プログラミング基礎

2023年度1Q　火曜日３,４時限(10:45~12:25)
金曜日1~４時限(8:50~12:25)

工学院　情報通信系

中山実,　渡辺義浩
伊藤泉,　小杉哲
TA: 小泊大輝,　千脇彰悟

5/19(金) 8:50〜12:25
「文字列処理とポインタ」
「ソート 1」

1. 文字列のライブラリ関数
2. ポインタの基礎
3. 文字列処理の例

4. ソーティング
5. 交換法と挿入法

文字列処理関数の参照

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int strlen2(char s[]);

int main(void) {

	char str[100];

	printf("Input a word¥n");
	scanf("%s", str);
	printf("The word is: %s, and length=%d ¥n", str, strlen2(str));
	printf("The word is: %s, and length=%d ¥n", str, strlen(str));


	return 0;
}

int strlen2(char s[])
{
	int i;

	i=0;
	while (s[i]!='¥0')
		++i;
	return i;
}
```

ライブラリ関数による出力

文字列コピー関数

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>


int main(void) {

        char str1[]="abcd";
        char str2[]="123";

        printf("%s¥n", str1);
        printf("%s %s¥n", strcpy(str1,str2), str1);


        return 0;
}
```

strcpy(str1, str2)
str2をstr1へコピー

‘1’,’2’,’3’,’¥0’

‘a’,’b’,’c’,’d’,’¥0’

‘1’,’2’,’3’,’¥0’,’¥0’

strcpyは文字型

strncpy(s,ct,n)　文字列ctのうち、最大n文字をsにコピーし、sを返す。
ctがn文字より少ないときは’¥0’をつめる。

文字列連結関数

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>


int main(void) {

        char str1[12]="abcd";
        char str2[]="123";

        printf("%s %s ¥n",strcat(str1,str2),str2);
        printf("%s %s ¥n",strcat(str1,str2),str2);


        return 0;
}
```

strcat(str1, str2)
str2をstr1に連結させる

‘a’,’b’,’c’,’d’,’¥0’
‘1’,’2’,’3’,’¥0’
‘a’,’b’,’c’,’d’,‘1’,’2’,’3’,’¥0’

ただし、str1の長さが不十分だと実行されない

出力例：
abcd123 123
abcd123123 123

strncat(s,ct,n)　文字列ctの最大n文字をsに連結し、最後に’¥0’をを付けてsを返す。

文字列比較関数

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>


int main(void) {

        char str1[12]="abcd";
        char str2[]="123";
        char str3[]="abCD"

        printf("%d %d¥n", strcmp(str1,str2),strcmp(str2,str1));
        printf("%d %d¥n", strncmp(str1,str3,2), strncmp(str1,str3,4);

        return 0;
}
```

strcmp(str1, str2)

