プログラミング 1

Report#8

提出日　：2013 年 7 月 18 日
所　属　：工学部情報工学科
学籍番号：e135732J
氏　名　：前城　健太郎

Report#8

入力した正の整数を降順に並べ換えて出力するプログラムを作成せよ。
プログラムは個別にコンパイルし、make コマンドで実行すること。

1．作成した Makefile

1.1 ソースコード全体 (Makefile)

```
cardinal.e:cardinal.o conv10.o select_sort.o print_num.o msg.o
    gcc -o cardinal.e cardinal.o conv10.o select_sort.o print_num.o msg.o

cardinal.o:cardinal.c
    gcc -c cardinal.c

conv10.o:conv10.c
    gcc -c conv10.c

select_sort.o:select_sort.c
    gcc -c select_sort.c

print_num.o:print_num.c
    gcc -c print_num.c

msg.o:msg.c
    gcc -c msg.c
```

※ここからの考察は,プログラムを実行した際に入力を順番に x2ab,694,9801,07614 を行った場合を考える.

2. carbinal ファイルの穴埋めと考察

2.1 ソースコード全体(carbinal.c)

```
#include <stdio.h>
#include <string.h>
#define  MAX 256

void conv10(char **x, int *k, int n);
void select_sort(int x[], int m[], int n);
void print_num(char *x[], int m[], int n);
void msg();

int main(){
  char *dt[50], num[10], buf[MAX], *p=buf;
  int  n=0, len, i10[50], move[50];

  puts("-------- Input");
  while(gets(num) != NULL) {
    len = strlen(num);
    if(p > buf+MAX-(len+1) )  break;
    strcpy(p, num);
    dt[n] = p;
    p += (len+1);
    n++;
  }
  puts("-------- Result");
  conv10(dt,i10,n);
  select_sort(i10,move,n);
  print_num(dt,move,n);
  msg();

  return(0);
}
```

2.2 考察

a) このファイル(cardinal.c)では各配列,ポインタ,変数の宣言,文字列の取り込み, 各サブルーチンへの引数引渡し,呼み込みなどが行われている。
b) ここではまず,文字の格納をしている 15-22 行目の文字の格納の考察をする.

1) 15 行目において,gets()関数によって配列 num[]に文字列を格納する.
【図 2.1】
{配列 num[]}

x	2	a	b	¥0					

2)16 行目において,strlen()関数によって配列 num[]に格納されている文字の数を変数 len に格納する.ただし NULL は含まない.
【図 2.2】
{変数 len}

4

3)18 行目において,配列 num 内の文字列をポインタ変数 p が指す変数 buf[]にコピーしている.
【図 2.3】
{変数 p}

(address)

{配列 buf[]}

x	2	a	b	¥0															……

4)19 行目において,ポインタ変数 p に格納されているアドレスを配列 dt[]に格納する.
【図 2.4】
{配列 dt[]}

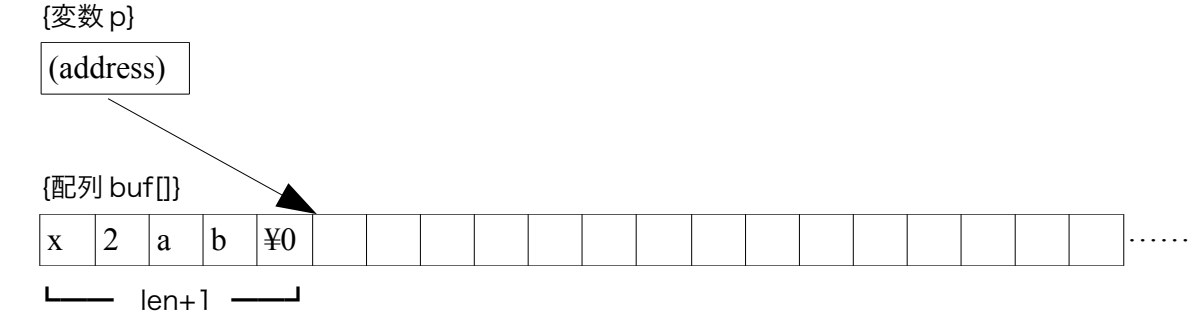
dt[0]									

