プログラミングⅠ

提出日：2013 年 5 月 23 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

1. sample#1.c を解析し、ASCⅡコード(0x00~0x7f)の各範囲(Scope)を判断するプログラムを作成せよ。

1-a.if 文を用いた ASCⅡコードの各範囲判断

高水準コードの一部

```
01 #include <stdio.h>
02
03 #define FALSE 0 /*FALSE を定義*/
04 #define TRUE  !FALSE /*FALSE 以外を定義*/
05
06 int main(){
07   int  value,c, count;/*変数宣言*/
08   char line[128]; /*文字型の配列宣言*/
09
10   count = 0;/*初期値*/
11
12   while(TRUE){ /*TRUE であれば{}内の処理を繰り返す*/
13     count++; /*インクリメント*/
14     if(count > 5) break; /*無限ループを抜ける条件*/
15
16     printf("Enter a HexValue ===> "); /*十六進数を入力*/
17     fgets(line, sizeof(line), stdin); /*キーボードから入力*/
18     sscanf(line, "%x", &c); /*16 進数に変換*/
19
20     printf("Colum=%02d:%%d(%3d)-%%x(%2x)",count,c,c); /*それぞれに代入*/
21
22     if(0x20 <= c && c <= 0x7e)
23       printf("-%%c(%c)¥n",c);
24     if(c <= 0x1f && 0x7f <= c)
25       printf("-Not Printable character¥n");
26
27     printf("¥t¥t¥t    ");
28       if(0x00 <= c && c <= 0x1f){
29       printf("====> Scope_control code¥n");
30     } if(0x7f == c){
31       printf("====> Scope_control code¥n");
32     } if(0x20 <= c && c <= 0x2f){
33       printf("====> Scope_symbol¥n");
34     } if(0x3a <= c && c <= 0x40){
35       printf("====> Scope_symbol¥n");
36     } if(0x5b <= c && c <= 0x60){
37       printf("====> Scope_symbol¥n");
38     } if(0x7b <= c && c <= 0x7e){
39       printf("====> Scope_symbol¥n");
40     } if(0x30 <= c && c <= 0x39){
41      printf("====> Scope_figuers¥n");
42     } if(0x41 <= c && c <= 0x5a){
43       printf("====> Scope_capital letter¥n");
44     } if(0x61 <= c && c <= 0x7a){
45       printf("====> Scope_smoll letter¥n");
46     }
47   }
48
49   return(0);
50 }
```

1-a-1. 実行結果

```
Enter a HexValue ===> 0
Colum=01:%d(  0)-%x( 0)                              ====> Scope_control code
Enter a HexValue ===> 21
Colum=02:%d( 33)-%x(21)-%c(!)
                                     ====> Scope_symbol
Enter a HexValue ===> 30
Colum=03:%d( 48)-%x(30)-%c(0)
                                     ====> Scope_figuers
Enter a HexValue ===> 41
Colum=04:%d( 65)-%x(41)-%c(A)
                                     ====> Scope_capital letter
Enter a HexValue ===> 61
Colum=05:%d( 97)-%x(61)-%c(a)
                                     ====> Scope_smoll letter
```

1-a-2. 考察

イ. if 文で作成したプログラムで control_code 以外の範囲判断はうまく出来た。

ロ. control_code の範囲に定めた数字を入力しても”-not printable character”が出力されなかった。

ハ. 24 行 18 桁の論理積'&&'は複数の命題が真である時のみ真であることから、どちらかにしか当てはまらず、偽としてみなされスルーされたと思われる。

ニ. 24 行 18 桁の論理積'&&'を論理和'||'にかえる事で改善が出来るはずである。

ホ. if 文を多用し過ぎている。

ヘ. 28~39 行目の scope_symbol と scope_control code の条件式をまとめる為に論理和(OR)や選択候補の区切りに有効な'|'を用いる。(今回は選択候補の区切りに用いる)

1-a の 28~39 行目を改善した高水準コードの一部

(-Not printable character の出力条件行)

```
   if(c <= 0x1f || 0x7f <= c)
     printf("-Not Printable character¥n");
```

