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。

i .サンプルプログラム sample#1.c

●ソースコードの一部　※後半のコメントアウトは省略

```
1   #include <stdio.h>
2
3   #define FALSE 0
4   #define TRUE  !FALSE
5
6   int main(){
7     int  value,c, count;
8     char line[128];
9
10    count = 0;
11
12    while(TRUE){
13      count++;
14      if(count > 5) break;
15
16      printf("Enter a HexValue ===> ");
17      fgets(line, sizeof(line), stdin);
18      sscanf(line, "%x", &c);
19
20      printf("Colum=%02d:%%d(%3d)-%%x(%2x)",count,c,c);
21      if(0x20 <= c && c <= 0x7e)
22        printf("-%%c(%c)\n",c);
23      else
24        printf("-Not Printable character\n");
25
26      if      (0x20 <= c && c <= 0x2f){
27        puts("====> Scope_A\n");
28      }else if(0x30 <= c && c <= 0x39){
29        puts("====> Scope_B\n");
30      }else if(0x3a <= c && c <= 0x40){
31        puts("====> Scope_C\n");
32      }else if(0x41 <= c && c <= 0x5a){
33        puts("====> Scope_D\n");
34      }else{
35        puts("====> Scope_E\n");
36      }
37    }
38
39    return(0);
40  }
```

●出力結果　　※"56"、"07"、"ff"、"7f"、"7e"をそれぞれ入力した

```
Enter a HexValue ===> 56
Colum=01:%d( 86)-%x(56)-%c(V)
====> Scope_D

Enter a HexValue ===> 07
Colum=02:%d(  7)-%x( 7)-Not Printable character
====> Scope_E

Enter a HexValue ===> ff
Colum=03:%d(255)-%x(ff)-Not Printable character
====> Scope_E

Enter a HexValue ===> 7f
Colum=04:%d(127)-%x(7f)-Not Printable character
====> Scope_E

Enter a HexValue ===> 7e
Colum=05:%d(126)-%x(7e)-%c(~)
====> Scope_E
```

●プログラムの解析

・12 行目-37 行目の while 文により 5 回プログラムが繰り返し実行されている.

・21 行目の if、23 行目の else により入力された 16 進法の数字が ASCII コード表の範囲内にあるか場合分けをしている.

・26 行目-36 行目の if-else 文により、さらに入力された数字が ASCII コード表のどの範囲にあるのかを判断している.

ii .ASCII コードの各範囲を判断するプログラムを作成する。

●ソースコード全体

```
#include <stdio.h>

#define FALSE 0
#define TRUE  !FALSE

int main(){
  int  value,c, count;
  char line[128];

  count = 0;

  while(TRUE){
    count++;
    if(count > 5) break;

    printf("Enter a HexValue(0x00~0x7f) ===> ");
    fgets(line, sizeof(line), stdin);
    sscanf(line, "%x", &c);

    printf("Colum=%02d:%%d(%3d)-%%x(%2x)",count,c,c);
    if(0x20 <= c && c <= 0x7e){
      printf("-%%c(%c)\n",c);

      if      (0x30 <= c && c <= 0x39){
        puts("====> Figures \n");
      }else if(0x41 <= c && c <= 0x5a){
        puts("====> Capital Letter\n");
      }else if(0x61 <= c && c <= 0x7a){
        puts("====> Small Letter\n");
      }else{
        puts("====> Symbol\n");
      }
    }
    else if(0x00 <= c && c <= 0x1f || c == 0x7f){
      printf("-Not Printable Character\n====> control code\n\n");
    }
    else{
      puts("-Error! Pleas Enter (0x00~0x7f)\n");
    }
  }

  return(0);
}
```

●実行結果　※"0"、"26"、"38"、"75"、"ff"をそれぞれ入力した

```
Enter a HexValue(0x00~0x7f) ===> 0
Colum=01:%d(  0)-%x( 0)-Not Printable Character
====> control code

Enter a HexValue(0x00~0x7f) ===> 26
Colum=02:%d( 38)-%x(26)-%c(&)
====> Symbol

Enter a HexValue(0x00~0x7f) ===> 38
Colum=03:%d( 56)-%x(38)-%c(8)
====> Figures

Enter a HexValue(0x00~0x7f) ===> 75
Colum=04:%d(117)-%x(75)-%c(u)
====> Small Letter

Enter a HexValue(0x00~0x7f) ===> ff
Colum=05:%d(255)-%x(ff)-Error! Pleas Enter (0x00~0x7f)
```

●考察

・サンプルプログラム sample#1.c を書き換えて、ASCII コードの各範囲を判断するプログラムを作成した.

・各範囲には、数字:Figures、英大文字:Capital Letter、 英小文字:Small Letter、空白含む記号:Symbol、制御文字:Control Code、という範囲名を割り振った.

・ASCII コード内の範囲を超えた数字を入力を入力した場合、範囲を示すのではなくエラー（-Error! Pleas Enter (0x00~0x7f)）を表すようした.

2. sample#2.c のプログラムの動作を考察せよ。

i .サンプルプログラム sample#2.c

●ソースコードの一部　※後半のコメントアウトは省略

```
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    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  }
```

●出力結果

```
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
```

●考察

・このプログラムは制御文である while 文と for 文から成っており、出力結果から意味は等価である.