{ 配列 buf[]}

x	2	a	b	¥0															……

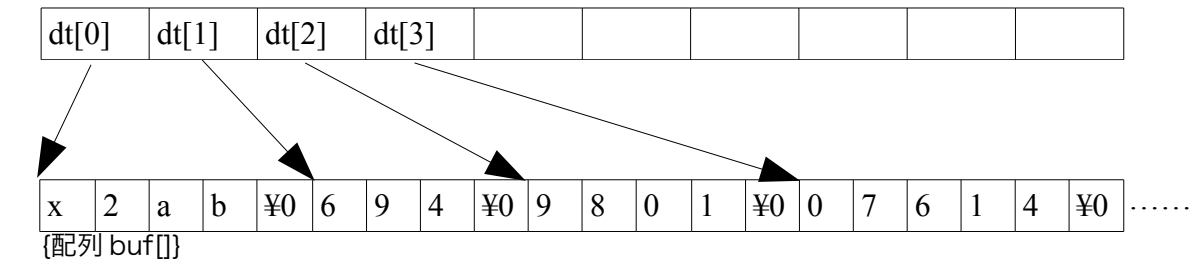
5)20 行目において,ポインタ変数 p に格納されているアドレスに(len+1)を足している.これは次の文字を buf[]に格納する際に p が指す要素を次の文字の先頭に移すためである.

【図 2.5】



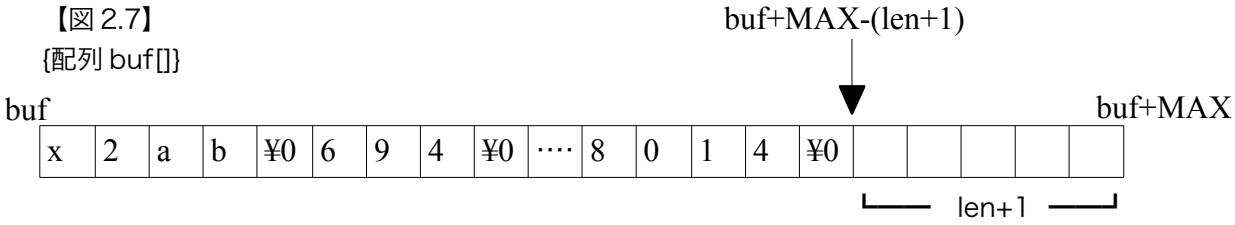
6)以上の作業を繰り返すことで,配列 dt[],配列 buf[]に以下のように文字が格納される.

【図 2.6】



7)また,17 行目の if 文の条件式は,配列 buf に格納できる容量が無いかどうかを判断するためである.図のように,ポインタ変数 p が指すアドレスが buf+MAX-(len+1)より大きいと数をすべて格納できる容量が無い.

【図 2.7】



3. conv10 ファイルの穴埋めと考察

3.1 ソースコード全体(conv10.c)

```
1   void conv10(char **x, int *k, int n){
2     while(n-- > 0){
3       switch(**x){
4       case '0' :
5         sscanf(*x+1, "%o", k);
6         break;
7       case 'x' :
8       case 'X' :
9         sscanf(*x+1, "%x", k);
10        break;
11      default :
12        sscanf(*x, "%u", k);
13        break;
14      }
15      x++;
16      k++;
17    }
18  }
```

3.2 考察

a) このファイル(conv10.c)では格納されている文字列の先頭の文字によって読み込む際の基数を決定し読み込む処理を行う.

