プログラミング I

REPORT#3

提出日	:	2012 年 6 月 7 日 木曜日
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

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。

範囲例）数字：figures,英大文字：capital letter,
英小文字：small letter,非表示文字：Not Printable character,その他：misc.等

a:if 文を使った ASCII コードの範囲判断。

高水準コードの全体▼

```
1   /*
2     Program   :ypa.c
3     Student-ID:125722G
4     Author    :Tatsunori Chinen
5     Update    :2012/06/2(SAT)
6     comments  :separate by ascii code.
7   */
8   #include <stdio.h>
9   #define FALSE 0
10  #define TRUE !FALSE
11
12  int main(){
13
14      int hex,number;
15      char line[128];
16      number = 0;
17
18      while(TRUE){
19          number = number + 1;
20          if(number > 5)break;
21
22          printf( "Input a Hex value(0x00~0x7f) ====> ");
23          fgets(line,sizeof(line),stdin);
24          sscanf(line,"%x",&hex);
25
26          printf( "Trial Number=%02d : 0d%03d / 0x%03x",number,hex,hex );
27
28          if ( 0x20 <= hex && hex <= 0x7e )
29              printf( " :%%c(%c)\n",hex);
30          if ( hex <= 0x1f | 0x7f <= hex )
31              printf( " :is not Printable character.\n" );
32
33          printf( "\t\t\t    " );
34          if ( 0x00 <= hex && hex <= 0x1f || hex == 0x7f )
35              printf( " ====> control code" );
36          if ( 0x30 <= hex && hex <= 0x39 )
37              printf( " ====> number" );
38          if ( 0x41 <= hex && hex <= 0x5a )
39              printf( " ====> large char" );
40          if ( 0x61 <= hex && hex <= 0x7a )
41              printf( " ====> small char" );
42          if ( 0x20 <= hex && hex <= 0x2f || 0x3a <= hex && hex <= 0x40 || 0x5b <= hex && hex <= 0x60 || 0x7b <= hex
    && hex <= 0x7e )
43              printf( " ====> symbol" );
44
45          puts( "\n" );
46      }
47      return(0);
48
49  }
```

実行結果▼　※scanf()関数にはいずれも"10"を入力した。

```
Input a Hex value(0x00~0x7f) ====> 0x00
Trial Number=01 : 0d000 / 0x000 :is not Printable character.
                                ====> control code

Input a Hex value(0x00~0x7f) ====> 0x20
Trial Number=02 : 0d032 / 0x020 :%c( )
                                ====> symbol

Input a Hex value(0x00~0x7f) ====> 0x30
Trial Number=03 : 0d048 / 0x030 :%c(0)
                                ====> number

Input a Hex value(0x00~0x7f) ====> 0x41
Trial Number=04 : 0d065 / 0x041 :%c(A)
                                ====> large char

Input a Hex value(0x00~0x7f) ====> 61
Trial Number=05 : 0d097 / 0x061 :%c(a)
                                ====> small char
```

考察▼

・if 文を繰り返し使うことで,ASCII コードの先頭（0x00）から順に入力された値がどの範囲にあてはまるのかを判断するプログラムを作った。
・出力時の表示方法のため,どの範囲にあてはまるのかを出力する際には printf()関数を使っている。
・各範囲には，制御文字：control code,空白含む記号：symbol,数字：number,英大文字：large char,
英小文字：small char という範囲名を割り振った。
・このプログラムを含め、本レポートでは入力される値はプログラムが対応出来る範囲内であると想定している。

・この方法では,複数の if 文の条件に当てはまったとき,それらの if 文が真であったときの処理を
全て実行してしまう。
・そのため,各文字の範囲を判定するには範囲ごとに条件が重複しないようにしなければならない。
・コードの行数を減らすため,記号に対する判断では論理演算子"||"を使って条件を入力した。
・テキスト 411P 演算子優先ルールによれば,優先順位は論理積"&&"よりも低いため
論理積"&&"で各範囲が決められたあとそのいずれかにあてはまる,という条件が構成されている。

b:if else 文を使った ASCII コードの範囲判断。

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   #define FALSE 0
10  #define TRUE !FALSE
11
12  int main(){
13
14      int hex,number;
15      char line[128];
16      number = 0;
17
18      while(TRUE){
19          number = number + 1;
20          if(number > 5)break;
21
22          printf( "Input a Hex value(0x00~0x7f) ====> ");
23          fgets(line,sizeof(line),stdin);
24          sscanf(line,"%x",&hex);
25
26          printf( "Trial Number=%02d : 0d%03d / 0x%03x",number,hex,hex );
27
28          if ( 0x20 <= hex && hex <= 0x7e )
29              printf( " :%%c(%c)\n",hex);
30          else
31              printf( " :is not Printable character.\n" );
32
33          printf( "\t\t\t    " );
34          if ( 0x00 <= hex && hex <= 0x1f | hex == 0x7f )
35              printf( " ====> control code" );
36          else {
37              if ( 0x30 <= hex && hex <= 0x39 )
38                  printf( " ====> number" );
39              else {
40                  if ( 0x41 <= hex && hex <= 0x5a )
41                      printf( " ====> large char" );
42                  else {
43                      if ( 0x61 <= hex && hex <= 0x7a )
44                          printf( " ====> small char" );
45                      else
46                          printf( " ====> symbol" );
47                  }
48              }
49          }
50          puts( "\n" );
51      }
52      return(0);
53
54  }
```

実行結果▼　※"0x00"、"0x20"、"0x30"、"0x41"、"61"を各 fgets()関数に入力した。

