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。

Cardinal.c

```
1.  #include <stdio.h>
2.  #include <string.h>/*strlen,strcpyを使うのに必要なヘッダ*/
3.  #define   MAX 256/*マクロ定義によりMAX =256を表す。*/
4.
5.  void conv10(char **x, int *k, int n);/*関数conv10のプロトタイプ宣言*/
6.  void select_sort(int x[], int m[], int n);/*関数select_sortのプロトタイプ宣言*/
7.  void print_num(char *x[], int m[], int n);/*関数print_numのプロトタイプ宣言*/
8.  void msg();/*関数msgのプロトタイプ宣言*/
9.
10. int main(){
11.       char *dt[50], num[10], buf[MAX], *p=buf;/*char型で、ポインタ変数dt,配列num,配列
    bufの宣言。*pにbufの値を代入。*/
12.       int   n=0, len, i10[50], move[50];/*int型で変数n ,配列i10,配列moveの宣言*/
13.
14.       puts("-------- Input");
15.       while(gets(num) != NULL){/*numに入力された文字を格納。入力したものが、NULL
    でなければ繰り返し。*/
16.                 len = strlen(num);/*strlenにより格納された文字の長さを判別、その後lenに
    長さを代入。
17.                 if(p > buf + MAX - len - 1）  break;/*配列bufに文字が格納できなくなったと
    き終了。
18.                 strcpy(p, num);/*strcpyを使いnumの文字列をアドレスpにコピー*/
19.                 dt[n] = p;/*配列dt[]に文字を格納した文字列の先頭アドレスをコピー*/
20.                 p += len + 1;/*pの先頭アドレスを文字数+1分だけ移動+1するのはNULLが
    文字列の最後に入るから。*/
21.                 n++;/*配列を1ずらす*/
22.                 }
23.       puts("-------- Result");
24.       conv10(dt, i10, n);/*関数conv10にdt,i10,nを引き渡している*/
25.       select_sort(i10, move, n);/*関数select_sortにi10,move,nを引き渡している*/
26.       print_num(dt, move, n);/*関数print_numにdt,move,nを引き渡している*/
```

27.　　msg();

28.　　return(0);

29. }

解説・考察

実行させたプログラムがどのようになっているかを図示する

n=0 のとき

num

ごみ	ごみ	ごみ	ごみ	ごみ	ごみ	ごみ	ごみ	ごみ	ごみ
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]

↓

Gets(num)　x21 を入力

↓

num

x	2	1	NULL	ごみ	ごみ	ごみ	ごみ	ごみ	ごみ
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]

↓

Strlen(num)文字の長さを判別。ただし NULL は数に入らない。　len に値を代入

↓

len | 3 |

↓

num

x	2	1	NULL	ごみ	ごみ	ごみ	ごみ	ごみ	ごみ
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]

↓

strcpy(p, num)により

↓

buf

ごみ	ごみ	ごみ	ごみ	ごみ	ごみ	ごみ	・・・・・	ごみ	ごみ	ごみ	ごみ	ごみ	ごみ
[0]	[1]	[2]	[3]	[4]	[5]	[6]	・・・・・	[250]	[251]	[252]	[253]	[254]	[255]

↓

↓

x	2	1	NULL	ごみ	ごみ	ごみ	・・・・	ごみ	ごみ	ごみ	ごみ	ごみ	ごみ
[0]	[1]	[2]	[3]	[4]	[5]	[6]	・・・・	[250]	[251]	[252]	[253]	[254]	[255]

↑

P(p は最初 buf の先頭アドレスをさす)

↓

dt

ごみ	ごみ	ごみ	ごみ	ごみ	・・・	ごみ	ごみ	ごみ	ごみ	ごみ
[0]	[1]	[2]	[3]	[4]	・・・	[45]	[46]	[47]	[48]	[49]