(scope_control code と scope_symbol の出力条件行)

```
     if(0x00 <= c && c <= 0x1f | 0x7f == c){
     printf("====> Scope_control code¥n");
   } if(0x20 <= c && c <= 0x2f | 0x3a <= c && c <= 0x40 | 0x5b <= c && c <= 0x60 | 0x7b <= c && c <= 0x7e){
     printf("====> Scope_symbol¥n");}
```

1-a-3. 考察

イ. control_code の範囲に定めた数字を入力すると実行結果ではうまく”Not printable characre”と出力する事が出来た。

ロ. 複数の条件式を'|'で区切ってまとめても、改善前と同様に出力することが

出来た。

1-b.if else 文をネストして用いた ASCⅡ コードの各範囲判断

1-b-1. 高水準コードの一部

```
    1       #include <stdio.h>
    2
    3       #define FALSE 0 /*FALSE を定義*/
    4       #define TRUE  !FALSE /*FALSE 以外を定義*/
    5
    6       int main(){
    7         int  value,c, count;/*変数宣言*/
    8         char line[128]; /*文字型の配列宣言*/
    9
   10         count = 0;/*初期値*/
   11
   12         while(TRUE){ /*TRUE であれば{}内の処理を繰り返す*/
   13           count++; /*インクリメント*/
   14           if(count > 5) break; /*無限ループを抜ける条件*/
   15
   16           printf("Enter a HexValue ===> "); /*十六進数を入力*/
   17           fgets(line, sizeof(line), stdin); /*キーボードから入力*/
   18           sscanf(line, "%x", &c); /*16 進数に変換*/
   19
   20           printf("Colum=%02d:%%d(%3d)-%%x(%2x)",count,c,c); /*それぞれに代入*/
   21
   22           if(0x20 <= c && c <= 0x7e)
   23             printf("-%%c(%c)¥n",c);
   24           else
   25             printf("-Not Printable character¥n");
   26
   27           printf("¥t¥t¥t    ");
   28             if(0x00 <= c && c <= 0x1f | 0x7f == c){
   29             printf("====> Scope_control code¥n");
   30            }else{
   31                 if(0x20 <= c && c <= 0x2f | 0x3a <= c && c <= 0x40 | 0x5b <= c && c <= 0x60 | 0x7b <= c
&& c <= 0x7e){
   32                    printf("====> Scope_symbol¥n");
   33                 }else{
   34                      if(0x30 <= c && c <= 0x39){
   35                         printf("====> Scope_figuers¥n");
   36                           }else{
   37                            if(0x41 <= c && c <= 0x5a){
   38                               printf("====> Scope_capital letter¥n");
   39                                 }else{
   40                                                 if(0x61 <= c && c <= 0x7a){
   41                                    printf("====> Scope_smoll letter¥n");
   42                                  }
   43                                           }
   44                                       }
   45                           }
   46                 }
   47         }
   48
   49
   50
   51         return(0);
   52       }
```

1-b-2. 実行結果

```
Enter a HexValue ===> 0
Colum=01:%d(  0)-%x( 0)-Not Printable character
                                        ====> Scope_control code
Enter a HexValue ===> 21
Colum=02:%d( 33)-%x(21)-%c(!)
                                        ====> Scope_symbol
Enter a HexValue ===> 30
Colum=03:%d( 48)-%x(30)-%c(0)
                                        ====> Scope_figuers
Enter a HexValue ===> 41
Colum=04:%d( 65)-%x(41)-%c(A)
                                        ====> Scope_capital letter
Enter a HexValue ===> 61
Colum=05:%d( 97)-%x(61)-%c(a)
                                        ====> Scope_smoll letter
```

1-b-3. 考察

イ. if-else 文のなかに if-else 文を書くこと(ネスト)で ASCⅡコードの各範囲判断をするプログラムを作成した。

ロ. if-else 文は if 文の条件式で真である場合に処理される文と偽である場合に処理される else 文が存在するので、この性質を利用し else 文にまた if-else 文を入れる事で処理させる文を操作することが出来る。