```
Input a Hex value(0x00~0x7f) ====> 0x00
Trial Number=01 : 0d000 / 0x000 :is not Printable character.
                            ====> control code

Input a Hex value(0x00~0x7f) ====> 0x20
Trial Number=02 : 0d032 / 0x020 :%c( )
                            ====> symbol

Input a Hex value(0x00~0x7f) ====> 0x30
Trial Number=03 : 0d048 / 0x030 :%c(0)
                            ====> number

Input a Hex value(0x00~0x7f) ====> 0x41
Trial Number=04 : 0d065 / 0x041 :%c(A)
                            ====> large char

Input a Hex value(0x00~0x7f) ====> 61
Trial Number=05 : 0d097 / 0x061 :%c(a)
                            ====> small char
```

考察▼
・else if 文を繰り返し使うことで,入力された 16 進数の値が ASCII コードでどの範囲にあてはまるのか
を判断するプログラムを作った。
・if 文を使った試行とは違い,「ある条件にあてはまらなければ,このような範囲にある」という判断ができる。
・その性質を用いて,「制御文字,数字,大文字,小文字でなければ空白及び記号である」と判断させている。
・性質を発揮させるために,if else 文を else 分岐の内部に入れ子状に入力した。

c:else if 文を使った ASCII コードの範囲判断。

高水準コードの一部▼

```
8   #include <stdio.h>
9   #define FALSE 0
10  #define TRUE !FALSE
11
12  int main(){
13
14      int hex,number;
15      char line[128];
16      number = 0;
17
18      while(TRUE){
19          number = number + 1;
20          if(number > 5)break;
21
22          printf( "Input a Hex value(0x00~0x7f) ====> ");
23          fgets(line,sizeof(line),stdin);
24          sscanf(line,"%x",&hex);
25
26          printf( "Trial Number=%02d : 0d%03d / 0x%03x",number,hex,hex );
27
28          if ( 0x20 <= hex && hex <= 0x7e )
29              printf( " :%%c(%c)\n",hex);
30          else
31              printf( " :is not Printable character.\n" );
32
33          printf( "\t\t\t    " );
34          if      ( 0x00 <= hex && hex <= 0x1f || hex == 0x7f )
35              printf( " ====> control code" );
36          else if ( 0x30 <= hex && hex <= 0x39 )
37              printf( " ====> number" );
38          else if ( 0x41 <= hex && hex <= 0x5a )
39              printf( " ====> large char" );
40          else if ( 0x61 <= hex && hex <= 0x7a )
41              printf( " ====> small char" );
42          else
43              printf( " ====> symbol" );
44          puts( "\n" );
45      }
46      return(0);
47
48  }
```

実行結果▼　※scanf()関数にはいずれも"10"を入力した。

```
Input a Hex value(0x00~0x7f) ====> 0x00
Trial Number=01 : 0d000 / 0x000 :is not Printable character.
                              ====> control code

Input a Hex value(0x00~0x7f) ====> 0x20
Trial Number=02 : 0d032 / 0x020 :%c( )
                              ====> symbol

Input a Hex value(0x00~0x7f) ====> 0x30
Trial Number=03 : 0d048 / 0x030 :%c(0)
                              ====> number

Input a Hex value(0x00~0x7f) ====> 0x41
Trial Number=04 : 0d065 / 0x041 :%c(A)
                              ====> large char

Input a Hex value(0x00~0x7f) ====> 61
Trial Number=05 : 0d097 / 0x061 :%c(a)
                              ====> small char
```

考察▼

・else if 文を繰り返し使うことで,入力された 16 進数の値が ASCII コードでどの範囲にあてはまるのか
　を判断するプログラムを作った。
・if else 文と同じように,「ある条件にあてはまらなければ,このような範囲にある」という判断ができる。
・if else 文と違う点として、入力した際に入れ子構造をとらない。
・次に、if 文の動作を検証する。

d:if 文の動作の観察。

高水準コードの一部▼

```
8   #include <stdio.h>
9   int main(){
10
11      int number;
12      number = 0;
13      scanf( "%d",&number );
14      if( number < 20 )
15          printf( "Number1(%%x) = %x\n",number );
16          printf( "Number2(%%d) = %d\n",number );
17
18      if( number < 20 ){
19          printf( "Number3(%%x) = %x\n",number );
20          printf( "Number4(%%d) = %d\n",number );
21      }
22
23      return(0);
24
25  }
```

実行結果▼　※scanf()関数には、左の試行には"10"を、右の試行には"30"を入力した。