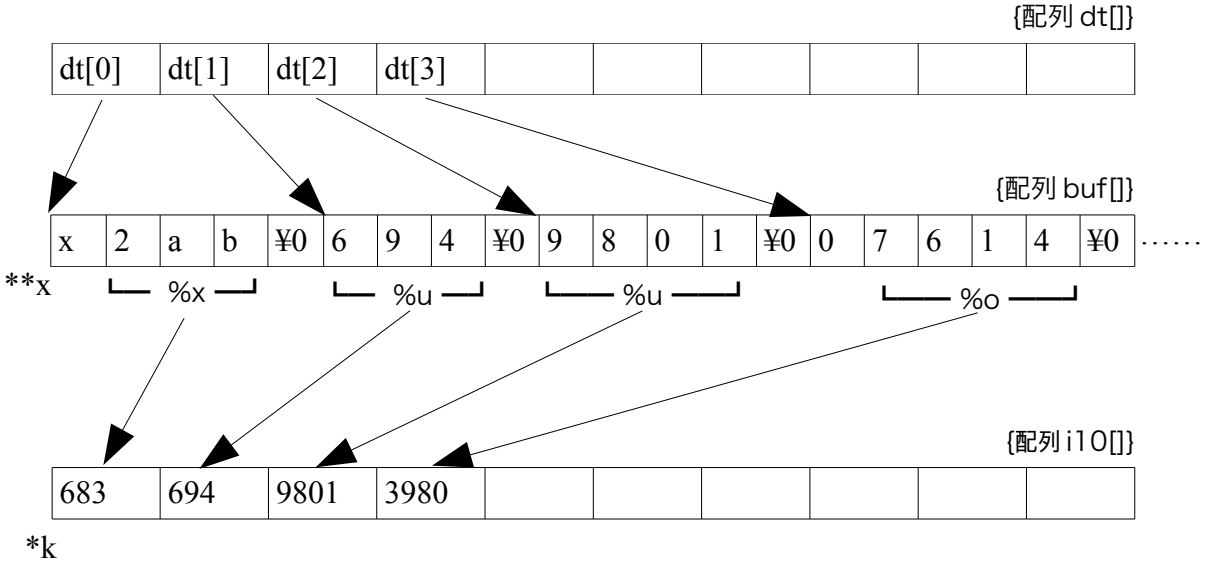
b) carbinal プログラムからは dt,i10,n を受け取っている.

dt	char **x
i10	int *k
n	int n

c) 3-14 行目において,switch 文でポインタ*x(配列 dt[])に格納されている先頭の 1 文字を判断して数を改めて格納している.

d) 先頭が x なら sscanf()関数によって 16 進数として,0 なら 8 進数として,それ以外は 10 進数として格納する.

【図 3.1】



e) while 文によって格納した数だけ繰り返し,15-16 行目でポインタ変数 x と k のアドレスをインクリメントしている.これにより配列 buf,配列 i10 に格納する場所を次の空いてる場所へずらしていく.

4. select_sort ファイルの穴埋めと考察

4.1 ソースコード全体(select_sort.c)

```
1   void select_sort(int x[], int m[], int n){
2     int  i, j, k, w;
3
4     for(i=0; i<n; i++)  m[i]=i;
5     for(i=0; i<n-1; i++){
6       k=i;
7       for(j=i+1; j<n; j++)
8         if( x[j] > x[k] )  k=j;
9       w    = x[i];
10      x[i] = x[k];
11      x[k] = w;
12      w    = m[i];
13      m[i] = m[k];
14      m[k] = w;
15    }
16  }
```

4.2 考察

a) このファイル(select_sort.c)では,値に番号を割り振り,値の大小を判断し割り振った番号と共に入れ替えてソートを行う.

b) carbinal プログラムからは i10,move,n を受け取っている.

i10	char x[]
move	int m[]
n	int n

c) ソートのアルゴリズムには「基本選択法」を使用している.これは,「要素の端から順に調べ、一番大きい値を持つ要素を抜き出して最初の要素と交換し、さらに未整列の部分から一番大きい値を持つ要素を抜き出して 2 番目の要素と交換し…という操作を繰り返す」というものである.

d) 4 行目において配列 m[](配列 move[])に,0 から順番に入力された文字列の個数だけ格納している.

e) 5-15 行目が実際のソートの手順になっている.以下に手順を示す.

1)k=0 からスタート.(5 行目の for 文ループ)
k[0]の要素をすべての要素と比較していき,一番大きいものと数を入れ替える.(7 行目の for 文ループ)

{配列 i10[]}

683	694	9801	3980						

2)次は k=1 から.
k[1]の要素と未整理の要素を比較していき,一番大きいものと数を入れ替える.

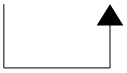
{配列 i10[]}

9801	**694**	683	3980						

3)次は k=2
k[2]の要素と未整理の要素を比較していき,一番大きいものと数を入れ替える.

{配列 i10[]}

9801	3980	**683**	694						



4)最高でもすべての要素の数より 1 小さい回数を行えばソートは完了する

{配列 i10[]}

9801	3980	694	683						

f) このプログラムでは,同時に配列 m[]も同様な順番で数をソートしている.以下はソートしたあとの配列 m[]の値である.