↓

buf[0]	ごみ	ごみ	ごみ	ごみ	・・・	ごみ	ごみ	ごみ	ごみ	ごみ
[0]	[1]	[2]	[3]	[4]	・・・	[45]	[46]	[47]	[48]	[49]

P+ =len +1(+1 しているのは NULL のことを考慮しているからである)

P=buf[4]

n=1 のとき

num(先ほど入力された物が残っている)

x	2	1	ごみ	ごみ	ごみ	ごみ	ごみ	ごみ	ごみ
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]

↓

Gets(num)　021 を入力

↓

num

0	2	1	NULL	ごみ	ごみ	ごみ	ごみ	ごみ	ごみ
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]

↓

Strlen(num)文字の長さを判別。ただし NULL は数に入らない。　len に値を代入

↓

len | 3 |

↓

num

x	2	1	NULL	ごみ	ごみ	ごみ	ごみ	ごみ	ごみ
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]

↓

strcpy(p(p=buf[4]となっている), num)により

↓

buf

x	2	1	NULL	ごみ	ごみ	ごみ	・・・	ごみ	ごみ	ごみ	ごみ	ごみ	ごみ
[0]	[1]	[2]	[3]	[4]	[5]	[6]	・・・	[250]	[251]	[252]	[253]	[254]	[255[

↓

↓

x	2	1	NULL	0	2	1	・・・	ごみ	ごみ	ごみ	ごみ	ごみ	ごみ
[0]	[1]	[2]	[3]	[4]	[5]	[6]	・・・	[250]	[251]	[252]	[253]	[254]	[255]

↑

P

↓

dt

buf[0]	ごみ	ごみ	ごみ	ごみ	・・・	ごみ	ごみ	ごみ	ごみ	ごみ
[0]	[1]	[2]	[3]	[4]	・・・	[45]	[46]	[47]	[48]	[49]

↓

buf[0]	buf[4]	ごみ	ごみ	ごみ	・・・	ごみ	ごみ	ごみ	ごみ	ごみ
[0]	[1]	[2]	[3]	[4]	・・・	[45]	[46]	[47]	[48]	[49]

P+ =len +1(+1 しているのは NULL のことを考慮しているからである)

P=buf[8]

以下同様にしてこれが続くものである

If 文により break するとき

buf

・・・・	NULL	0	2	1	NULL		
・・・・	[249]	[250]	[251]	[252]	[253]	[254]	[255]

N=k-1 の時上記の状態であるとする

n=k のとき

num

0	2	1	ごみ	ごみ	ごみ	ごみ	ごみ	ごみ	ごみ

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]

↓

Gets(num)　321 を入力

↓

num

3	2	1	NULL	ごみ	ごみ	ごみ	ごみ	ごみ	ごみ
[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]

↓

Strlen(num)文字の長さを判別。ただし NULL は数に入らない。　len に値を代入

↓

len | 3 |

↓

If 文(p > buf + MAX - len - 1)p はここでは p=buf[254]

buf[254]>buf[0+255-3-1]=buf[251]となり()の中が真となるので break

・・・・	NULL	0	2	1	NULL	ごみ	ごみ
・・・・	[249]	[250]	[251]	[252]	[253]	[254]	[255]

↓

puts("-------- Result");

:

以下 Conv10 に続く。

Conv10.c

```
1.  void conv10(char **x, int *k, int n){/*dt,i10,nがそれぞれ**x,*k,nに対応している*/
2.      while(n-- > 0){/*nから1引いたものが0より大きければ繰り返す。文字が入力されているか判断する役割と思われる。*/
3.          switch(**x){/*  **xの値により作業が分岐する。*/
4.              case '0' :/*8進数の場合*/
5.                  sscanf(*x + 1, "%o", k);/*アドレスに1を足したところから8進数としてkに入力*/
6.                  break;
7.              case 'x' :/*16進数の場合大文字、小文字の判別をしない*/
8.              case 'X' :
9.                  sscanf(*x + 1, "%x", k);/*アドレスに1を足したところから16進数としてkに入力*/
10.                 break;
```

```
11.                             default :/*その他(10進数の場合)*/
12.                                     sscanf(*x, "%u", k);/*10進数としてkに代入*/
13.                                     break;
14.
15.                     }
16.                     x++;/*   *xのアドレスに1を足す。*/
17.                     k++;/*kのアドレスに1を足す。*/
18.
19.             }
20.             return;
21.  }
```