```
10
Number1(%x) = a
Number2(%d) = 10
Number3(%x) = a
Number4(%d) = 10
```

```
30
Number2(%d) = 30
```

考察▼

・if 文の,中括弧{}のある無しにおける動作の変化を確かめた。
・条件を満たす左の試行では,全ての printf 関数が出力された。
・条件を満たさない右の試行では,2 つ目の printf()関数だけが出力された。
・このことから,if 文は中括弧{}無しの場合,次の行までを分岐した際に実行する範囲とすることが分かった。
・中括弧{}でブロックが構成されている場合,ブロック内部の処理が全て範囲とされる。
・if else 文,else if 文においては,文に続くコードが 2 行以上になると、中括弧{}無しではコンパイルの際にエラーが表示された。
・よって、if else 文,else if 文も同様に、2 行以上のプログラムを処理させたい場合は中括弧{}を用いてブロックを構成する必要がある。

2.sample#2.c のプログラムの動作を考察せよ。

a:sample 文の While 文部分の動作について観察。

高水準コードの一部▼

```
8   #include <stdio.h>
9   #define FALSE 0
10  #define TRUE !FALSE
11  int main(){
12
13      int count;
14      count = 0;
15      while(TRUE){
16          count+=1;
17          if( count > 5 ) break;
18          printf( "While-Count = %2d\n",count);
19      }
20      return(0);
21
22  }
```

実行結果▼

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

考察▼

・sample#2 のコードから,while 文に関わる部分だけを抜きだし考察する。
・9 行目,10 行目の"#define"は,テキスト 137P によればある文字列を異なる文字列へ変更するコマンドに相当するという。
・実際のコードとしては、9 行目の"#define"で以降の全ての"FALSE"が"0"に変更され,
Wiki『Open Lecture』によれば"0"の否定値は"1"であるから,10 行目の"#define"について,
以降の全ての"TRUE"が"1"に変更されていると読み取れる。
・ここで,テキスト 78P より,while 文は条件が真のとき繰り返され、偽であれば実行されないという。
また、『Open Lecture』より,C 言語における偽は"0"で，真は"0 以外"である。
・文字列"FALSE"は変数や条件として扱われていないので,上記の考察が正しければ
"#define"を用いて"TRUE"を"1"に変更するだけで同じ結果を得られるはずだ。

b:"#define"の動作について観察。

高水準コードの一部▼

```
8   #include <stdio.h>
9   #define TRUE 1
10  int main(){
11
12      int count;
13      count = 0;
14      while(TRUE){
15          count+=1;
16          if( count > 5 ) break;
17          printf( "While-Count = %2d\n",count);
18      }
19      return(0);
20
21  }
```

実行結果▼

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

考察▼

・仮説通りの結果が得られた。
・"#define TRUE n"の n の値に 0 以外のどのような数字を入力しても while 文は実行された。
・9 行目を消去し while 文の条件に直接"0 以外"の値を入力しても while 文が実行された，
また,"define TRUE n"の n や、while 文の条件に直接"0"を入力すると while 文は実行されなかった。
これにより C 言語における真が"0 以外"であることが確認できた。
・次に,while 文と for 文の動作について考察する。

c-1:while 文と for 文の、デクリメント（--）による動作の観察。

高水準コードの一部▼

```
8   #include <stdio.h>
9   #define FALSE 0
10  #define TRUE !FALSE
11  int main(){
12
13      int count;
14      count = 0;
15      while(TRUE){
16          count--;
17          if( count < -9 ) break;
18          printf( "While-Count = %2d |",count);
19          if( (count % 3)== 0 ) printf( "\n" );
20      }
21
22      for(count=-1; count>=-9; count--){
23          printf("for  -Count = %2d |",count);
24          if( (count % 3)== 0 ) printf( "\n" );
25      }
26      return(0);
27
28  }
```

実行結果▼

```
While-Count = -1 |While-Count = -2 |While-Count = -3 |
While-Count = -4 |While-Count = -5 |While-Count = -6 |
While-Count = -7 |While-Count = -8 |While-Count = -9 |
for  -Count = -1 |for  -Count = -2 |for  -Count = -3 |
for  -Count = -4 |for  -Count = -5 |for  -Count = -6 |
for  -Count = -7 |for  -Count = -8 |for  -Count = -9 |
```

考察▼

・デクリメント（--）を用いて、最小値を条件とした While 文、for 文を作成した。
・マイナスの域であっても、問題なく実行されることが確認できた。
・for 文は、直後の括弧内に
「変数をどの値で初期化するか」「どの条件下で実行するか」「実行した後にどうするか」を入力するのに対し、
while 文は条件以外の 2 つを他のコードもしくは演算に依存する。

c-2:while 文と for 文の、初期値と加算量を大きく変えた動作の観察。

高水準コードの一部▼

```
8   #include <stdio.h>
9   #define FALSE 0
10  #define TRUE !FALSE
11  int main(){
12
13      int count;
14      count = 0;
15      while(TRUE){
16          count+=111;
17          if( count > 999 ) break;
18          printf( "While-Count = %2d |",count);
19          if( ((count / 111) % 3)== 0 ) printf( "\n" );
20      }
21
22      for(count=111; count<=999; count+=111){
23          printf("for  -Count = %2d |",count);
24          if( ((count / 111) % 3)== 0 ) printf( "\n" );
25      }
26      return(0);
27
28  }
```

実行結果▼

