdin));
9  }
10 int mygetint(char var_name[]){
11   int num;
12   while(1){
13     printf("input an interger number(%s): ",var_name);
14     num = getint();
15     if(2 <= num && num <= 9){
16       return num;
17     }else{
18       printf("Invalid value. Retype an integer in range
           2-9.\n");
19     }
20   }
21 }
22 int myrand(int n){
23   return rand()%n +1;
24 }
25 int randint(int a, int b){
26   srand(time(NULL));
27   if(a == b){
28     return a;
29   }else{
30     return a-1 + myrand(b-a+1);
31   }
32 }
33 void play_A(void){
34     int a, b, c;
35     int n;
36     n = randint(6,90);
37     printf("a * b + c = %d\n",n);
38     while(1){
39       a = mygetint("a");
40       b = mygetint("b");
```

```
41        c = mygetint("c");
42
43        if(a * b + c == n){
44          printf("good!\n");
45          break;
46        }else{
47          printf("bad!\n");
48        }
49      }
50  }
51  void play_B(void){
52      int a, b, c;
53      int n;
54      n = randint(0, 79);
55      printf("a * b - c = %d\n",n);
56      while(1){
57        a = mygetint("a");
58        b = mygetint("b");
59        c = mygetint("c");
60
61        if(a * b - c == n){
62          printf("good!\n");
63          break;
64        }else{
65          printf("bad!\n");
66        }
67      }
68  }
69  int main(void){
70      play_B();
71  }
```

6.3 確認

変更した部分について特に確認すればよい。まず 61 行目の printf の結果で n の値の範囲と書き換えた式を確認できる。あとは if 文での正誤判定が妥当かを確認すればよい。以下の実行結果から要件を満たしていることが確認できる。

```
$ cc -std=c89 keisan05.c
$ ./a.out
a * b - c = 30
input an interger number(a): a
Invalid value. Retype an integer in range 2-9.
input an interger number(a): 6
```

```
  input an interger number(b): 6
  input an interger number(c): 6
  good!
  $ ./a.out
  a * b - c = 27
input an interger number(a): 4
input an interger number(b): 4
input an interger number(c): 4
bad!
input an interger number(a): 5
input an interger number(b): 5
input an interger number(c): 5
bad!
input an interger number(a): 5
input an interger number(b): 5
input an interger number(c): 6
bad!
input an interger number(a): 5
input an interger number(b): 6
input an interger number(c): 3
good!
```

6.4 難しかった点など

特になかった。

7 設問 (6)

7.1 課題内容と方針

課題の要件は乱数により、式 (A) か式 (B) かを決定して動作することと、絶対に正答が得られない n の値が出たときには求め直すことである。

方針としては、まず、main 関数は randint(0,1) で式を選ぶ乱数を出し、if-else で play_A か play_B を走らせる。play_A と play_B 内の n の値を決めるプロセスでは do-while で例外値が該当しなくなるまで乱数生成を繰り返す。ここで、ループで短期間に繰り返し乱数 seed をリセットする可能性がある場合、1 秒待たないと同じ数値が生成され続ける可能性に気づいたのでここでコードを変更する。[3]

[3] あまりないと思うが、念のため。これまでは呼び出し間隔が 1 秒以上開く状況であったため、この問題が発生する可能性は低い。

7.2 実装

以下に実装例を提示する。まず、play_A と play_B 関数は、do-while で n の値が例外に該当する間は乱数生成をやり直し続けるようにした。do-while を使うのは、一度は必ず実行が必要だからである。randint 関数からは srand を削除した。これは main 関数内で一度だけ呼び出されるように変更する。main 関数では、まず、どちらの式を使うのかの値を入れる変数 rand_expression を宣言する。そして、srand で乱数の seed 値を初期化してから、乱数として 0 または 1 を代入している。あとは、rand_expression の値に応じて play_A または play_B を呼び出す。

Listing 6: keisan06.c

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

#define SSIZE 100

int getint(void){
  char s[SSIZE];
  return atoi(fgets(s, SSIZE, stdin));
}
int mygetint(char var_name[]){
  int num;
  while(1){
    printf("input an interger number(%s): ",var_name);
    num = getint();
    if(2 <= num && num <= 9){
      return num;
    }else{
      printf("Invalid value. Retype an integer in range
          2-9.\n");
    }
  }
}
int myrand(int n){
  return rand()%n +1;
}
int randint(int a, int b){
  if(a == b){
    return a;
  }else{
    return a-1 + myrand(b-a+1);
  }
}
void play_A(void){
    int a, b, c;
    int n;
```

```
35      do{
36          n = randint(6,90);
37      }while(n == 82);
38      printf("a * b + c = %d\n",n);
39      while(1){
40        a = mygetint("a");
41        b = mygetint("b");
42        c = mygetint("c");
43
44        if(a * b + c == n){
45          printf("good!\n");
46          break;
47        }else{
48          printf("bad!\n");
49        }
50      }
51  }
52  void play_B(void){
53      int a, b, c;
54      int n;
55      do{
56          n = randint(0, 79);
57      }while(n == 71);
58      printf("a * b - c = %d\n",n);
59      while(1){
60        a = mygetint("a");
61        b = mygetint("b");
62        c = mygetint("c");
63
64        if(a * b - c == n){
65          printf("good!\n");
66          break;
67        }else{
68          printf("bad!\n");
69        }
70      }
71  }
72  int main(void){
73      int rand_expression;
74      srand(time(NULL));
75      rand_expression = randint(0,1);
76      if(rand_expression == 0){
77          play_A();
78      }else{
79          play_B();
80      }
81  }
```

