プログラミング1
Report#03

提出日:2009年5月28日(木)

所属:工学部情報工学科
学籍番号:095736E
氏名:玉城 翔

・サンプルプログラム1

```
#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 ===> ");
    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);
    else
      printf("-Not Printable character\n");

    if      (0x20 <= c && c <= 0x2f){
      puts("====> Scope_A\n");
    }else if(0x30 <= c && c <= 0x39){
      puts("====> Scope_B\n");
    }else if(0x3a <= c && c <= 0x40){
      puts("====> Scope_C\n");
    }else if(0x41 <= c && c <= 0x5a){
      puts("====> Scope_D\n");
    }else{
      puts("====> Scope_E\n");
    }
  }

  return(0);
}
```

出力結果

```
--------------------------------------------------------------------------------
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
--------------------------------------------------------------------------------
```

考察

　サンプルプログラムは出力結果から分かる様に、16進数を入力して、16進数に対応した整数型、ASCII文字を抽出しています。

　while文を使うことによって、プログラムを5回まで繰り返し使うことが出来る様になっている。

　if文を使うことによって、サンプルプログラムで指定した各範囲（例：0x20 <= c && c <= 0x7e）によって分けられるいくつかの文を、出力出来る様になっている。

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、非表示表示文字：Not Printable character、その他：misc.等

ソースプログラム

＜2＞095736E

```
/*
  Program    : sample#1.c
  Student-ID : 095736E
  Author     : TAMASHIRO,Kakeru
  UpDate     : 2009/05/28(Thu)
  Comments   : ASCII解析
*/

#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("16進数を入力せよ ===> ");
    fgets(line, sizeof(line), stdin);
    sscanf(line, "%x", &c);

    printf("ASCIIコード判断=%02d回目:%%d整数型(%3d)-%%x16
              進数(%2x)",count,c,c);
    if(0x20 <= c && c <= 0x7e)
      printf("-%%c文字(%c)\n",c);
    else
      printf("-特殊・英数文字はありません\n");

    if (0x20 <= c && c <= 0x2f){
      puts("====> 特殊記号文字です\n");
    }else if (0x3a <= c && c <= 0x40){
      puts("====> 特殊記号文字です\n");
    }else if (0x5b <= c && c <= 0x60){
      puts("====> 特殊記号文字です\n");
    }else if (0x7b <= c && c <= 0x7e){
      puts("====> 特殊記号文字です\n");
    }else if(0x30 <= c && c <= 0x39){
```

```
     puts("====>  数字です\n");
    }else if(0x41 <= c && c <= 0x5a){
     puts("====> 英大文字です\n");
    }else if(0x61 <= c && c <= 0x7a){
     puts("====> 英小文字です\n");
    }else{
     puts("====> 非表示文字です\n");
    }
   }

  return(0);
}
```

出力結果

```
16進数を入力せよ ===> 1
ASCIIコード判断=01回目:%d整数型(  1)-%x16進数( 1)-特殊・英数文字はありません
====> 非表示文字です

16進数を入力せよ ===> 20
ASCIIコード判断=02回目:%d整数型( 32)-%x16進数(20)-%c文字( )
====> 特殊記号文字です

16進数を入力せよ ===> 31
ASCIIコード判断=03回目:%d整数型( 49)-%x16進数(31)-%c文字(1)
====>  数字です

16進数を入力せよ ===> 41
ASCIIコード判断=04回目:%d整数型( 65)-%x16進数(41)-%c文字(A)
====> 英大文字です

16進数を入力せよ ===> 62
ASCIIコード判断=05回目:%d整数型( 98)-%x16進数(62)-%c文字(b)
====> 英小文字です
```

考察

　ASCIIコードを、非表示文字・特殊記号文字（例：&、$、?）・数字・英大文字・英小文字の5つの範囲で分けることにする。

　とりあえず5つの範囲なので、5回ほど調べることが出来ればいいので、while文はそのまま使わせてもらうことにする。また、printf関数・puts関数などで出力される文章を分かりやすい文章に変えておく。

各範囲を指定するプログラミング部分のif文を変えようと思うのだが、特殊記号文字がASCIIコード表ではまとまってはおらず、そこの部分だけは、if文の条件を4つ使うことにした。

出力結果からも分かる様に、プログラムは正常に作動しており、5つの範囲をきちんと判断して出力されている。

2.sample#2.c のプログラムの動作を考察せよ。

サンプルプログラム2

```
#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);
  }

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

  return(0);
}
```