・テキスト P.137 によると、ソースコードの３行目、４行目の "#define”は文字列を別の別の文字列へ置換する機能があるという.

・また、テキスト P.78 によると while 文は条件が真である間だけ実行を繰り返すという.

・この場合の条件とは "TRUE”であり、 "#define”により"TRUE”は "FALSE”以外（０以外）と置換されている.

・つまり、 "#define”によって"TRUE”を０以外の数字又は文字へ置換しても、同じ結果が得られるはずである.

ⅱ. "#define"の動作について観察。

●ソースコード全体

```
1   #include <stdio.h>
2
3   #define TRUE 1
4
5   int main(){
6     int count;
7
8     count = 0;
9     while(TRUE){
10      count++;
11      if( count > 5 ) break;
12      printf( "While-Count = %2d\n",count);
13    }
14
15    return(0);
16  }
```

●出力結果

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

●考察

・予想通りの結果が得られた.

・3 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.

・次は、"#befine”を用いない while 文の動作を観察する.

iii. "#befine”を用いない wihle 文

●ソースコード全体

```
1   #include <stdio.h>
2
3   int main(){
4     int count;
5
6     count = 0;
7     while(count < 5){
8       count++;
9       printf( "While-Count = %2d\n",count);
10    }
11
12    return(0);
13  }
```

●出力結果

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

●考察

・break 文を用いず、while 文の条件へサンプルプログラムと同じ結果になるように直接繰り返しの条件を入力した.

・うまくサンプルプログラムの while 文と同じ結果を得ることができた.

iv. while 文と for 文の繰り返し回数

●ソースコード全体

```
#include <stdio.h>

#define FALSE 0
#define TRUE  !FALSE

int main(){
  int  count;

  count = 0;
  while(TRUE){
    count++;
    if(count > 5) break;
    printf("While-Count=%2d\n",count);
  }
  printf("count=\n",count);

  for(count=1; count<=5; count++){
    printf("for  -Count=%2d\n",count);
  }
  printf("count=",count);

  return(0);
}
```

●出力結果

```
While-Count= 1
While-Count= 2
While-Count= 3
While-Count= 4
While-Count= 5
Wcount=6
for  -Count= 1
for  -Count= 2
for  -Count= 3
for  -Count= 4
for  -Count= 5
Fcount=6
```

●考察

・while 文、for 文が終了した直後にそれぞれ変数 "count”に格納されている数を表示した.
・結果、どちらも繰り返した回数より 1 多い数が表示された.
・while 文では、書かれているプログラムが上から順に実行していくので制御文内で 6 回目の "count++”を実行して break 文によりループから抜けている.
・for 文では、条件”count<=5”の間だけ実行されるので、条件を満たさない時、つまり "count++”により "count”が 6 になった時点でループを終了している.
・よって、それぞれの制御文によるループが終了した時点では、変数 "count”の中には 6 が格納されていることになる.
・これを踏まえて、それぞれの制御文の変数 "count”の初期値と、条件文を統一して、同じ結果を表示させることを考える.

ｖ. while 文と for 文の関係

●ソースコード全体

```
1   #include <stdio.h>
2
3   int main(){
4     int  count;
5
6     count = 1;
7     while(count<=5){
8       printf("While-Count=%2d\n",count);
9       count++;
10    }
11
12    for(count=1; count<=5; count++){
13      printf("for  -Count=%2d\n",count);
14    }
15
16    return(0);
17  }
```

●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
```

●考察

・狙い通りの結果を得ることができた.

・while 文では上から順にコードを実行していくので、 "count++"を本文である printf 関数の後（9 行目）に置き、変数 "count"の初期値を 1 とした.

・これにより、while 文と for 文は下記のように同じ値により書き換え可能であることが分かった.

○while 文と for 文の書き換え

```
初期値;
  while(条件){
    本文;
    反復式;
  }

  for(初期値; 条件; 反復式){
    本文;
  }
```

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

i .サンプルプログラム sample#3.c

●ソースコードの一部　※後半のコメントアウトは省略

```
1   #include <stdio.h>
2
3   int main(){
4     int c;
5
6     for(c = 0x20; c<=0x40; c++){
7       if((c % 4) == 0) printf("\n");
8       printf("%%x(%x)-%%c(%c) | ",c,c);
9     }
10    printf("\n");
11
12    return(0);
13  }
```

●出力結果

```
%x(20)-%c( ) | %x(21)-%c(!) | %x(22)-%c(") | %x(23)-%c(#) |
%x(24)-%c($) | %x(25)-%c(%) | %x(26)-%c(&) | %x(27)-%c(') |
%x(28)-%c(() | %x(29)-%c()) | %x(2a)-%c(*) | %x(2b)-%c(+) |
%x(2c)-%c(,) | %x(2d)-%c(-) | %x(2e)-%c(.) | %x(2f)-%c(/) |
%x(30)-%c(0) | %x(31)-%c(1) | %x(32)-%c(2) | %x(33)-%c(3) |
%x(34)-%c(4) | %x(35)-%c(5) | %x(36)-%c(6) | %x(37)-%c(7) |
%x(38)-%c(8) | %x(39)-%c(9) | %x(3a)-%c(:) | %x(3b)-%c(;) |
%x(3c)-%c(<) | %x(3d)-%c(=) | %x(3e)-%c(>) | %x(3f)-%c(?) |
%x(40)-%c(@) |
```

●考察

・for 文により、"0x20"から "0x40"までの ASCII コードが繰り返し表示されている.

・7 行目の if 文により、変数 "c"が 4 で割り切れる時（4 の倍数の時）に改行するようにしている.

ii .表示可能な文字による ASCII コード表を作成。

●ソースコード全体