2. sample#2.c のプログラムの動作を考察せよ。

```
01 #include <stdio.h>
02
03 #define FALSE 0
04 #define TRUE  !FALSE
05
06 int main(){
07   int  count;
08
09   count = 0;
10   while(TRUE){
11     count++;
12     if(count > 5) break;
13     printf("While-Count=%2d¥n",count);
14   }
15
16   for(count=1; count<=5; count++){
17     printf("for  -Count=%2d¥n",count);
18   }
19
20   return(0);
21 }
```

2-a-1. 実行結果

```
While-Count= 1
While-Count= 2
While-Count= 3
While-Count= 4
While-Count= 5
for  -Count= 1
for  -Count= 2
for  -Count= 3
for  -Count= 4
for  -Count= 5
```

2-a-2. 考察

イ. for 文と while 文を用いて条件を満たしている間{}内の処理を繰り返している。

ロ. sample#2.c での for 文と while 文は同価値の結果を出している。

ハ. 教科書を参考にすると、for 文と while 文の違いは繰り返し回数が決まっているかどうかの違いによるということなのでそれぞれを使い分ける。

備考

for 文の書式

```
for(式 1; 条件; 式 2){
……文……;
}
```

イ. 式 1 はループに入る前に 1 度だけ実行され、通常は変数をどの値で初期化するかに用いる。

ロ. 条件を満たしている間ループを繰り返す。

ハ. 式 2 は{}内の文が処理される度に実行され、通常はループする変数のカウントアップに使用する。

while 文の書式

```
while(条件){
……文……;
}
```

イ. 条件を満たしている間ループを繰り返す。

2-b. while 文の適用

2 の何乗で 10 億にとぶか求めるプログラムの作成

2-b-1. 高水準コードの一部

```
01 #include <stdio.h>
02
03 #define FALSE 0
04 #define TRUE  !FALSE
05
06 int main(){
07   int  count,colum;
08
09   count = 1,colum = 0;
10   while(TRUE){
11     count=count*2,colum++;
12     if(count > 1000000000) break;
13     if(colum > 100) break;
14     printf("2^%02d=%010d¥n",colum,count);
15
16   }
17
18   return(0);
19 }
```

2-b-2. 実行結果

```
2^01=0000000002
2^02=0000000004
2^03=0000000008
2^04=0000000016
2^05=0000000032
2^06=0000000064
2^07=0000000128
2^08=0000000256
2^09=0000000512
2^10=0000001024
2^11=0000002048
2^12=0000004096
2^13=0000008192
2^14=0000016384
2^15=0000032768
2^16=0000065536
2^17=0000131072
2^18=0000262144
2^19=0000524288
2^20=0001048576
2^21=0002097152
2^22=0004194304
2^23=0008388608
2^24=0016777216
2^25=0033554432
2^26=0067108864
2^27=0134217728
2^28=0268435456
2^29=0536870912
```

2-b-3. 考察

イ. while 文が適正するプログラムを作成する事が出来た。

ロ. 2 の 30 乗で 10 億にとぶことが分かる。

2-c. for 文の適用

‘A’から’Z’までの文字を横に表示するプログラムの作成

2-c-1. 高水準コードの一部

```
 1      #include <stdio.h>
 2
 3      #define FALSE 0
 4      #define TRUE  !FALSE
 5
 6      int main(){
 7        int  count;
 8
 9        count = 0;
10        for(count=65; count<=90; count++){
11          printf("%2c",count);
12        }
13        printf("¥n");
14        return(0);
15      }
```

2-c-2. 実行結果

```
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
```

2-c-3. 考察

イ. for 文が適正するプログラムを作成する事が出来た。

3. sample#3.c を解析し、表示可能な文字による ASCⅡコード表を作成せよ。

3-a. for 文による実行

3-a-1. 高水準コードの一部

```
01 #include <stdio.h>
02
03 int main(){
04   int c;
05
06   for(c = 0x20; c<=0x7e; c++){
07     if((c % 3) == 0) printf("¥n");
08     printf("%%d(%03d)-%%x(%x)-%%c(%c) | ",c,c,c);
09   }
10   printf("¥n");
11
12   return(0);
13 }
```