{配列 m[]}

2	3	0	1						

5. print_num ファイルの穴埋めと考察

5.1 ソースコード全体(print_num.c)

```
1  void print_num(char *x[], int m[], int n){
2    int  i;
3
4    for(i=0; i<n; i++){
5      puts( x[m[i]] );
6    }
7  }
```

5.2 考察

a) このファイル(print_num.c)では, "select_sort.c"で割り振られた番号を用いてソートされた順通りに文字列を出力する.

b) carbinal プログラムからは dt,move,n を受け取っている.

dt	char *x[]
move	int m[]
n	int n

c) ソートされた数を直接受け取って値を表示するのではなく,同じ順番でソートされた配列 m[]を用いて間接的に表示している.

6. msg ファイル

6.1 ソースコード全体(msg.c)

```
1  int msg(){
2    printf("#### Message from C #### By Taniguchi\n");
3    return(0);
4  }
```

7. 実行結果

```
-------- Input
warning: this program uses gets(), which is unsafe.
x2ab
694
9801
07614
-------- Result
9801
07614
694
x2ab
#### Message from C #### By Taniguchi
```

7. プログラム全体の確認

7.1 すべてのソースコード全体

---carbinal.c

```
#include <stdio.h>
#include <string.h>
#define  MAX 256

void conv10(char **x, int *k, int n);
void select_sort(int x[], int m[], int n);
void print_num(char *x[], int m[], int n);
void msg();

int main(){
  char *dt[50], num[10], buf[MAX], *p=buf;
  int  n=0, len, i10[50], move[50];
  int a;
  puts("-------- Input");
  while(gets(num) != NULL) {
    len = strlen(num);
    if(p > buf+MAX-(len+1) )  break;
    strcpy(p, num);
    dt[n] = p;
    p += (len+1);
    n++;
  }
  puts("-------- Result");
  for(a=0;n>a;a++)
    printf("dt[%d] --> %s\n",a,dt[a]);
  conv10(dt,i10,n);
  select_sort(i10,move,n);
  print_num(dt,move,n);
  msg();

  return(0);
}
```

---conv10.c

```
#include <stdio.h>
void conv10(char **x, int *k, int n){
  puts("--conv10--");
  int a=0;
  while(n-- > 0){
    switch(**x){
    case '0' :
      sscanf(*x+1, "%o", k);
      break;
    case 'x' :
    case 'X' :
      sscanf(*x+1, "%x", k);
      break;
    default :
      sscanf(*x, "%u", k);
      break;
    }
    printf("i10[%d] ==> %d\n",a,*k);
    x++;
    k++;
    a++;
  }
}
```

---select_sort.c

```
#include <stdio.h>
void select_sort(int x[], int m[], int n){
  int  i, j, k, w, a, b;

  puts("--sort--");
  for(i=0; i<n; i++)  m[i]=i;
  b=i;
  for(i=0; i<n-1; i++){
    k=i;
    for(j=i+1; j<n; j++)
      if( x[j] > x[k] )  k=j;
    for(a=0;a<b;a++)
      printf("m[%d] ==> %d\n",a,m[a]);
    puts("-----");
    w    = x[i];
    x[i] = x[k];
    x[k] = w;
    w    = m[i];
    m[i] = m[k];
    m[k] = w;
  }
}

```

---print_num.c

```
void print_num(char *x[], int m[], int n){
  int  i;

  for(i=0; i<n; i++){
    puts( x[m[i]] );
  }
}
```

---msg.c

```
int msg(){
  printf("#### Message from C #### By Taniguchi\n");
  return(0);
}
```

7.2 実行結果

```
-------- Input
warning: this program uses gets(), which is unsafe.
x2ab
694
9801
07614
-------- Result
dt[0] --> x2ab
dt[1] --> 694
dt[2] --> 9801
dt[3] --> 07614
--conv10--
i10[0] ==> 683
i10[1] ==> 694
i10[2] ==> 9801
i10[3] ==> 3980
--sort--
m[0] ==> 0
m[1] ==> 1
m[2] ==> 2
m[3] ==> 3
-----
m[0] ==> 2
m[1] ==> 1
m[2] ==> 0
m[3] ==> 3
-----
m[0] ==> 2
m[1] ==> 3
m[2] ==> 0
m[3] ==> 1
-----
9801
07614
694
x2ab
#### Message from C #### By Taniguchi
```

7.3 考察

a) プログラムにコードを追加し,配列に格納されている数を出力し今までの考察の確認をした.

b) 実行結果を観察すると,考察が正しいことがわかる.

--あとがき--

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

2.反省・感想

ラストのレポート.今までの集大成だと感じました.

一番レポートしてて楽しかったです.