```
1   #include <stdio.h>
2
3   int main(){
4     int c;
5
6     printf("ASCII Code List(0x20-0x7e)");
7     for(c = 0x20; c<=0x7e; c++){
8       if(((c-0x02) % 5) == 0) printf("\n");
9       printf("%%d:%3d-%%x:%2x-[%c] | ",c,c,c);
10    }
11    printf("\n");
12
13    return(0);
14  }
```

●出力結果

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

●考察

・ASCII コード表によれば、表示可能な文字（図形文字）は "0x20"から "0x7e"までの範囲に割り振られているという.

・サンプルプログラム sample#3.c を基本に、表示する範囲の拡大、表の列を増やす等を行った.

iii. ii 項の表の縦表示

●ソースコード全体

```
#include <stdio.h>

int main(){
  int c,n;

  printf("ASCII Code List(0x20-0x7e)\n");
  for(n = 0x20; n<=0x32; n++){
    for(c = n; c<=0x7e; c=c+19){
      printf("%%d:%3d-%%x:%2x-[%c] | ",c,c,c);
    }
    printf("\n");
  }

  return(0);
}
```

●出力結果

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

●考察

・ii 項で制作したプログラムを改良し、左上から右へではなく下へ連続するようにした.

・1 つの行の中で、右へ+19 づつ for 文でもって繰り返し表示し（8 行目）、それを for 文で 19 行表示するように繰り返した（7 行目）.

iv.iii 項のプログラムを while 文で表示

●ソースコード全体

```
#include <stdio.h>

int main(){
  int c,n;

  printf("ASCII Code List(0x20-0x7e)\n");
  n=0x20;
  while(n<=0x32){
    c=n;
    while(c<=0x7e){
      printf("%%d:%3d-%%x:%2x-[%c] | ",c,c,c);
      c=c+19;
    }
    printf("\n");
    n++;
  }

  return(0);
}
```

●出力結果

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

●考察

・iii 項で制作したプログラムを while 文で書き換えた.

・狙い通りに出力することができた.

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')?

ⅰ.文字同士の減法の演算

●ソースコード全体

```
#include <stdio.h>

int main(){
  int A1,A2,A3,A4,A5,A6;
  A1='a'-'A'; A2='z'-'Z';
  A3='A'-'a'; A4='A'-'z';
  A5='A'-'$'; A6='X'-'5';

  printf("'a'-'A' = %%c[%c] %%x[%08x] %%d[%d]\n",A1,A1,A1);
  printf("'z'-'Z' = %%c[%c] %%x[%08x] %%d[%d]\n",A2,A2,A2);
  printf("'A'-'a' = %%c[%c] %%x[%08x] %%d[%d]\n",A3,A3,A3);
  printf("'A'-'z' = %%c[%c] %%x[%08x] %%d[%d]\n",A4,A4,A4);
  printf("'A'-'$' = %%c[%c] %%x[%08x] %%d[%d]\n",A5,A5,A5);
  printf("'X'-'5' = %%c[%c] %%x[%08x] %%d[%d]\n",A6,A6,A6);

  return(0);
}
```

●出力結果

```
'a'-'A' = %c[ ] %x[00000020] %d[32]
'z'-'Z' = %c[ ] %x[00000020] %d[32]
'A'-'a' = %c[?] %x[fffffffe0] %d[-32]
'A'-'z' = %c[?] %x[fffffffc7] %d[-57]
'A'-'$' = %c[] %x[0000001d] %d[29]
'X'-'5' = %c[#] %x[00000023] %d[35]
```

●考察

・文字同士の演算を行うプログラムを用意した.

・変数に格納する物を、アポストロフィ（'）で囲むことで文字（大文字、小文字、数字、記号）として格納させてる.

・出力結果から、文字同士の減法については可能であることがわかる.

・また、変数に格納された文字を ASCII コード表における数字（例：'A'=65,0x41）として判断し、演算していることがわかる.

・ASCII コード表に割り振られていないコードが計算結果にあるとき、文字"%c"の欄には"?"が出力されている.

・制御文字が割り振られているコードについては、"?"すらも出力されなかった.

ii.文字同士の加法、除法、余りの演算

●ソースコード全体