3-a-2. 実行結果

```
%d(032)-%x(20)-%c( ) |
%d(033)-%x(21)-%c(!) | %d(034)-%x(22)-%c(") | %d(035)-%x(23)-%c(#) |
%d(036)-%x(24)-%c($) | %d(037)-%x(25)-%c(%) | %d(038)-%x(26)-%c(&) |
%d(039)-%x(27)-%c(') | %d(040)-%x(28)-%c(() | %d(041)-%x(29)-%c()) |
%d(042)-%x(2a)-%c(*) | %d(043)-%x(2b)-%c(+) | %d(044)-%x(2c)-%c(,) |
%d(045)-%x(2d)-%c(-) | %d(046)-%x(2e)-%c(.) | %d(047)-%x(2f)-%c(/) |
%d(048)-%x(30)-%c(0) | %d(049)-%x(31)-%c(1) | %d(050)-%x(32)-%c(2) |
%d(051)-%x(33)-%c(3) | %d(052)-%x(34)-%c(4) | %d(053)-%x(35)-%c(5) |
%d(054)-%x(36)-%c(6) | %d(055)-%x(37)-%c(7) | %d(056)-%x(38)-%c(8) |
%d(057)-%x(39)-%c(9) | %d(058)-%x(3a)-%c(:) | %d(059)-%x(3b)-%c(;) |
%d(060)-%x(3c)-%c(<) | %d(061)-%x(3d)-%c(=) | %d(062)-%x(3e)-%c(>) |
%d(063)-%x(3f)-%c(?) | %d(064)-%x(40)-%c(@) | %d(065)-%x(41)-%c(A) |
%d(066)-%x(42)-%c(B) | %d(067)-%x(43)-%c(C) | %d(068)-%x(44)-%c(D) |
%d(069)-%x(45)-%c(E) | %d(070)-%x(46)-%c(F) | %d(071)-%x(47)-%c(G) |
%d(072)-%x(48)-%c(H) | %d(073)-%x(49)-%c(I) | %d(074)-%x(4a)-%c(J) |
%d(075)-%x(4b)-%c(K) | %d(076)-%x(4c)-%c(L) | %d(077)-%x(4d)-%c(M) |
%d(078)-%x(4e)-%c(N) | %d(079)-%x(4f)-%c(O) | %d(080)-%x(50)-%c(P) |
%d(081)-%x(51)-%c(Q) | %d(082)-%x(52)-%c(R) | %d(083)-%x(53)-%c(S) |
%d(084)-%x(54)-%c(T) | %d(085)-%x(55)-%c(U) | %d(086)-%x(56)-%c(V) |
%d(087)-%x(57)-%c(W) | %d(088)-%x(58)-%c(X) | %d(089)-%x(59)-%c(Y) |
%d(090)-%x(5a)-%c(Z) | %d(091)-%x(5b)-%c([) | %d(092)-%x(5c)-%c(¥) |
%d(093)-%x(5d)-%c(]) | %d(094)-%x(5e)-%c(^) | %d(095)-%x(5f)-%c(_) |
%d(096)-%x(60)-%c(`) | %d(097)-%x(61)-%c(a) | %d(098)-%x(62)-%c(b) |
%d(099)-%x(63)-%c(c) | %d(100)-%x(64)-%c(d) | %d(101)-%x(65)-%c(e) |
%d(102)-%x(66)-%c(f) | %d(103)-%x(67)-%c(g) | %d(104)-%x(68)-%c(h) |
%d(105)-%x(69)-%c(i) | %d(106)-%x(6a)-%c(j) | %d(107)-%x(6b)-%c(k) |
%d(108)-%x(6c)-%c(l) | %d(109)-%x(6d)-%c(m) | %d(110)-%x(6e)-%c(n) |
%d(111)-%x(6f)-%c(o) | %d(112)-%x(70)-%c(p) | %d(113)-%x(71)-%c(q) |
%d(114)-%x(72)-%c(r) | %d(115)-%x(73)-%c(s) | %d(116)-%x(74)-%c(t) |
%d(117)-%x(75)-%c(u) | %d(118)-%x(76)-%c(v) | %d(119)-%x(77)-%c(w) |
%d(120)-%x(78)-%c(x) | %d(121)-%x(79)-%c(y) | %d(122)-%x(7a)-%c(z) |
%d(123)-%x(7b)-%c({) | %d(124)-%x(7c)-%c(|) | %d(125)-%x(7d)-%c(}) |
%d(126)-%x(7e)-%c(~) |
```