文字列の比較
str1>str2 返り値＞0
str1<str2 返り値<0
str1=str2 返り値＝0

出力結果
48 -48
0 32

strncmp(cs,ct,n)　文字列csと文字列ctの最大n文字を比較し、cs<ctなら負値を、cs==ctなら0を、cs>ctなら正値を返す。

ASCII コード表

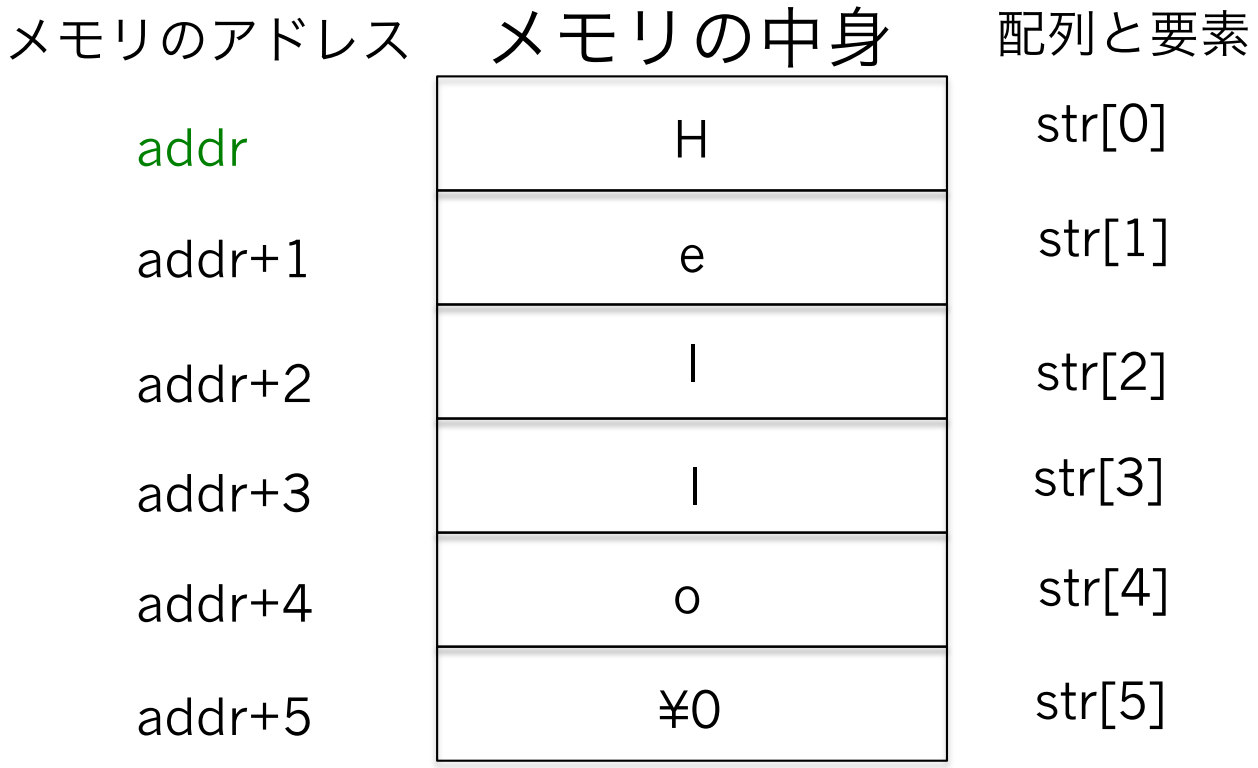
★制御文字 ★サロゲートペア ★合成文字 ★未定義

	US-ASCII - us-ascii															
	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+A	+B	+C	+D	+E	+F
0																
10																
20		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
30	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
40	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
50	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
60	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
70	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
128文字																

https://charset.uic.jp/show/us-ascii/

文字型変数の表現

char str[6]={'H','e','l','l','o','¥0'};



ポインタ：メモリの「アドレス」を保持する**変数**

charポインタ型

```
char str[6]={'H','e','l','l','o','¥0'};
char *sp ;

sp = str;
printf("%s¥n",sp);

do{printf("%c ",*sp);}while(*sp++!='¥0');
```

spは「char型ポインタ」

*はポインタで示される
　アドレスの中身

str[i]は、*(sp+i)を示す。

&はアドレスを参照

```
Hello
H e l l o
```

文字型
sp = str は sp = &str[0] と等価

ポインタに関する演算子：
＊: 間接演算子 *spのようにポインタ型変数が参照するアドレスの中身
&: アドレス演算子 &strのように変数のメモリでのアドレスを返す
ポインタの加算、減算が可能: sp++

アドレスとchar型の中身

```
char *sp, str[6]={'H','e','l','l','o',’¥0’};

for(i=0; i<6; i++){
        sp = &(str[i]);
        printf("%p, *sp=%c¥n", sp, *sp);
}
```

「ポインタが示すアドレス」
*付きで「ポインタで示される
　アドレスの中身」の印刷
%pはポインタの指定子

```
0x7ff7b72ca3ea,*sp=H
0x7ff7b72ca3eb,*sp=e
0x7ff7b72ca3ec,*sp=l
0x7ff7b72ca3ed,*sp=l
0x7ff7b72ca3ee,*sp=o
0x7ff7b72ca3ef,*sp=
```

アドレスと中身の結果

文字列処理でのポインタ利用(文字列長)

```
#include <stdlib.h>

int strlen2(char s[]);
int strlenp(char *ss);

int main(void) {
        char *sp, str[10]={'H','e','l','l','o','¥0'};

        sp=str;
        printf("strlen2=%d¥n", strlen2(str));
        printf("strlenp=%d¥n", strlenp(sp));
        return 0;
}

int strlenp(char *ss){
        char *p = ss;
        while(*p!='¥0') p++;
        return p-ss;
}
```

プロトタイプ宣言

spは「char型ポインタ」
sp = str は sp = &str と等価

文字列長の関数にポインタを渡す

ポインタ版と配列版の比較
結果：
strlen2=5
strlenp=5

```
int strlen2(char s[]){
        int i=0;
        while (s[i]!='¥0') ++i;
        return i;
}
```

5/19(金) 10:45〜12:15
「文字列処理とポインタ」
「ソート 1」

1. 文字列のライブラリ関数
2. ポインタの基礎
3. 文字列処理の例

4. ソーティング
5. 交換法と挿入法

ソーティング(整列)

- 数値などのデータを大小関係などによって、順序関係を満たすように並び替える操作。
- 例：
  - 45, 23, 30, 5, 17, 33
  - g, d, f, z, k, q

  ➔ 5, 17, 23, 30, 33, 45 (昇順ソート)

  ➔ 45, 33, 30, 23, 17, 5 (降順ソート)
  - g, d, f, z, k, q

  ➔ d, f, g, k, q, z (昇順ソート)