```
#include <stdio.h>

int main(){
  int A1,A2,A3,A4,A5,A6;
  A1='a'+'A'; A2='z'+'Z';
  A3='A'+'a'; A4='A'+'z';
  A5='A'+'$'; A6='X'+'5';

  printf("'a'+'A' = %%c[%c] %%x[%08x] %%d[%d]\n",A1,A1,A1);
  printf("'z'+'Z' = %%c[%c] %%x[%08x] %%d[%d]\n",A2,A2,A2);
  printf("'A'+'a' = %%c[%c] %%x[%08x] %%d[%d]\n",A3,A3,A3);
  printf("'A'+'z' = %%c[%c] %%x[%08x] %%d[%d]\n",A4,A4,A4);
  printf("'A'+'$' = %%c[%c] %%x[%08x] %%d[%d]\n",A5,A5,A5);
  printf("'X'+'5' = %%c[%c] %%x[%08x] %%d[%d]\n",A6,A6,A6);

  return(0);
}
```

●出力結果

```
'a'+'A' = %c[?] %x[000000a2] %d[162]
'z'+'Z' = %c[?] %x[000000d4] %d[212]
'A'+'a' = %c[?] %x[000000a2] %d[162]
'A'+'z' = %c[?] %x[000000bb] %d[187]
'A'+'$' = %c[e] %x[00000065] %d[101]
'X'+'5' = %c[?] %x[0000008d] %d[141]
```

●考察

・文字同士の加法についても同様に計算できることがわかった.

・文字の除法、余りの演算についても問題なく行うことができた.

iii.文字同士の乗法の演算

●ソースコード全体

```
#include <stdio.h>

int main(){
  int A1,A2,A3,A4,A5,A6;
  A1='a'*'A'; A2='z'*'Z';
  A3='A'*'a'; A4='A'*'z';
  A5='A'*'$'; A6='X'*'5';

  printf("'a'*'A' = %%c[%c] %%x[%08x] %%d[%d]\n",A1,A1,A1);
  printf("'z'*'Z' = %%c[%c] %%x[%08x] %%d[%d]\n",A2,A2,A2);
  printf("'A'*'a' = %%c[%c] %%x[%08x] %%d[%d]\n",A3,A3,A3);
  printf("'A'*'z' = %%c[%c] %%x[%08x] %%d[%d]\n",A4,A4,A4);
  printf("'A'*'$' = %%c[%c] %%x[%08x] %%d[%d]\n",A5,A5,A5);
  printf("'X'*'5' = %%c[%c] %%x[%08x] %%d[%d]\n",A6,A6,A6);

  return(0);
}
```

●出力結果

```
'a'*'A' = %c[?] %x[000018a1] %d[6305]
'z'*'Z' = %c[?] %x[00002ae4] %d[10980]
'A'*'a' = %c[?] %x[000018a1] %d[6305]
'A'*'z' = %c[?] %x[00001efa] %d[7930]
'A'*'$' = %c[$] %x[00000924] %d[2340]
'X'*'5' = %c[8] %x[00001238] %d[4664]
```

●考察

・文字同士の乗法も問題なく行うことができることがわかった.

・出力結果を観察すると、ASCII コード表には割り当てられてないコードでも文字が出力されている箇所がある.

iv.”0x2000”-”0x2200”における ASCII コード

●方針

・3- ii 項の ASCII コード表を表示するプログラムを用いて、”0x2000”-”0x2200”の範囲におけるコード表を出力する.

・”0x2000”-”0x2100”、”0x2100”-”0x2200”の範囲に分けて、それぞれの出力結果を観察する.

●ソースコード全体(1)

```
#include <stdio.h>

int main(){
  int c;

  printf(”Code List(0x2000-0x2100)\n”);
  for(c = 0x2000; c<=0x2100; c++){
    if(((c-0x02) % 4) == 0) printf(”\n”);
    printf(”%%d:%3d-%%x:%2x-[%c] | ”,c,c,c);
  }
  printf(”\n”);

  return(0);
}
```

●ソースコード全体(2)

```
#include <stdio.h>

int main(){
  int c;

  printf(”Code List(0x2100-0x2200)\n”);
  for(c = 0x2100; c<=0x2200; c++){
    if(((c-0x02) % 4) == 0) printf(”\n”);
    printf(”%%d:%3d-%%x:%2x-[%c] | ”,c,c,c);
  }
  printf(”\n”);

  return(0);
}
```

●出力結果(1)