解説・考察

2 行目:なにも数字が代入されていないときに n-- =-1>0 は偽なので何もせず終了。

実行された様子を絵にしていく。

**x(dt[])

*x=0 のとき

buf[0]	buf[4]	buf[8]	buf[12]	buf[15]		
[0]	[1]	[2]	[3]	[4]	[5]	・・・・

x(buf[0])

x	2	1	NULL

*k

21(16 進数)						
[0]	[1]	[2]	[3]	[4]	[5]	[6]

*x=1 のとき

buf[0]	buf[4]	buf[8]	buf[12]	buf[15]		
[0]	[1]	[2]	[3]	[4]	[5]	・・・・

x(buf[4])

0	2	1	NULL

*k

21(16 進数)	21(8 進数)					
[0]	[1]	[2]	[3]	[4]	[5]	[6]

*x=3 のとき

buf[0]	buf[4]	buf[8]	buf[12]	buf[15]		
[0]	[1]	[2]	[3]	[4]	[5]	・・・・

x(buf[12])

2	2	NULL

*k

21(16 進数)	21(8 進数)	55(8 進数)	22			
[0]	[1]	[2]	[3]	[4]	[5]	[6]

以下続いていく。

conv10 関数では、switch 文を使用して、リストのそれぞれの文字列の先頭文字が、'0'、'x'、'X'、またはそれ以外の文字かという判定をして、その文字列が何進数の表記方法をとっているかを区別している。その区別した場合にあわせて、10 進数なら、%u で最初の 1 文字目から、16 進数、8 進数なら、それぞれ、%x や %o で 2 文字目から、読み出している

Select_sort.c

```
1.  void select_sort(int x[], int m[], int n){/* i10, move, nがそれぞれ配列x,配列m,変数n,に対応している。*/
2.      int   i, j, k, w;/*変数の宣言*/
3.
4.      for(i=0; i<n; i++)   m[i]=i;/*配列mの番号と値を一致させる。*/
5.      for(i=0; i<n-1; i++){/*iの値がn-1より大きくなるまで繰り返し、繰り返すたびにiにインクリメントする。
6.          k=i;/*kの初期値をiと同じにする。k は最終的にx[i]以降の最大値の要素の添字となる */
7.          for(j=i+1; j<n; j++)/*jの値がn以上になるまでインクリメントする。*/
8.              if(x[k] < x[j]) k = j;/*x[k]の値がx[j]の値よりも小さいときkの値をjに置き換える。
9.          w      = x[i]; /*wにx[i]の値を代入する*/
10.         x[i] = x[k];   /*x[i]にx[k]の値を代入する*/
```

```
11.                 x[k] = w;       /*x[k]にwの値を代入する*/
12.                 w       = m[i];/*wにm[i]の値を代入する*/
13.                 m[i] = m[k]; /*m[i]にm[k]の値を代入する*/
14.                 m[k] = w;     /*m[k]にwの値を代入する*/
15.
16.         }
17.
18.  }
```

解説・考察

x

x21	021	055	22	・・・・			
[0]	[1]	[2]	[3]	・・・・	[47]	[48]	[49]

↑

x[i],x[k],x[j]

↓

for(j=i+1; j<n; j++) if(x[k] < x[j]) k = j;

↓

x21	021	055	22	・・・・			
[0]	[1]	[2]	[3]	・・・・	[47]	[48]	[49]

↑ ↑

x[i],x[k] x[j]