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

　while文を使うことにより、変数countが1ずつプラスされ、値が5以上になったら、if文により動作を終了するプログラムと、for文を使って、while文と同じ動きをするプログラムである。

　while文のプログラムは、printf関数で出力される文章が数字の部分だけ1ずつ増えていき、5になった所で動作が終了している。つまり、5回同じプログラムが繰り返されていることが分かる。

　for文も一緒の動作をしていることから、for文は、while文の役割と同じ動作を行うことが出来ることが分かる。

サンプルプログラム3

```
#include <stdio.h>

int main(){
  int c;

  for(c = 0x20; c<=0x40; c++){
    if((c % 4) == 0) printf("\n");
    printf("%%x(%x)-%%c(%c) | ",c,c);
  }
  printf("\n");

  return(0);
}
```

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文を使うことにより、16進数（20）から16進数（40）までの間を繰り返し処理するプログラムを作り、if文で、4で割ったが余りが0になった時に改行を行うプログラムがつくられている。

　上記の様なプログラムを実行することにより、出力結果の様なASCIIコード表を作ることができる。

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。

ソースプログラム

```
--------------------------------------------------------------------------------
/*
 Program    : sample#3.c
 Student-ID : 095736E
 Author     : TAMASHIRO,Kakeru
 UpDate     : 2009/05/28(Thu)
 Comments   : ASCIIコード表
*/

#include <stdio.h>

int main(){
 int c;
 printf("\n");
 puts("ASCIIコード表（表示可能文字）");
 printf("\n");
 for(c = 0x20; c<=0x7e; c++){
  if((c % 3) == 0) printf("\n");
  printf("%%x16進数(%x)-%%c文字(%c)|",c,c);
 }
 printf("\n");
 printf("\n");

 return(0);
}
--------------------------------------------------------------------------------
```

出力結果

ASCIIコード表（表示可能文字）

%x16進数(20)-%c文字( )|
%x16進数(21)-%c文字(!)|%x16進数(22)-%c文字(")|%x16進数(23)-%c文字(#)|
%x16進数(24)-%c文字($)|%x16進数(25)-%c文字(%)|%x16進数(26)-%c文字(&)|
%x16進数(27)-%c文字(')|%x16進数(28)-%c文字(()|%x16進数(29)-%c文字())|
%x16進数(2a)-%c文字(*)|%x16進数(2b)-%c文字(+)|%x16進数(2c)-%c文字(,)|
%x16進数(2d)-%c文字(-)|%x16進数(2e)-%c文字(.)|%x16進数(2f)-%c文字(/)|
%x16進数(30)-%c文字(0)|%x16進数(31)-%c文字(1)|%x16進数(32)-%c文字(2)|
%x16進数(33)-%c文字(3)|%x16進数(34)-%c文字(4)|%x16進数(35)-%c文字(5)|
%x16進数(36)-%c文字(6)|%x16進数(37)-%c文字(7)|%x16進数(38)-%c文字(8)|
%x16進数(39)-%c文字(9)|%x16進数(3a)-%c文字(:)|%x16進数(3b)-%c文字(;)|
%x16進数(3c)-%c文字(<)|%x16進数(3d)-%c文字(=)|%x16進数(3e)-%c文字(>)|
%x16進数(3f)-%c文字(?)|%x16進数(40)-%c文字(@)|%x16進数(41)-%c文字(A)|
%x16進数(42)-%c文字(B)|%x16進数(43)-%c文字(C)|%x16進数(44)-%c文字(D)|
%x16進数(45)-%c文字(E)|%x16進数(46)-%c文字(F)|%x16進数(47)-%c文字(G)|
%x16進数(48)-%c文字(H)|%x16進数(49)-%c文字(I)|%x16進数(4a)-%c文字(J)|
%x16進数(4b)-%c文字(K)|%x16進数(4c)-%c文字(L)|%x16進数(4d)-%c文字(M)|
%x16進数(4e)-%c文字(N)|%x16進数(4f)-%c文字(O)|%x16進数(50)-%c文字(P)|
%x16進数(51)-%c文字(Q)|%x16進数(52)-%c文字(R)|%x16進数(53)-%c文字(S)|
%x16進数(54)-%c文字(T)|%x16進数(55)-%c文字(U)|%x16進数(56)-%c文字(V)|
%x16進数(57)-%c文字(W)|%x16進数(58)-%c文字(X)|%x16進数(59)-%c文字(Y)|
%x16進数(5a)-%c文字(Z)|%x16進数(5b)-%c文字([)|%x16進数(5c)-%c文字(\)|
%x16進数(5d)-%c文字(])|%x16進数(5e)-%c文字(^)|%x16進数(5f)-%c文字(_)|
%x16進数(60)-%c文字(`)|%x16進数(61)-%c文字(a)|%x16進数(62)-%c文字(b)|
%x16進数(63)-%c文字(c)|%x16進数(64)-%c文字(d)|%x16進数(65)-%c文字(e)|
%x16進数(66)-%c文字(f)|%x16進数(67)-%c文字(g)|%x16進数(68)-%c文字(h)|
%x16進数(69)-%c文字(i)|%x16進数(6a)-%c文字(j)|%x16進数(6b)-%c文字(k)|
%x16進数(6c)-%c文字(l)|%x16進数(6d)-%c文字(m)|%x16進数(6e)-%c文字(n)|
%x16進数(6f)-%c文字(o)|%x16進数(70)-%c文字(p)|%x16進数(71)-%c文字(q)|
%x16進数(72)-%c文字(r)|%x16進数(73)-%c文字(s)|%x16進数(74)-%c文字(t)|
%x16進数(75)-%c文字(u)|%x16進数(76)-%c文字(v)|%x16進数(77)-%c文字(w)|
%x16進数(78)-%c文字(x)|%x16進数(79)-%c文字(y)|%x16進数(7a)-%c文字(z)|
%x16進数(7b)-%c文字({)|%x16進数(7c)-%c文字(|)|%x16進数(7d)-%c文字(})|
%x16進数(7e)-%c文字(~)|