```
Code List(0x2000-0x2100)
%d:8192-%x:2000-[] | %d:8193-%x:2001-[] |
%d:8194-%x:2002-[] | %d:8195-%x:2003-[] | %d:8196-%x:2004-[] | %d:8197-%x:2005-[] |
%d:8198-%x:2006-[] | %d:8199-%x:2007-[] | %d:8200-%x:2008-] | %d:8201-%x:2009-[  ] |
%d:8202-%x:200a-[
] | %d:8203-%x:200b-[
                    ] | %d:8204-%x:200c-[
] |                                       ] | %d:8205-%x:200d-[
%d:8206-%x:200e-[] | %d:8207-%x:200f-[] | %d:8208-%x:2010-[] | %d:8209-%x:2011-[] |
%d:8210-%x:2012-[] | %d:8211-%x:2013-[] | %d:8212-%x:2014-[] | %d:8213-%x:2015-[] |
%d:8214-%x:2016-[] | %d:8215-%x:2017-[] | %d:8216-%x:2018-[] | %d:8217-%x:2019-[] |
%d:8218-%x:201a-[] | %d:8219-%x:201b-[ %d:8220-%x:201c-[] | %d:8221-%x:201d-[] |
%d:8222-%x:201e-[] | %d:8223-%x:201f-[] | %d:8224-%x:2020-[ ] | %d:8225-%x:2021-[!] |
%d:8226-%x:2022-["] | %d:8227-%x:2023-[#] | %d:8228-%x:2024-[$] | %d:8229-%x:2025-[%] |
%d:8230-%x:2026-[&] | %d:8231-%x:2027-['] | %d:8232-%x:2028-[(] | %d:8233-%x:2029-[)] |
%d:8234-%x:202a-[*] | %d:8235-%x:202b-[+] | %d:8236-%x:202c-[,] | %d:8237-%x:202d-[-] |
%d:8238-%x:202e-[.] | %d:8239-%x:202f-[/] | %d:8240-%x:2030-[0] | %d:8241-%x:2031-[1] |
%d:8242-%x:2032-[2] | %d:8243-%x:2033-[3] | %d:8244-%x:2034-[4] | %d:8245-%x:2035-[5] |
%d:8246-%x:2036-[6] | %d:8247-%x:2037-[7] | %d:8248-%x:2038-[8] | %d:8249-%x:2039-[9] |
%d:8250-%x:203a-[:] | %d:8251-%x:203b-[;] | %d:8252-%x:203c-[<] | %d:8253-%x:203d-[=] |
%d:8254-%x:203e-[>] | %d:8255-%x:203f-[?] | %d:8256-%x:2040-[@] | %d:8257-%x:2041-[A] |
%d:8258-%x:2042-[B] | %d:8259-%x:2043-[C] | %d:8260-%x:2044-[D] | %d:8261-%x:2045-[E] |
%d:8262-%x:2046-[F] | %d:8263-%x:2047-[G] | %d:8264-%x:2048-[H] | %d:8265-%x:2049-[I] |
%d:8266-%x:204a-[J] | %d:8267-%x:204b-[K] | %d:8268-%x:204c-[L] | %d:8269-%x:204d-[M] |
%d:8270-%x:204e-[N] | %d:8271-%x:204f-[O] | %d:8272-%x:2050-[P] | %d:8273-%x:2051-[Q] |
%d:8274-%x:2052-[R] | %d:8275-%x:2053-[S] | %d:8276-%x:2054-[T] | %d:8277-%x:2055-[U] |
%d:8278-%x:2056-[V] | %d:8279-%x:2057-[W] | %d:8280-%x:2058-[X] | %d:8281-%x:2059-[Y] |
%d:8282-%x:205a-[Z] | %d:8283-%x:205b-[[] | %d:8284-%x:205c-[\] | %d:8285-%x:205d-[]] |
%d:8286-%x:205e-[^] | %d:8287-%x:205f-[_] | %d:8288-%x:2060-[`] | %d:8289-%x:2061-[a] |
%d:8290-%x:2062-[b] | %d:8291-%x:2063-[c] | %d:8292-%x:2064-[d] | %d:8293-%x:2065-[e] |
%d:8294-%x:2066-[f] | %d:8295-%x:2067-[g] | %d:8296-%x:2068-[h] | %d:8297-%x:2069-[i] |
%d:8298-%x:206a-[j] | %d:8299-%x:206b-[k] | %d:8300-%x:206c-[l] | %d:8301-%x:206d-[m] |
%d:8302-%x:206e-[n] | %d:8303-%x:206f-[o] | %d:8304-%x:2070-[p] | %d:8305-%x:2071-[q] |
%d:8306-%x:2072-[r] | %d:8307-%x:2073-[s] | %d:8308-%x:2074-[t] | %d:8309-%x:2075-[u] |
%d:8310-%x:2076-[v] | %d:8311-%x:2077-[w] | %d:8312-%x:2078-[x] | %d:8313-%x:2079-[y] |
%d:8314-%x:207a-[z] | %d:8315-%x:207b-[{] | %d:8316-%x:207c-[|] | %d:8317-%x:207d-[}] |