```
While-Count = 111 |While-Count = 222 |While-Count = 333 |
While-Count = 444 |While-Count = 555 |While-Count = 666 |
While-Count = 777 |While-Count = 888 |While-Count = 999 |
for  -Count = 111 |for  -Count = 222 |for  -Count = 333 |
for  -Count = 444 |for  -Count = 555 |for  -Count = 666 |
for  -Count = 777 |for  -Count = 888 |for  -Count = 999 |
```

考察▼

・初期値や、加算していく値を変えた場合であっても正常に動作した。
・次に、int 型整数の最大値付近での動作を観察する。

c-3:int 型変数の上限値付近での挙動の観察。

高水準コードの一部▼

```
8   #include <stdio.h>
9   #define FALSE 0
10  #define TRUE !FALSE
11  #define MAX 2147483647
12  #define START MAX - 5
13  int main(){
14
15      int count;
16      count = START;
17      while(TRUE){
18          count+=1;
19          if( count == MAX ) break;
20          printf( "While-Count = %2d |\n" ,count);
21      }
22
23      for(count=START+1; count < MAX; count+=1){
24          printf( "for  -Count = %2d |\n" ,count);
25      }
26      return(0);
27
28  }
```

実行結果▼

```
While-Count = 2147483643 |
While-Count = 2147483644 |
While-Count = 2147483645 |
While-Count = 2147483646 |
for  -Count = 2147483643 |
for  -Count = 2147483644 |
for  -Count = 2147483645 |
for  -Count = 2147483646 |
```

考察▼

・REPORT#2 で確認したように int 型変数の上限値は"2147483647"であり,変数"count"もその上限をもつ。
・そこで,上限値付近における while 文と for 文の動作を確かめた。
・上限値を出力しないような条件指定では,どちらも正常に実行された。
・次は,Count の最後の値として"2147483647"を出力させるためのコードについて考えたい。

c-4:int 型変数の上限値での挙動の観察。

高水準コードの一部▼

```
8   #include <stdio.h>
9   #define FALSE 0
10  #define TRUE !FALSE
11  #define MAX 2147483647
12  #define START MAX - 5
13  int main(){
14
15      int count;
16      count = START;
17      while(TRUE){
18          count+=1;
19          printf( "While-Count = %2d |\n" ,count);
20          if( count == MAX ) break;
21      }
22
23      for(count=START+1;START <= count && count <= MAX; count+=1){
24          printf( "for  -Count = %2d |\n" ,count);
25      }
26      return(0);
27
28  }
```

実行結果▼

```
While-Count = 2147483643 |
While-Count = 2147483644 |
While-Count = 2147483645 |
While-Count = 2147483646 |
While-Count = 2147483647 |
for  -Count = 2147483643 |
for  -Count = 2147483644 |
for  -Count = 2147483645 |
for  -Count = 2147483646 |
for  -Count = 2147483647 |
```

考察▼

・上限値"2147483647"を出力結果を含むために,上記のようなコードを用意した。
・while 文では,while 文から抜け出すかどうかを判断する"break"の宣言をブロックの最後に処理するだけで
上限値を出力することができた。
・このことから,while 文ではブロック内のコードが順次処理されていることが分かる。
・for 文の場合,括弧内の実行条件を c-2 項のように"count < MAX"としていると上限値は出力されない。
・for 文は while 文のようにループから抜けるかどうかを判断する"break"の宣言位置を自由に調節できないため,
実行条件に論理積（&&）を用いることで上限値を出力した。
・論理積（&&）を用いず,"count <= MAX"のみを for 文の条件として表記すると,
5 回目のループ時に REPORT#2 で述べたように上限値を超えた値が変数"count"に代入され,
負の値"-2147483648"となって格納される。
・負の値"-2147483648"は条件"count <= MAX"を満たすため,for 文は際限なく繰り返される結果となった。
・そのような事態を防ぐために,負の値の場合は実行しない条件を入力した。

c-5:for 文における"break"宣言。

高水準コードの一部▼

```
8   #include <stdio.h>
9   #define FALSE 0
10  #define TRUE !FALSE
11  #define MAX 2147483647
12  #define START MAX - 5
13  int main(){
14
15      int count;
16      for(count=START+1; count <= MAX; count+=1){
17          printf( "for  -Count = %2d\n" ,count);
18          if( count < 1 )break;
19      }
20      return(0);
21
22  }
```

実行結果▼

```
for  -Count = 2147483643
for  -Count = 2147483644
for  -Count = 2147483645
for  -Count = 2147483646
for  -Count = 2147483647
for  -Count = -2147483648
```

考察▼

・for 文のブロックの中で,c-3 項で述べたようなループを起こさないようにするために"break"宣言を行った。
・for 文であっても"break"宣言は有効であり,これにより for 文の場合でも特定の環境下で終了する動作が
可能である。

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。

a-1:while 文を用いた実行。

高水準コードの一部▼