考察

　表示可能文字を出力せよなので、for文の条件の部分を、範囲を16進数（20）～16進数（7e）までに変更する。

　printf関数の文章を追加したので文章が長くなったため、if文の条件式を、3で割った余りが0になったら改行するようにした。そのことにより、きれいに文章が並んだASCIIコード

<8>095736E

表(表示可能文字)が出力された。

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')?

ソースプログラム

```
--------------------------------------------------------------------------------
/*
  Program    : sample#4.c
  Student-ID : 095736E
  Author     : TAMASHIRO,Kakeru
  UpDate     : 2009/05/28(Thu)
  Comments   : 文字の演算
*/

#include <stdio.h>

#define FALSE 0
#define TRUE  !FALSE

int main(){

  int  a,  b,  c,  count;
  char line[128];

  count = 0;

  while(TRUE){
    count++;
    if(count > 3) break;

    printf("文字の演算%0d回目\n",count);
    printf("文字を入力せよ ===> ");
    fgets(line, sizeof(line), stdin);
    sscanf(line, "%c", &a);

    printf("文字を入力せよ ===> ");
    fgets(line, sizeof(line), stdin);
    sscanf(line, "%c", &b);

  c = a - b;
```

```
  printf("%%d   整数型    = %09d\n",c);
  printf("%%x  16進数   = %09x\n",c);
  printf("%%c ASCIIコード = \"%2c\"\n",c);
  printf("\n");

  }

  return(0);
}
```

出力結果

```
文字の演算1回目
文字を入力せよ ===> a
文字を入力せよ ===> A
%d   整数型    = 000000032
%x  16進数   = 000000020
%c ASCIIコード = "  "

文字の演算2回目
文字を入力せよ ===> f
文字を入力せよ ===> A
%d   整数型    = 000000037
%x  16進数   = 000000025
%c ASCIIコード = " %"

文字の演算3回目
文字を入力せよ ===> z
文字を入力せよ ===> A
%d   整数型    = 000000057
%x  16進数   = 000000039
%c ASCIIコード = " 9"
```

考察

文字の演算についての考察なので、上記の様なプログラムを作ってみた。

while文を使って、3回繰り返し同じプログラムを実行するようにする。

fgets関数とsscanf関数を使って、出力画面で演算に使う文字を入力することが出来る様にする。

入力した文字を実際に演算し、その値を変数cに代入する。

演算結果を、整数型・16進数・ASCIIコードで出力する。

出力結果から、文字の演算を行うと、その答えはASCIIコード表の文字と対応した整数

型どうしの演算結果となっている。

例えば、文字の演算1回目は”a”は整数型だと”97”,”A”は整数型だと”65”で、演算式は97−65＝32となり、出力結果が示す整数型の表示が”32”となる。また、16進数・ASCIIコードは、ASCIIコード表で整数型（32）と対応する値が表示されている。

文字の演算2・3回目も同様の結果が得られていることから、このプログラムは正常に動作していることが伺える。

反省・感想

while文とfor文は、動作的にはほぼ同じ動きをするのだが、どれが使いやすいのか、どういう場面でどれを使えばいいのかなどの判断が、私には今の所出来ないので、そこんところはもっとwhile文・for文の意味を理解しなければいけないと思う。

参考文献

C実践プログラミング 第3版：
著 Steve Oualline　監訳 望月 康司　　訳 谷口 功