%d:8318-%x:207e-[~] | %d:8319-%x:207f-[] | %d:8320-%x:2080-[?] | %d:8321-%x:2081-[?] |
%d:8322-%x:2082-[?] | %d:8323-%x:2083-[?] | %d:8324-%x:2084-[?] | %d:8325-%x:2085-[?] |
%d:8326-%x:2086-[?] | %d:8327-%x:2087-[?] | %d:8328-%x:2088-[?] | %d:8329-%x:2089-[?] |
%d:8330-%x:208a-[?] | %d:8331-%x:208b-[?] | %d:8332-%x:208c-[?] | %d:8333-%x:208d-[?] |
%d:8334-%x:208e-[?] | %d:8335-%x:208f-[?] | %d:8336-%x:2090-[?] | %d:8337-%x:2091-[?] |
%d:8338-%x:2092-[?] | %d:8339-%x:2093-[?] | %d:8340-%x:2094-[?] | %d:8341-%x:2095-[?] |
%d:8342-%x:2096-[?] | %d:8343-%x:2097-[?] | %d:8344-%x:2098-[?] | %d:8345-%x:2099-[?] |
%d:8346-%x:209a-[?] | %d:8347-%x:209b-[?] | %d:8348-%x:209c-[?] | %d:8349-%x:209d-[?] |
%d:8350-%x:209e-[?] | %d:8351-%x:209f-[?] | %d:8352-%x:20a0-[?] | %d:8353-%x:20a1-[?] |
%d:8354-%x:20a2-[?] | %d:8355-%x:20a3-[?] | %d:8356-%x:20a4-[?] | %d:8357-%x:20a5-[?] |
%d:8358-%x:20a6-[?] | %d:8359-%x:20a7-[?] | %d:8360-%x:20a8-[?] | %d:8361-%x:20a9-[?] |
%d:8362-%x:20aa-[?] | %d:8363-%x:20ab-[?] | %d:8364-%x:20ac-[?] | %d:8365-%x:20ad-[?] |
%d:8366-%x:20ae-[?] | %d:8367-%x:20af-[?] | %d:8368-%x:20b0-[?] | %d:8369-%x:20b1-[?] |
%d:8370-%x:20b2-[?] | %d:8371-%x:20b3-[?] | %d:8372-%x:20b4-[?] | %d:8373-%x:20b5-[?] |
%d:8374-%x:20b6-[?] | %d:8375-%x:20b7-[?] | %d:8376-%x:20b8-[?] | %d:8377-%x:20b9-[?] |
%d:8378-%x:20ba-[?] | %d:8379-%x:20bb-[?] | %d:8380-%x:20bc-[?] | %d:8381-%x:20bd-[?] |
%d:8382-%x:20be-[?] | %d:8383-%x:20bf-[?] | %d:8384-%x:20c0-[?] | %d:8385-%x:20c1-[?] |
%d:8386-%x:20c2-[?] | %d:8387-%x:20c3-[?] | %d:8388-%x:20c4-[?] | %d:8389-%x:20c5-[?] |
%d:8390-%x:20c6-[?] | %d:8391-%x:20c7-[?] | %d:8392-%x:20c8-[?] | %d:8393-%x:20c9-[?] |
%d:8394-%x:20ca-[?] | %d:8395-%x:20cb-[?] | %d:8396-%x:20cc-[?] | %d:8397-%x:20cd-[?] |
%d:8398-%x:20ce-[?] | %d:8399-%x:20cf-[?] | %d:8400-%x:20d0-[?] | %d:8401-%x:20d1-[?] |
%d:8402-%x:20d2-[?] | %d:8403-%x:20d3-[?] | %d:8404-%x:20d4-[?] | %d:8405-%x:20d5-[?] |
%d:8406-%x:20d6-[?] | %d:8407-%x:20d7-[?] | %d:8408-%x:20d8-[?] | %d:8409-%x:20d9-[?] |
%d:8410-%x:20da-[?] | %d:8411-%x:20db-[?] | %d:8412-%x:20dc-[?] | %d:8413-%x:20dd-[?] |
%d:8414-%x:20de-[?] | %d:8415-%x:20df-[?] | %d:8416-%x:20e0-[?] | %d:8417-%x:20e1-[?] |
%d:8418-%x:20e2-[?] | %d:8419-%x:20e3-[?] | %d:8420-%x:20e4-[?] | %d:8421-%x:20e5-[?] |
%d:8422-%x:20e6-[?] | %d:8423-%x:20e7-[?] | %d:8424-%x:20e8-[?] | %d:8425-%x:20e9-[?] |
%d:8426-%x:20ea-[?] | %d:8427-%x:20eb-[?] | %d:8428-%x:20ec-[?] | %d:8429-%x:20ed-[?] |
%d:8430-%x:20ee-[?] | %d:8431-%x:20ef-[?] | %d:8432-%x:20f0-[?] | %d:8433-%x:20f1-[?] |
%d:8434-%x:20f2-[?] | %d:8435-%x:20f3-[?] | %d:8436-%x:20f4-[?] | %d:8437-%x:20f5-[?] |
%d:8438-%x:20f6-[?] | %d:8439-%x:20f7-[?] | %d:8440-%x:20f8-[?] | %d:8441-%x:20f9-[?] |
%d:8442-%x:20fa-[?] | %d:8443-%x:20fb-[?] | %d:8444-%x:20fc-[?] | %d:8445-%x:20fd-[?] |
%d:8446-%x:20fe-[?] | %d:8447-%x:20ff-[?] | %d:8448-%x:2100-[] |
```

●出力結果(2)