```
8   #include <stdio.h>
9   #define FALSE 0       /* 以降のFALSEを0に置き換える */
10  #define TRUE !FALSE   /* 以降のTRUEを!FALSE（=!0）に置き換える */
11  int main(){
12
13      int number = 0x20 ,count = 0; /* number,countを宣言し,初期値を入力する */
14
15      while(TRUE){
16          printf( "  C / %%d/%%x ||" );                       /* C/%d/%xなどの表示 */
17          count++;                                            /* countをインクリメントする */
18          if ( count == 5 ) {                                 /* countを5つ出力したら改行してブレイク */
19              printf( "\n" ); break;
20          }
21      }
22      while(TRUE){
23          if( number == 0x7f )break;                          /* 値が0x7fであればブレイク */
24          printf( " \"%c\"=%3d=%x ||" ,number,number,number); /* 文字,10進数,16進数の順に表示する */
25          if( ((number - 0x1f) % 5) == 0 ) printf( "\n" );    /* 1行に5行出力したら改行する */
26          number+=0x01;                                       /* numberに"0x01"を足す */
27      }
28      printf( " ----List(0x20~0x7e) outputs finished.\n" );
29      return(0);
30
31  }
```

実行結果▼

```
  C / %d/%x ||  C / %d/%x ||  C / %d/%x ||  C / %d/%x ||  C / %d/%x ||
 " "= 32=20 || "!"= 33=21 || ""="= 34=22 || "#"= 35=23 || "$"= 36=24 ||
 "%"= 37=25 || "&"= 38=26 || "'"= 39=27 || "("= 40=28 || ")"= 41=29 ||
 "*"= 42=2a || "+"= 43=2b || ","= 44=2c || "-"= 45=2d || "."= 46=2e ||
 "/"= 47=2f || "0"= 48=30 || "1"= 49=31 || "2"= 50=32 || "3"= 51=33 ||
 "4"= 52=34 || "5"= 53=35 || "6"= 54=36 || "7"= 55=37 || "8"= 56=38 ||
 "9"= 57=39 || ":"= 58=3a || ";"= 59=3b || "<"= 60=3c || "="= 61=3d ||
 ">"= 62=3e || "?"= 63=3f || "@"= 64=40 || "A"= 65=41 || "B"= 66=42 ||
 "C"= 67=43 || "D"= 68=44 || "E"= 69=45 || "F"= 70=46 || "G"= 71=47 ||
 "H"= 72=48 || "I"= 73=49 || "J"= 74=4a || "K"= 75=4b || "L"= 76=4c ||
 "M"= 77=4d || "N"= 78=4e || "O"= 79=4f || "P"= 80=50 || "Q"= 81=51 ||
 "R"= 82=52 || "S"= 83=53 || "T"= 84=54 || "U"= 85=55 || "V"= 86=56 ||
 "W"= 87=57 || "X"= 88=58 || "Y"= 89=59 || "Z"= 90=5a || "["= 91=5b ||
 "\"= 92=5c || "]"= 93=5d || "^"= 94=5e || "_"= 95=5f || "`"= 96=60 ||
 "a"= 97=61 || "b"= 98=62 || "c"= 99=63 || "d"=100=64 || "e"=101=65 ||
 "f"=102=66 || "g"=103=67 || "h"=104=68 || "i"=105=69 || "j"=106=6a ||
 "k"=107=6b || "l"=108=6c || "m"=109=6d || "n"=110=6e || "o"=111=6f ||
 "p"=112=70 || "q"=113=71 || "r"=114=72 || "s"=115=73 || "t"=116=74 ||
 "u"=117=75 || "v"=118=76 || "w"=119=77 || "x"=120=78 || "y"=121=79 ||
 "z"=122=7a || "{"=123=7b || "|"=124=7c || "}"=125=7d || "~"=126=7e ||
 ----List(0x20~0x7e) outputs finished.
```

考察▼

・sample#3.c を解析し,ASCII コードと対応する文字を表示するプログラムを作成した。
・sample#3.c では for 文が用いられているが,while 文を使っての出力を行った。
・テキスト 407P によれば,"0x00"から"0x7f"までは文字や制御文字が割り振られている。
・制御文字は記号や文字ではないため,ここでは出力しないような範囲をとっている。

a-2:for 文による実行。

高水準コードの一部▼

```
8   #include <stdio.h>
9   int main(){
10
11      int number,count;
12
13      for(count=0;count<5;count++){       /* countを初期値0から,値が5より小さい限り実行しその後インクリメントする */
14          printf( "  C / %%d/%%x ||" );
15          if ( count == 4 ) printf( "\n" );
16      }
17      for(number=0x20;number<0x7f;number+=1){ /* numberを0x20から,値が0x7fより小さい限り実行しその後1を足す */
18          printf( " \"%c\"=%3d=%x ||" ,number,number,number);
19          if( ((number - 0x1f) % 5) == 0 ) printf( "\n" );
20      }
21      printf( " ----List(0x20~0x7e) outputs finished.\n" );
22      return(0);
23
24  }
```

実行結果▼