↓

x21	021	055	22	・・・・			
[0]	[1]	[2]	[3]	・・・・	[47]	[48]	[49]

↑ ↑

x[i] x[j], x[k]

↓

w	x[i](i=0)	x[k](k=2)
	x21	055

↓

w	x[i](i=0)	x[k](k=2)
x21	x21	055

↓

w	x[i](i=0)	x[k](k=2)
x21	055	055

↓

w	x[i](i=0)	x[k](k=2)
x21	055	x21

↓

055	021	x21	22	・・・・			
[0]	[1]	[2]	[3]	・・・・	[47]	[48]	[49]

↑ ↑

X[i] x[k],x[j]

i にインクリメント

↓

055	021	x21	22	・・・・			
[0]	[1]	[2]	[3]	・・・・	[47]	[48]	[49]

↑ ↑

X[i],x[k] x[j]

↓

055	021	x21	22	・・・・			
[0]	[1]	[2]	[3]	・・・・	[47]	[48]	[49]

↑ ↑

X[i] x[k],x[j]

↓

w	x[i](i=1)	x[k](k=2)
x21	021	x21

↓

w	x[i](i=1)	x[k](k=2)
021	021	x21

↓

w	x[i](i=0)	x[k](k=2)
021	x21	021

↓

055	x21	021	22	・・・・			
[0]	[1]	[2]	[3]	・・・・	[47]	[48]	[49]

以下同様にして処理が行われる

M[]についても同様の手順でいれかえが行われる。

print_num.c

```
1.  void print_num(char *x[], int m[], int n){ /* dt, move, nがそれぞれ配列*x,配列m,変数n,に
    対応している。*/
2.      int   i;
3.
4.      for(i=0; i<n; i++){/*関数Select_sortにより置き換えられたものを表示させる。*/
5.              puts(x[m[i]]);
6.    }
7.  }
```

解説・考察

配列の中に配列を書くことによって、配列の値を添字として使うことができる。そうすることで、線形リストの成分をソート時に入れ替えることをしなくても、添字の配列を順番にアクセスすることで、ソートした順番に画面に出力することができる。

Msg.c

```
int msg(){
        printf("#### Message from C #### By Yu SUGIMOTO¥n");
        return(0);

}
```

解説・考察

Printf の中身を表示させる。

Makefile

```
OBJECTS = cardinal.obj conv10.obj select_sort.obj print_num.obj msg.obj
cardinal.exe: $(OBJECTS)
        cl -o cardinal.exe $(OBJECTS)
cardinal.obj: cardinal.c
        cl -c cardinal.c
conv10.obj: conv10.c
        cl -c conv10.c
select_sort.obj: select_sort.c
```

```
	cl -c select_sort.c
print_num.obj: print_num.c
	cl -c print_num.c
msg.obj: msg.c
	cl -c msg.c
```

実行結果

```
-------- Input
x21
021
055
22
^Z
-------- Result
055
x21
22
021
#### Message from C #### By Yu SUGIMOTO
```