3-a-3．考察

イ．初期値は 16 進数の 20、条件は 16 進数の 7e 以下で、ループ毎にインクリメントさせる。

ロ．変数を 3 で割った余りが 0 の時改行をさせる。

ハ．変数を 10 進数と 16 進数と文字で出力させる。

3-b. while 文による実行

3-b-1. 高水準コードの一部

```
 1      #include <stdio.h>
 2
 3      #define FALSE 0
 4      #define TRUE  !FALSE
 5
 6      int main(){
 7        int c;
 8
 9        c = 31;
10
11        while(TRUE){
12          c++;
13            if(c > 0x7e) break;
14          if((c % 3) == 0) printf("¥n");
15          printf("%%d(%03d)-%%x(%x)-%%c(%c) | ",c,c,c);
16        }
17        printf("¥n");
18
19        return(0);
20      }
```

3-b-2. 実行結果

```
%d(032)-%x(20)-%c( ) |
%d(033)-%x(21)-%c(!) | %d(034)-%x(22)-%c(") | %d(035)-%x(23)-%c(#) |
%d(036)-%x(24)-%c($) | %d(037)-%x(25)-%c(%) | %d(038)-%x(26)-%c(&) |
%d(039)-%x(27)-%c(') | %d(040)-%x(28)-%c(() | %d(041)-%x(29)-%c()) |
%d(042)-%x(2a)-%c(*) | %d(043)-%x(2b)-%c(+) | %d(044)-%x(2c)-%c(,) |
%d(045)-%x(2d)-%c(-) | %d(046)-%x(2e)-%c(.) | %d(047)-%x(2f)-%c(/) |
%d(048)-%x(30)-%c(0) | %d(049)-%x(31)-%c(1) | %d(050)-%x(32)-%c(2) |
%d(051)-%x(33)-%c(3) | %d(052)-%x(34)-%c(4) | %d(053)-%x(35)-%c(5) |
%d(054)-%x(36)-%c(6) | %d(055)-%x(37)-%c(7) | %d(056)-%x(38)-%c(8) |
%d(057)-%x(39)-%c(9) | %d(058)-%x(3a)-%c(:) | %d(059)-%x(3b)-%c(;) |
%d(060)-%x(3c)-%c(<) | %d(061)-%x(3d)-%c(=) | %d(062)-%x(3e)-%c(>) |
%d(063)-%x(3f)-%c(?) | %d(064)-%x(40)-%c(@) | %d(065)-%x(41)-%c(A) |
%d(066)-%x(42)-%c(B) | %d(067)-%x(43)-%c(C) | %d(068)-%x(44)-%c(D) |
%d(069)-%x(45)-%c(E) | %d(070)-%x(46)-%c(F) | %d(071)-%x(47)-%c(G) |
%d(072)-%x(48)-%c(H) | %d(073)-%x(49)-%c(I) | %d(074)-%x(4a)-%c(J) |
%d(075)-%x(4b)-%c(K) | %d(076)-%x(4c)-%c(L) | %d(077)-%x(4d)-%c(M) |
%d(078)-%x(4e)-%c(N) | %d(079)-%x(4f)-%c(O) | %d(080)-%x(50)-%c(P) |
%d(081)-%x(51)-%c(Q) | %d(082)-%x(52)-%c(R) | %d(083)-%x(53)-%c(S) |
%d(084)-%x(54)-%c(T) | %d(085)-%x(55)-%c(U) | %d(086)-%x(56)-%c(V) |
%d(087)-%x(57)-%c(W) | %d(088)-%x(58)-%c(X) | %d(089)-%x(59)-%c(Y) |
%d(090)-%x(5a)-%c(Z) | %d(091)-%x(5b)-%c([) | %d(092)-%x(5c)-%c(¥) |
%d(093)-%x(5d)-%c(]) | %d(094)-%x(5e)-%c(^) | %d(095)-%x(5f)-%c(_) |
%d(096)-%x(60)-%c(`) | %d(097)-%x(61)-%c(a) | %d(098)-%x(62)-%c(b) |
%d(099)-%x(63)-%c(c) | %d(100)-%x(64)-%c(d) | %d(101)-%x(65)-%c(e) |
%d(102)-%x(66)-%c(f) | %d(103)-%x(67)-%c(g) | %d(104)-%x(68)-%c(h) |
%d(105)-%x(69)-%c(i) | %d(106)-%x(6a)-%c(j) | %d(107)-%x(6b)-%c(k) |
%d(108)-%x(6c)-%c(l) | %d(109)-%x(6d)-%c(m) | %d(110)-%x(6e)-%c(n) |
%d(111)-%x(6f)-%c(o) | %d(112)-%x(70)-%c(p) | %d(113)-%x(71)-%c(q) |
%d(114)-%x(72)-%c(r) | %d(115)-%x(73)-%c(s) | %d(116)-%x(74)-%c(t) |
%d(117)-%x(75)-%c(u) | %d(118)-%x(76)-%c(v) | %d(119)-%x(77)-%c(w) |
%d(120)-%x(78)-%c(x) | %d(121)-%x(79)-%c(y) | %d(122)-%x(7a)-%c(z) |
%d(123)-%x(7b)-%c({) | %d(124)-%x(7c)-%c(|) | %d(125)-%x(7d)-%c()) |
%d(126)-%x(7e)-%c(~) |
```