```
 C / %d/%x ||  C / %d/%x ||  C / %d/%x ||  C / %d/%x ||  C / %d/%x ||
" "= 32=20 || "!"= 33=21 || """= 34=22 || "#"= 35=23 || "$"= 36=24 ||
"%"= 37=25 || "&"= 38=26 || "'"= 39=27 || "("= 40=28 || ")"= 41=29 ||
"*"= 42=2a || "+"= 43=2b || ","= 44=2c || "-"= 45=2d || "."= 46=2e ||
"/"= 47=2f || "0"= 48=30 || "1"= 49=31 || "2"= 50=32 || "3"= 51=33 ||
"4"= 52=34 || "5"= 53=35 || "6"= 54=36 || "7"= 55=37 || "8"= 56=38 ||
"9"= 57=39 || ":"= 58=3a || ";"= 59=3b || "<"= 60=3c || "="= 61=3d ||
">"= 62=3e || "?"= 63=3f || "@"= 64=40 || "A"= 65=41 || "B"= 66=42 ||
"C"= 67=43 || "D"= 68=44 || "E"= 69=45 || "F"= 70=46 || "G"= 71=47 ||
"H"= 72=48 || "I"= 73=49 || "J"= 74=4a || "K"= 75=4b || "L"= 76=4c ||
"M"= 77=4d || "N"= 78=4e || "O"= 79=4f || "P"= 80=50 || "Q"= 81=51 ||
"R"= 82=52 || "S"= 83=53 || "T"= 84=54 || "U"= 85=55 || "V"= 86=56 ||
"W"= 87=57 || "X"= 88=58 || "Y"= 89=59 || "Z"= 90=5a || "["= 91=5b ||
"\"= 92=5c || "]"= 93=5d || "^"= 94=5e || "_"= 95=5f || "`"= 96=60 ||
"a"= 97=61 || "b"= 98=62 || "c"= 99=63 || "d"=100=64 || "e"=101=65 ||
"f"=102=66 || "g"=103=67 || "h"=104=68 || "i"=105=69 || "j"=106=6a ||
"k"=107=6b || "l"=108=6c || "m"=109=6d || "n"=110=6e || "o"=111=6f ||
"p"=112=70 || "q"=113=71 || "r"=114=72 || "s"=115=73 || "t"=116=74 ||
"u"=117=75 || "v"=118=76 || "w"=119=77 || "x"=120=78 || "y"=121=79 ||
"z"=122=7a || "{"=123=7b || "|"=124=7c || "}"=125=7d || "~"=126=7e ||
----List(0x20~0x7e) outputs finished.
```

考察▼

・a-1 項におけるプログラムを for 文で再現した。

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')?

a-1:演算についての考察。

高水準コードの一部▼

```
8   #include <stdio.h>
9
10  int main(){
11      int answer1,answer2,answer3,answer4,answer5,answer6;
12      answer1 = 'a'-'A';    answer2 = 'z'-'Z';
13      answer3 = 'A'-'a';    answer4 = 'Z'-'z';
14      answer5 = 'Z'-'a';    answer6 = 'Z'-'9';
15      printf( " 'a'-'A' = %%c(%c) %%x(%x) %%d(%d)\n",answer1,answer1,answer1);
16      printf( " 'z'-'Z' = %%c(%c) %%x(%x) %%d(%d)\n",answer2,answer2,answer2);
17      printf( " 'A'-'a' = %%c(%c) %%x(%x) %%d(%d)\n",answer3,answer3,answer3);
18      printf( " 'Z'-'z' = %%c(%c) %%x(%x) %%d(%d)\n",answer4,answer4,answer4);
19      printf( " 'A'-'a' = %%c(%c) %%x(%x) %%d(%d)\n",answer5,answer5,answer5);
20      printf( " 'Z'-'9' = %%c(%c) %%x(%X) %%d(%d)\n",answer6,answer6,answer6);
21      return(0);
22
23  }
```

実行結果▼

```
 'a'-'A' = %c( ) %x(20) %d(32)
 'z'-'Z' = %c( ) %x(20) %d(32)
 'A'-'a' = %c(?) %x(ffffffe0) %d(-32)
 'Z'-'z' = %c(?) %x(ffffffe0) %d(-32)
 'A'-'a' = %c(?) %x(fffffff9) %d(-7)
 'Z'-'9' = %c(!) %x(21) %d(33)
```

考察▼

・文字同士の演算を行った。
・ASCII コードの値が大きいアルファベットから小さいアルファベットを引くと,
その差が基数 16 において"20"（基数 10 における"32"）であることが分かった。
・値の小さいものから大きいものを引くと"-"の値が出るが,このときの基数 16 の値は
REPORT#2 で触れた内部表現のように円図を反時計回りに回っていったものである。
・この場合,コードに文字が割り振られていないためか文字として出力しようとすると"?"が出力された。
・アルファベット同士だけでなく、アルファベットと数値との計算も可能である。
・加法についても可能であり,以下の様な結果が得られる。

実行結果▼