7.3 確認

ここで特に確認すべきことは、両方の式がランダムに登場するかどうかである。以下では中断を繰り返して両方の値が登場するか確認している。また、どちらの式が与えられても正しく動作することが確認できる。確率的に低いので例外の数値が出ないかは確認しにくいが、より例外が発生しやすい状況で do-while のテストコードを書いて確かに例外の数値が出ないことを確かめたので実装としては問題ない。

```
$ cc -std=c89 keisan06.c
$ ./a.out
a * b - c = 72
input an interger number(a): ^C
$ ./a.out
a * b - c = 65
input an interger number(a): ^C
$ ./a.out
a * b - c = 33
input an interger number(a): ^C
$ ./a.out
a * b + c = 45
input an interger number(a): ^C
$ ./a.out
a * b + c = 29
input an interger number(a): ^C
$ ./a.out
a * b - c = 17
input an interger number(a): 4
input an interger number(b): 5
input an interger number(c): 3
good!
$ ./a.out
a * b + c = 33
input an interger number(a): 9
input an interger number(b): 3
input an interger number(c): 6
good!
```

7.4　難しかった点など

前のコードを改造する中でコピーした部分を 1 行だけ消し忘れてデバッグに手間取った。

8　設問 (7)

8.1　課題内容と方針

最後の課題の要件はコマンドライン変数を 1 つ取り、その回数だけ繰り返す、ただし、コマンドライン変数が 1 つでない場合は終了することである。

方針としては、まず argc でコマンドライン変数の長さを確認して、argc が 2 でなければ終了する。なぜなら、コマンドライン引数の 0 番目 argv[0] にはプログラム名を表す文字列など[4]ユーザーが与えるコマンドライン引数とは別のものが入るからである。argc が 2 のときは atoi 関数で文字列引数を int に変換する。あとは、前問で書いた処理を for 文で規定回数実行するだけである。

8.2　実装

以下に実装例を提示する。今回の実装は main 関数だけである。前問までとは異なり、main には 2 つ引数を取る。本文ではまず変数を宣言したあと、まず argc の大きさを確認して、コマンドライン引数が 1 つであることを確認する。ここでコマンドライン変数が 1 つでなかった場合はわかりやすいようにメッセージを表示して、return 1 で終了する。return 1 は、多くの場合、異常終了として扱われている。コマンドライン変数が 1 つのときは次に進む。次は、コマンドライン引数が入っている argv[1] を数値に変換して変数 rep に格納する。次に処理はおまけであり、rep が 0 のときもわかりやすくメッセージを残して異常終了する。この部分はなくてもメッセージが出ないだけの違いしかない。最後に for 文で前問までの処理を rep の大きさだけ回している。

Listing 7: keisan07.c

```
1 #include<stdio.h>
2 #include <stdlib.h>
3
4 #define SSIZE 100
5
6 int getint(void){
7   char s[SSIZE];
```