3-b-3. 考察

イ．初期値が 16 進数の'1f'では無効になってしまったので 10 進数の'31'とした。

ロ．変数が 7e より大きい場合にループを抜けられるようにした。

4. 文字(文字列では無い)の演算について考察せよ。例) (‘a’-‘A’)?, (‘f’-‘a’)?

4-a-1. 文字演算(引き算)

```
 1      #include <stdio.h>
 2
 3      int main(){
 4        int answer1,answer2,answer3;
 5        answer1 = 'a'%'A';answer2 = 'f'%'a';answer3 = 'f'+('a'+'A');
 6      printf("'a'+'A' = %%d(%03d) %%x(%x) %%c(%c)¥n",answer1,answer1,answer1);
 7      printf("'f'+'a' = %%d(%03d) %%x(%x) %%c(%c)¥n",answer2,answer2,answer2);
 8      printf("'f'+('a'+'A') = %%d(%03d) %%x(%x) %%c(%c)¥n",answer3,answer3,answer3);
 9
10       return(0);
11      }
```

4-a-2. 実行結果

```
'a'-'A' = %d(032) %x(20) %c( )
'A'-'a' = %d(-32) %x(fffffe0) %c(?)
'f'-('a'-'A') = %d(070) %x(46) %c(F)
```

4-a-3. 考察

イ. 文字から文字を引くことは、ASCⅡコードの進数などでの引き算と同価値である。

ロ. ASCⅡコードで低い数値から高い数値を引いたところ 10 進数がマイナスの値を出したりしたが、これはコンピュータの内部表現に一致している。

ハ. ()を用いた演算でも正常に行われている。

足し算でも同様な考えが当てはまった。

4-a-4. 実行結果

```
'a'+'A' = %d(162) %x(a2) %c(?)
'f'+'a' = %d(199) %x(c7) %c(?)
'f'+('a'+'A') = %d(264) %x(108) %c)
```

4-a-5. 考察

イ. 引き算でも見られたが文字で表すことの出来ない値では’?’や’(‘、’)’のみとなったり出力されないことがある。

ロ. 掛け算、割り算、剰余算も行う事が出来、実行結果や考察に関しても引き算や足し算と同様の結果が得られた。

5. 反省・感想

文字演算のあたりで ASCⅡコードの数値を超えた値の文字表記が'?'やなんにも書かれていない時の区別が結局わかりませんでした。疑問が残る結果はあんまりいい気がしません。

while 文や if 文などを学んで、また出来る事が増えて嬉しいです。今までで一番やりがいがあったかなと思いますけど、まぁまだ 4 回目なんですけど。

今回は随分と自分を追い込んでしまったので次回からは余裕を持てるようにこなしていきたいです