```
 'a'+'A' = %c(?) %x(a2) %d(162)
 'z'+'Z' = %c(?) %x(d4) %d(212)
 'A'+'a' = %c(?) %x(a2) %d(162)
 'Z'+'z' = %c(?) %x(d4) %d(212)
 'A'+'a' = %c(?) %x(bb) %d(187)
 'Z'+'9' = %c(?) %x(93) %d(147)
```

考察▼

・"0x80"以上のコードには文字が登録されていないためか,演算はできるものの文字として出力することは
できなかった。
・"0x00"や"0x7f"など,制御文字の登録されているコードについては半角スペースや"?"なども出力されず
"("と")"が隣接した結果が得られた。
・除法や余りの演算についても行うことができた。

実行結果▼

```
 'a'*'A' = %c(?) %x(18a1) %d(6305)
 'z'*'Z' = %c(?) %x(2ae4) %d(10980)
 'A'*'a' = %c(?) %x(18a1) %d(6305)
 'Z'*'z' = %c(?) %x(2ae4) %d(10980)
 'A'*'a' = %c() %x(221a) %d(8730)
 'Z'*'9' = %c(
) %x(140A) %d(5130)
```

考察▼

・乗法についても計算が行えた。
・ある値で再び括弧が隣接する結果や改行のような結果が得られた。
・このことから,ある値以上のコードから再び文字や制御文字が割り振られていると予想できる。

a-2:基数 16 における"2000"付近の文字コードの考察。

高水準コードの一部▼

```
8   #include <stdio.h>
9
10  int main(){
11      int answer,number,cout;
12      number = 0x2000;
13      while(1){
14          if(number == 0x2050)break;
15          printf( " %%c(%c) %%x(%x) %%x(%d) ",number,number,number);
16          if((number - 0x1fff) % 3 == 0) printf( "\n" );
17          number+=0x01;
18      }
19      printf( "\n" );
20      return(0);
21
22  }
```

実行結果▼

```
 %c() %x(2000) %x(8192)  %c() %x(2001) %x(8193)  %c() %x(2002) %x(8194)
 %c() %x(2003) %x(8195)  %c() %x(2004) %x(8196)  %c() %x(2005) %x(8197)
 %c() %x(2006) %x(8198)  %c() %x(2007) %x(8199)  %c) %x(2008) %x(8200)
 %c(      ) %x(2009) %x(8201)  %c(
) %x(200a) %x(8202)  %c(
                       ) %x(200b) %x(8203)
 %c(
) %x(200d) %x(8205)  %c() %x(200e) %x(8206)
 %c() %x(200f) %x(8207)  %c() %x(2010) %x(8208)  %c() %x(2011) %x(8209)
 %c() %x(2012) %x(8210)  %c() %x(2013) %x(8211)  %c() %x(2014) %x(8212)
 %c() %x(2015) %x(8213)  %c() %x(2016) %x(8214)  %c() %x(2017) %x(8215)
 %c() %x(2018) %x(8216)  %c() %x(2019) %x(8217)  %c() %x(201a) %x(8218)
 %c(%x(201b) %x(8219)  %c() %x(201c) %x(8220)  %c() %x(201d) %x(8221)
 %c() %x(201e) %x(8222)  %c() %x(201f) %x(8223)  %c( ) %x(2020) %x(8224)
 %c(!) %x(2021) %x(8225)  %c(") %x(2022) %x(8226)  %c(#) %x(2023) %x(8227)
 %c($) %x(2024) %x(8228)  %c(%) %x(2025) %x(8229)  %c(&) %x(2026) %x(8230)
 %c(') %x(2027) %x(8231)  %c(() %x(2028) %x(8232)  %c()) %x(2029) %x(8233)
 %c(*) %x(202a) %x(8234)  %c(+) %x(202b) %x(8235)  %c(,) %x(202c) %x(8236)
 %c(-) %x(202d) %x(8237)  %c(.) %x(202e) %x(8238)  %c(/) %x(202f) %x(8239)
 %c(0) %x(2030) %x(8240)  %c(1) %x(2031) %x(8241)  %c(2) %x(2032) %x(8242)
 %c(3) %x(2033) %x(8243)  %c(4) %x(2034) %x(8244)  %c(5) %x(2035) %x(8245)
 %c(6) %x(2036) %x(8246)  %c(7) %x(2037) %x(8247)  %c(8) %x(2038) %x(8248)
 %c(9) %x(2039) %x(8249)  %c(:) %x(203a) %x(8250)  %c(;) %x(203b) %x(8251)
 %c(<) %x(203c) %x(8252)  %c(=) %x(203d) %x(8253)  %c(>) %x(203e) %x(8254)
 %c(?) %x(203f) %x(8255)  %c(@) %x(2040) %x(8256)  %c(A) %x(2041) %x(8257)
 %c(B) %x(2042) %x(8258)  %c(C) %x(2043) %x(8259)  %c(D) %x(2044) %x(8260)
 %c(E) %x(2045) %x(8261)  %c(F) %x(2046) %x(8262)  %c(G) %x(2047) %x(8263)
 %c(H) %x(2048) %x(8264)  %c(I) %x(2049) %x(8265)  %c(J) %x(204a) %x(8266)
 %c(K) %x(204b) %x(8267)  %c(L) %x(204c) %x(8268)  %c(M) %x(204d) %x(8269)
 %c(N) %x(204e) %x(8270)  %c(O) %x(204f) %x(8271)
```

考察▼

・
・得られた出力結果より,基数 16 における"2000"付近には再び文字や制御文字が割り振られていることが分かる。
・テキスト 407P によれば,"0x00"から"0x7f"までには,計"0x80"個(0d128 個)の文字や制御文字が割り振られている。
・半角スペースを表す出力は基数 16 における"2020"の地点でされているため,この値を基準とし
もう一度 ASCII コード表を構成することで文字コードが割り振られていることを確認したい。

a-3:基数 16 における"2020"以降の ASCII コード表。

高水準コードの一部▼