```
Code List(0x2100-0x2200)
%d:8448-%x:2100-[] | %d:8449-%x:2101-[] |
%d:8450-%x:2102-[] | %d:8451-%x:2103-[] | %d:8452-%x:2104-[] | %d:8453-%x:2105-[] |
%d:8454-%x:2106-[] | %d:8455-%x:2107-[] | %d:8456-%x:2108-] | %d:8457-%x:2109-[  ] |
%d:8458-%x:210a-[
] | %d:8459-%x:210b-[
                    ] | %d:8460-%x:210c-[
] |                                       ] | %d:8461-%x:210d-[
%d:8462-%x:210e-[] | %d:8463-%x:210f-[] | %d:8464-%x:2110-[] | %d:8465-%x:2111-[] |
%d:8466-%x:2112-[] | %d:8467-%x:2113-[] | %d:8468-%x:2114-[] | %d:8469-%x:2115-[] |
%d:8470-%x:2116-[] | %d:8471-%x:2117-[] | %d:8472-%x:2118-[] | %d:8473-%x:2119-[] |
%d:8474-%x:211a-[] | %d:8475-%x:211b-[ %d:8476-%x:211c-[] | %d:8477-%x:211d-[] |
%d:8478-%x:211e-[] | %d:8479-%x:211f-[] | %d:8480-%x:2120-[ ] | %d:8481-%x:2121-[!] |
%d:8482-%x:2122-["] | %d:8483-%x:2123-[#] | %d:8484-%x:2124-[$] | %d:8485-%x:2125-[%] |
%d:8486-%x:2126-[&] | %d:8487-%x:2127-['] | %d:8488-%x:2128-[(] | %d:8489-%x:2129-[)] |
%d:8490-%x:212a-[*] | %d:8491-%x:212b-[+] | %d:8492-%x:212c-[,] | %d:8493-%x:212d-[-] |
%d:8494-%x:212e-[.] | %d:8495-%x:212f-[/] | %d:8496-%x:2130-[0] | %d:8497-%x:2131-[1] |
%d:8498-%x:2132-[2] | %d:8499-%x:2133-[3] | %d:8500-%x:2134-[4] | %d:8501-%x:2135-[5] |
%d:8502-%x:2136-[6] | %d:8503-%x:2137-[7] | %d:8504-%x:2138-[8] | %d:8505-%x:2139-[9] |
%d:8506-%x:213a-[:] | %d:8507-%x:213b-[;] | %d:8508-%x:213c-[<] | %d:8509-%x:213d-[=] |
%d:8510-%x:213e-[>] | %d:8511-%x:213f-[?] | %d:8512-%x:2140-[@] | %d:8513-%x:2141-[A] |
%d:8514-%x:2142-[B] | %d:8515-%x:2143-[C] | %d:8516-%x:2144-[D] | %d:8517-%x:2145-[E] |
%d:8518-%x:2146-[F] | %d:8519-%x:2147-[G] | %d:8520-%x:2148-[H] | %d:8521-%x:2149-[I] |
%d:8522-%x:214a-[J] | %d:8523-%x:214b-[K] | %d:8524-%x:214c-[L] | %d:8525-%x:214d-[M] |
%d:8526-%x:214e-[N] | %d:8527-%x:214f-[O] | %d:8528-%x:2150-[P] | %d:8529-%x:2151-[Q] |
%d:8530-%x:2152-[R] | %d:8531-%x:2153-[S] | %d:8532-%x:2154-[T] | %d:8533-%x:2155-[U] |
%d:8534-%x:2156-[V] | %d:8535-%x:2157-[W] | %d:8536-%x:2158-[X] | %d:8537-%x:2159-[Y] |
%d:8538-%x:215a-[Z] | %d:8539-%x:215b-[[] | %d:8540-%x:215c-[\] | %d:8541-%x:215d-[]] |
%d:8542-%x:215e-[^] | %d:8543-%x:215f-[_] | %d:8544-%x:2160-[`] | %d:8545-%x:2161-[a] |
%d:8546-%x:2162-[b] | %d:8547-%x:2163-[c] | %d:8548-%x:2164-[d] | %d:8549-%x:2165-[e] |
%d:8550-%x:2166-[f] | %d:8551-%x:2167-[g] | %d:8552-%x:2168-[h] | %d:8553-%x:2169-[i] |
%d:8554-%x:216a-[j] | %d:8555-%x:216b-[k] | %d:8556-%x:216c-[l] | %d:8557-%x:216d-[m] |
%d:8558-%x:216e-[n] | %d:8559-%x:216f-[o] | %d:8560-%x:2170-[p] | %d:8561-%x:2171-[q] |
%d:8562-%x:2172-[r] | %d:8563-%x:2173-[s] | %d:8564-%x:2174-[t] | %d:8565-%x:2175-[u] |
%d:8566-%x:2176-[v] | %d:8567-%x:2177-[w] | %d:8568-%x:2178-[x] | %d:8569-%x:2179-[y] |
%d:8570-%x:217a-[z] | %d:8571-%x:217b-[{] | %d:8572-%x:217c-[|] | %d:8573-%x:217d-[}] |
%d:8574-%x:217e-[~] | %d:8575-%x:217f-[] | %d:8576-%x:2180-[?] | %d:8577-%x:2181-[?] |
%d:8578-%x:2182-[?] | %d:8579-%x:2183-[?] | %d:8580-%x:2184-[?] | %d:8581-%x:2185-[?] |