リスト構造プログラムの動作を解析しなさい。

```
1.  #include <stdio.h>
2.  #include <stdlib.h>
3.
4.  #define FALSE 0/*マクロ定義FALSEはを表す。*/
5.  #define TRUE   !FALSE/*マクロ定義　TRUEは以外を表す*/
6.
7.  typedef struct Node{/*typedefすると後に「struct」というのを省略可能。構造体の設定タグ名は
    Node*/
8.      int num;/*データ型int型変数名num*/
9.      struct Node *next_ptr;/*構造体の宣言,ポインタ変数next_ptrを宣言*/
10.
11. }node;/*nodeという変数名。これのおかげで構造体のタグ名の省略ができる*/
12.
13. node *start_ptr = NULL;/*struct Node *start_ptr=NULLということ*/
```

```
14.
15.  void ins(int idata){/*void型関数insの設定*/
16.          node *p = start_ptr;/*  ここではまだ、ポインタしかなく、実体は確保されていない。*/
17.          start_ptr = (node *)malloc(sizeof(node))/*nodeのサイズを測りその分だけメモリが帰ってくる
     */
18.          if (start_ptr == NULL) puts("Not enough memory!"), exit(0);/*start_ptrの値がNULLだった場
     合Not enough memory!と表示させてプログラムを正常終了させる。*/
19.          start_ptr->num = idata;/*構造体からnumを呼び出し値をidataに置き換える*/
20.          start_ptr->next_ptr = p;/*構造体からnext_ptrを呼び出し値をpに置き換える*/
21.
22.  }
23.
24.  int main(){
25.          int idata;
26.          node *p;
27.
28.          puts("Enter a sequence of integers:");
29.          while(scanf("%d", &idata) == TRUE) ins(idata);/*idataのアドレスがNULLだった場合、関数
     insにidataを引き渡す*/
30.
31.          puts("In reverse order:");
32.          for(p = start_ptr; p != NULL; p = p->next_ptr){
33.                  printf("%5d-", p->num);    /*  アドレスと値の出力へ更新しなさい*/
34.
35.          }
36.          puts("/end/");
37.          return(0);
38.  }
```

解説

7~10 行目：struct Node 型の構造体を作成、この構造体のメンバは、値を
格納する int 型の変数 num と struct Node 型をさすポインタ next_ptr となっている。
7 行目で typedef により struct Node 型に node という名前をつける。
10 行目：node 型（struct Node 型）を指すポインタ start_ptr の宣言をし
そのポインタに NULL を設定する
12 行目：main 関数の範囲、ソースでは void ins 関数の方が先にあるが main 関数の方が

先に処理される。
25,26 行目：int 型の変数 idate と node 型を指すポインタ p の宣言
28 行目：整数の入力を促す文を標準出力する。
29 行目：scanf 関数で idate に整数値を入力し、その値を引数にし、ins 関数にとぶ。
scanf 関数が読み込みを行わなかった場合（入力に文字を入れる、何も入れずに
Enter キーを押すなど）は、while 文の条件を満たしていないため、31 行目 に進む。
14~22 行目：ins 関数の範囲。数多くのポインタが動き回るリスト処理を行う。
（1 回目の呼び出しの場合）
15 行目：node 型のポインタ p の宣言。p は start_ptr を指し、start_ptr は最初 NULL を指しているため、つまり p は NULL を指す。
17 行目：malloc により node 型の大きさ分メモリを確保し（データ 1 とする）、start_ptr がそこの先頭アドレスを指す
18 行目：molloc が領域を確保できなかったとき、エラーメッセージを出力。滅多なことではエラーは起こらない
20,21 行目：start_ptr が指しているデータ 1 のメンバ num に idate の値を代入。メンバ next_ptr には p のアドレスを代入する。p は NULL を指しているため、つまり next_ptr は NULL を指す。
（ins 関数 2 回目の呼び出しの場合）
15 行目：node 型のポインタ p の宣言。p は start_ptr を指し、start_ptr はこの時点でデータ 1 を指しているため、つまり p はデータ 1 を指す。
17 行目：malloc により node 型の大きさ分メモリを確保し（データ 2 とする）、start_ptr がそこの先頭アドレスを指す
20,21 行目：start_ptr が指しているデータ 2 のメンバ num に idate の値を代入。メンバ next_ptr には p のアドレスを代入する。p はデータ 1 を指しているため、つまり next_ptr はデータ 1 を指す。
31~33 行目：リスト処理した結果のデータ 1,2,3…を出力していく。

感想
テストと重なりかなり疲れました。

参考資料
「c 実践プログラミング」
「新 The UNIX Super Text(上)」
「初心者のためのポイント学習Ｃ言語」http://www9.plala.or.jp/sgwr-t/index.html