```
8   #include <stdio.h>
9
10  int main(){
11      int answer,number,cout;
12      number = 0x2000;
13      while(1){
14          if(number == 0x2050)break;
15          printf( " %%c(%c) %%x(%x) %%x(%d) ",number,number,number);
16          if((number - 0x1fff) % 3 == 0) printf( "\n" );
17          number+=0x01;
18      }
19      printf( "\n" );
20      return(0);
21
22  }
```

実行結果▼ ※実際に表示したウィンドウは 80×24 文字の設定であったため,自動改行がされていた。

```
 C /  %d/  %x ||  C /  %d/  %x ||  C /  %d/  %x ||  C /  %d/  %x ||  C /  %d/  %x ||
" "=8224=2020 || "!"=8225=2021 || """=8226=2022 || "#"=8227=2023 || "$"=8228=2024 ||
"%"=8229=2025 || "&"=8230=2026 || "'"=8231=2027 || "("=8232=2028 || ")"=8233=2029 ||
"*"=8234=202a || "+"=8235=202b || ","=8236=202c || "-"=8237=202d || "."=8238=202e ||
"/"=8239=202f || "0"=8240=2030 || "1"=8241=2031 || "2"=8242=2032 || "3"=8243=2033 ||
"4"=8244=2034 || "5"=8245=2035 || "6"=8246=2036 || "7"=8247=2037 || "8"=8248=2038 ||
"9"=8249=2039 || ":"=8250=203a || ";"=8251=203b || "<"=8252=203c || "="=8253=203d ||
">"=8254=203e || "?"=8255=203f || "@"=8256=2040 || "A"=8257=2041 || "B"=8258=2042 ||
"C"=8259=2043 || "D"=8260=2044 || "E"=8261=2045 || "F"=8262=2046 || "G"=8263=2047 ||
"H"=8264=2048 || "I"=8265=2049 || "J"=8266=204a || "K"=8267=204b || "L"=8268=204c ||
"M"=8269=204d || "N"=8270=204e || "O"=8271=204f || "P"=8272=2050 || "Q"=8273=2051 ||
"R"=8274=2052 || "S"=8275=2053 || "T"=8276=2054 || "U"=8277=2055 || "V"=8278=2056 ||
"W"=8279=2057 || "X"=8280=2058 || "Y"=8281=2059 || "Z"=8282=205a || "["=8283=205b ||
"\"=8284=205c || "]"=8285=205d || "^"=8286=205e || "_"=8287=205f || "`"=8288=2060 ||
"a"=8289=2061 || "b"=8290=2062 || "c"=8291=2063 || "d"=8292=2064 || "e"=8293=2065 ||
"f"=8294=2066 || "g"=8295=2067 || "h"=8296=2068 || "i"=8297=2069 || "j"=8298=206a ||
"k"=8299=206b || "l"=8300=206c || "m"=8301=206d || "n"=8302=206e || "o"=8303=206f ||
"p"=8304=2070 || "q"=8305=2071 || "r"=8306=2072 || "s"=8307=2073 || "t"=8308=2074 ||
"u"=8309=2075 || "v"=8310=2076 || "w"=8311=2077 || "x"=8312=2078 || "y"=8313=2079 ||
"z"=8314=207a || "{"=8315=207b || "|"=8316=207c || "}"=8317=207d || "~"=8318=207e ||
----List(0x2020~0x207e) outputs finished.
```

考察▼

・得られた出力結果より,基数 16 における"2020"以降にも文字が割り振られていることが分かった。
・a-2 項の結果を踏まえると,基数 16 における"2000"から"207f"までに"00"から"7f"と同様の文字,制御文字が割り振られている。
・ちなみに,"0x2020"に"0x2000"を加えた"0x4020"や,更に"0x2000"を加えた"0x6020"から表の出力を開始しても,同様の文字が割り振られていることを確認した。

XXX.あとがき。（反省・感想・参考）

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。

・乗法の試行で再び文字が出力されたときは何が起こったかと思いました。
全角文字が登録されているのかと思いましたが,"0x00"から"0x7f"までの登録が繰り返されているようでした。
どういう目的があって行われているんでしょうか。

あんまり試行ができなかったように思えます。
次のレポートは色んなことに挑戦してみたいです。時間に余裕を持って。

今回のレポートでできることが更に広がりました。
これを生かして次のレポートに取り組みたいです。

最後までご覧いただきありがとうございました。