%d:8582-%x:2186-[?] | %d:8583-%x:2187-[?] | %d:8584-%x:2188-[?] | %d:8585-%x:2189-[?] |
%d:8586-%x:218a-[?] | %d:8587-%x:218b-[?] | %d:8588-%x:218c-[?] | %d:8589-%x:218d-[?] |
%d:8590-%x:218e-[?] | %d:8591-%x:218f-[?] | %d:8592-%x:2190-[?] | %d:8593-%x:2191-[?] |
%d:8594-%x:2192-[?] | %d:8595-%x:2193-[?] | %d:8596-%x:2194-[?] | %d:8597-%x:2195-[?] |
%d:8598-%x:2196-[?] | %d:8599-%x:2197-[?] | %d:8600-%x:2198-[?] | %d:8601-%x:2199-[?] |
%d:8602-%x:219a-[?] | %d:8603-%x:219b-[?] | %d:8604-%x:219c-[?] | %d:8605-%x:219d-[?] |
%d:8606-%x:219e-[?] | %d:8607-%x:219f-[?] | %d:8608-%x:21a0-[?] | %d:8609-%x:21a1-[?] |
%d:8610-%x:21a2-[?] | %d:8611-%x:21a3-[?] | %d:8612-%x:21a4-[?] | %d:8613-%x:21a5-[?] |
%d:8614-%x:21a6-[?] | %d:8615-%x:21a7-[?] | %d:8616-%x:21a8-[?] | %d:8617-%x:21a9-[?] |
%d:8618-%x:21aa-[?] | %d:8619-%x:21ab-[?] | %d:8620-%x:21ac-[?] | %d:8621-%x:21ad-[?] |
%d:8622-%x:21ae-[?] | %d:8623-%x:21af-[?] | %d:8624-%x:21b0-[?] | %d:8625-%x:21b1-[?] |
%d:8626-%x:21b2-[?] | %d:8627-%x:21b3-[?] | %d:8628-%x:21b4-[?] | %d:8629-%x:21b5-[?] |
%d:8630-%x:21b6-[?] | %d:8631-%x:21b7-[?] | %d:8632-%x:21b8-[?] | %d:8633-%x:21b9-[?] |
%d:8634-%x:21ba-[?] | %d:8635-%x:21bb-[?] | %d:8636-%x:21bc-[?] | %d:8637-%x:21bd-[?] |
%d:8638-%x:21be-[?] | %d:8639-%x:21bf-[?] | %d:8640-%x:21c0-[?] | %d:8641-%x:21c1-[?] |
%d:8642-%x:21c2-[?] | %d:8643-%x:21c3-[?] | %d:8644-%x:21c4-[?] | %d:8645-%x:21c5-[?] |
%d:8646-%x:21c6-[?] | %d:8647-%x:21c7-[?] | %d:8648-%x:21c8-[?] | %d:8649-%x:21c9-[?] |
%d:8650-%x:21ca-[?] | %d:8651-%x:21cb-[?] | %d:8652-%x:21cc-[?] | %d:8653-%x:21cd-[?] |
%d:8654-%x:21ce-[?] | %d:8655-%x:21cf-[?] | %d:8656-%x:21d0-[?] | %d:8657-%x:21d1-[?] |
%d:8658-%x:21d2-[?] | %d:8659-%x:21d3-[?] | %d:8660-%x:21d4-[?] | %d:8661-%x:21d5-[?] |
%d:8662-%x:21d6-[?] | %d:8663-%x:21d7-[?] | %d:8664-%x:21d8-[?] | %d:8665-%x:21d9-[?] |
%d:8666-%x:21da-[?] | %d:8667-%x:21db-[?] | %d:8668-%x:21dc-[?] | %d:8669-%x:21dd-[?] |
%d:8670-%x:21de-[?] | %d:8671-%x:21df-[?] | %d:8672-%x:21e0-[?] | %d:8673-%x:21e1-[?] |
%d:8674-%x:21e2-[?] | %d:8675-%x:21e3-[?] | %d:8676-%x:21e4-[?] | %d:8677-%x:21e5-[?] |
%d:8678-%x:21e6-[?] | %d:8679-%x:21e7-[?] | %d:8680-%x:21e8-[?] | %d:8681-%x:21e9-[?] |
%d:8682-%x:21ea-[?] | %d:8683-%x:21eb-[?] | %d:8684-%x:21ec-[?] | %d:8685-%x:21ed-[?] |
%d:8686-%x:21ee-[?] | %d:8687-%x:21ef-[?] | %d:8688-%x:21f0-[?] | %d:8689-%x:21f1-[?] |
%d:8690-%x:21f2-[?] | %d:8691-%x:21f3-[?] | %d:8692-%x:21f4-[?] | %d:8693-%x:21f5-[?] |
%d:8694-%x:21f6-[?] | %d:8695-%x:21f7-[?] | %d:8696-%x:21f8-[?] | %d:8697-%x:21f9-[?] |
%d:8698-%x:21fa-[?] | %d:8699-%x:21fb-[?] | %d:8700-%x:21fc-[?] | %d:8701-%x:21fd-[?] |
%d:8702-%x:21fe-[?] | %d:8703-%x:21ff-[?] | %d:8704-%x:2200-[] |
```

●考察
・出力結果より、0x2000、0x2100、0x2200 のコードから ASCII コード表と同じ配列の文字が続いてることがわかる.
・同様に、0x100-0x200、0x300-0x400、0x3000-0x3100 等で試しても同じ結果が得られた.
・よって、ASCII コードは 0x0 からだけではなく、0xn00（n は正の整数）のコードから繰り返し存在していることがわかった.

-あとがき-

a)参考文献

・『C 実践プログラミング 第三版』

b)感想と反省

・今回のレポートで、プログラムの基本「順次」「選択」「反復」の C 言語における表現を理解することができた。プログラミングによってできることが更に拡大した。

・相変わらずギリギリのレポート作成。今度こそ時間に余裕をもってレポートに臨みたいです…。