[4]「など」と書いたのは、稀なケースだが、処理系や状況によっては異なることがあるからである

```
8   return atoi(fgets(s, SSIZE, stdin));
9 }
10 int mygetint(char var_name[]){
11   int num;
12   while(1){
13     printf("input an interger number(%s): ",var_name);
14     num = getint();
15     if(2 <= num && num <= 9){
16       return num;
17     }else{
18       printf("Invalid value. Retype an integer in range
           2-9.\n");
19     }
20   }
21 }
22 int myrand(int n){
23   return rand()%n +1;
24 }
25 int randint(int a, int b){
26   if(a == b){
27     return a;
28   }else{
29     return a-1 + myrand(b-a+1);
30   }
31 }
32 void play_A(void){
33     int a, b, c;
34     int n;
35     do{
36         n = randint(6,90);
37     }while(n == 82);
38     printf("a * b + c = %d\n",n);
39     while(1){
40       a = mygetint("a");
41       b = mygetint("b");
42       c = mygetint("c");
43
44       if(a * b + c == n){
45         printf("good!\n");
46         break;
47       }else{
48         printf("bad!\n");
49       }
50     }
51 }
52 void play_B(void){
53     int a, b, c;
```

```
54      int n;
55      do{
56          n = randint(0, 79);
57      }while(n == 71);
58      printf("a * b - c = %d\n",n);
59      while(1){
60        a = mygetint("a");
61        b = mygetint("b");
62        c = mygetint("c");
63
64        if(a * b - c == n){
65          printf("good!\n");
66          break;
67        }else{
68          printf("bad!\n");
69        }
70      }
71  }
72  int main(int argc, char** argv){
73      int rand_expression;
74      srand(time(NULL));
75      int rep;
76      int i=0;
77      if(argc != 2){
78          printf("Invalid argment(s)!\n");
79          return 1;
80      }
81      rep = atoi(argv[1]);
82      if(rep == 0){
83          printf("Invalid argment(s)!\n");
84          return 1;
85      }
86      for(i=0; i< rep; i++){
87          rand_expression = randint(0,1);
88          if(rand_expression == 0){
89              play_A();
90          }else{
91              play_B();
92          }
93      }
94  }
```

8.3 確認

確認すべき部分は引数が不正な場合、ちゃんと終了するか、ということと、適切な引数を与えた場合、要求された回数の処理が回るかである。実行結果から、正

しく処理できていることが確認できる。

```
$ cc -std=c89 keisan07.c
$ ./a.out
Invalid argment(s)!
$ ./a.out 1 3 4
Invalid argment(s)!
$ ./a.out 4
a * b - c = 67
input an interger number(a): 8
input an interger number(b): 9
input an interger number(c): 5
good!
a * b + c = 75
input an interger number(a): 8
input an interger number(b): 9
input an interger number(c): 3
good!
a * b + c = 18
input an interger number(a): 3
input an interger number(b): 5
input an interger number(c): 3
good!
a * b - c = 67
input an interger number(a): 8
input an interger number(b): 9
input an interger number(c): 5
good!
```

8.4 難しかった点など

C89 では for 文の内部で変数を宣言できないとコンパイラに怒られたが、それ以外は特になかった。

9 感想

私は C 言語は Python ほどは使えないが、全く経験が無いわけではなかったので今回の内容にそれほど苦労はなかった。それゆえ、特に記述するべき参考文献はない (よって参考文献リストがないのは不備ではない)。昔は C 系の言語はエラー

をたくさん出しながら書いていた記憶があるが、今回はあまりエラーがなかったので多少の成長を感じられた。どちらかというと、C89 に起因するコンパイルエラーが多くそのたびにやるせない気持ちになったが、C89 は私が小学生の頃に読んだ入門書より古い内容であるから、手になじまないのも仕方ないかもしれない。

しかし、私の友人の一言で考えが変わった。曰く、「C89 の頃の C はアセンブリとコードが逐次翻訳の域を出ていなかった」と。なるほど、古いとはいえ、アセンブリに比べれば大分マシな代物である。四則演算も勝手にやってくれるし、関数の引数も勝手にスタックに積んでくれる。あえて C89 を学習する意味があるのかはわからないが、アセンブリを学習するのと同列の意味合いなのかもしれない。