ソーティング

- 安定性
  - 数値が等しいデータについて、整列前の順序関係が整列後も保存される
    - 整列対象以外の属性を用いない

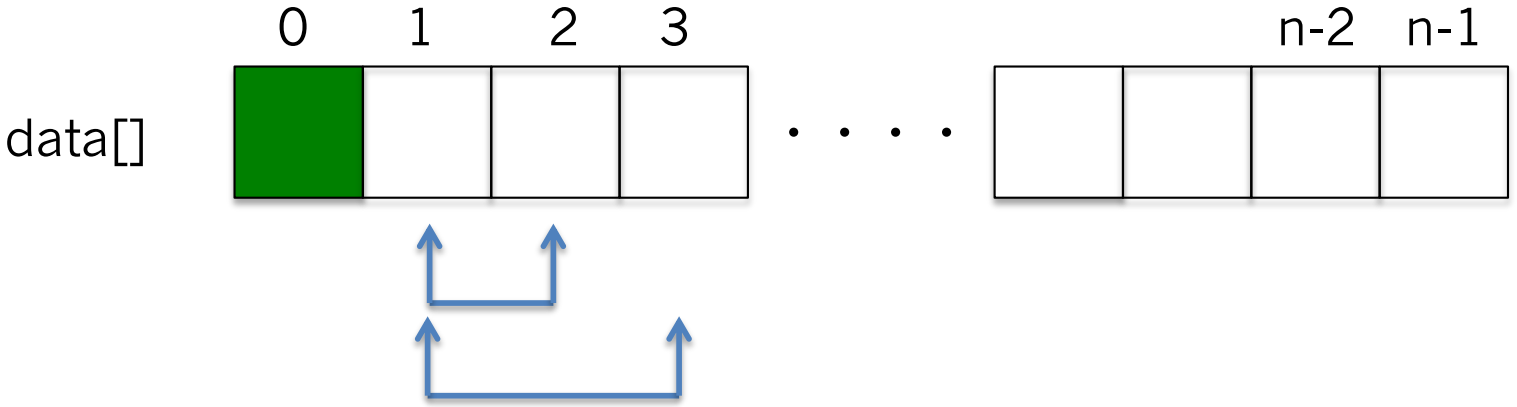
  例：添字記号は便宜上のもの

  2a, 1a, 2b, 1b, 2c, 3

  ➔ 1a, 1b, 2a, 2b, 2c, 3

- 内部ソートと外部ソート
  - 内部ソート：元データの格納領域で整列
  - 外部ソート：外部作業領域を必要

単純選択ソート



1. 最小データを、先頭データdata[0]と交換。先頭は最小値として確定。
2. 次のdata[1]を1~n-1の範囲で最小値と交換する。
3. i番目からn-2番目まで、それぞれの最小値と置換する。残るn-1番目は最大値となる。

```
#include <stdio.h>
#include <math.h>

float my_random(float lower, float upper);
void set_data(float a[], int n, int seed);
void print_data(float a[], int n);

int main(void) {
        float data[10], swap;
        int i,j, min, n=10;

        set_data(data, n, 0);
        print_data(data, n);

        for(i=0; i<n-1; i++){
                min = i;
                for(j=i+1; j<n; j++){
                        if(data[j]<data[min]){
                                min=j;
                        }
                }
                swap = data[i];
                data[i]=data[min];
                data[min]=swap;
        }
        print_data(data,n);

        return 0;
}
```

順番に最小値に置き換える。
未確定なデータ間で比較。

最小データと入れ替え

```
void set_data(float a[], int n, int seed){
        int i;
        srand(seed);
        for(i=0; i<n; i++){
                a[i]=my_random(0.0, 1000.0);
        }
}
```

関数rand()の初期化

```
float my_random(float lower, float upper){
        float r;
        r = (upper -lower)*rand()/(RAND_MAX)+lower;
        return r;
}
```

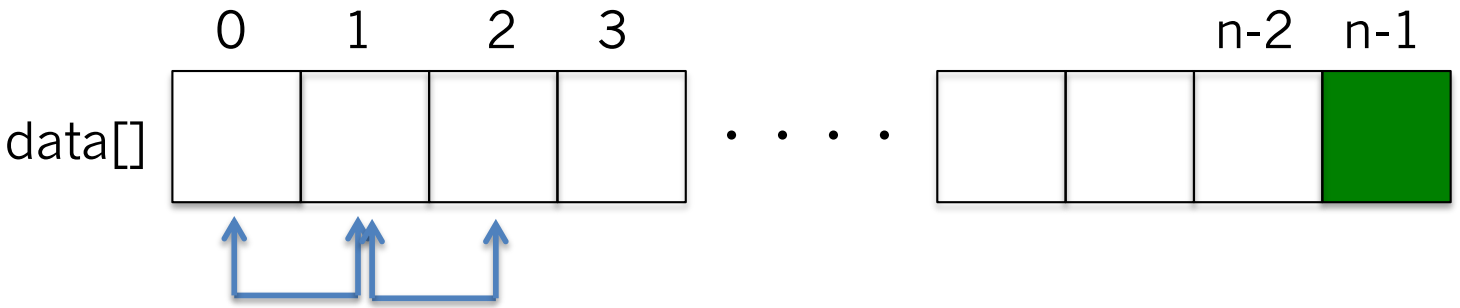
関数rand()を用いたデータ生成

```
void print_data(float a[], int n){
        int i;
        for (i=0; i<n; i++){
                printf("%8.2f ¥n",a[i]);
        }
        printf("Total %d data ¥n", n);
}
```

データを表示する関数

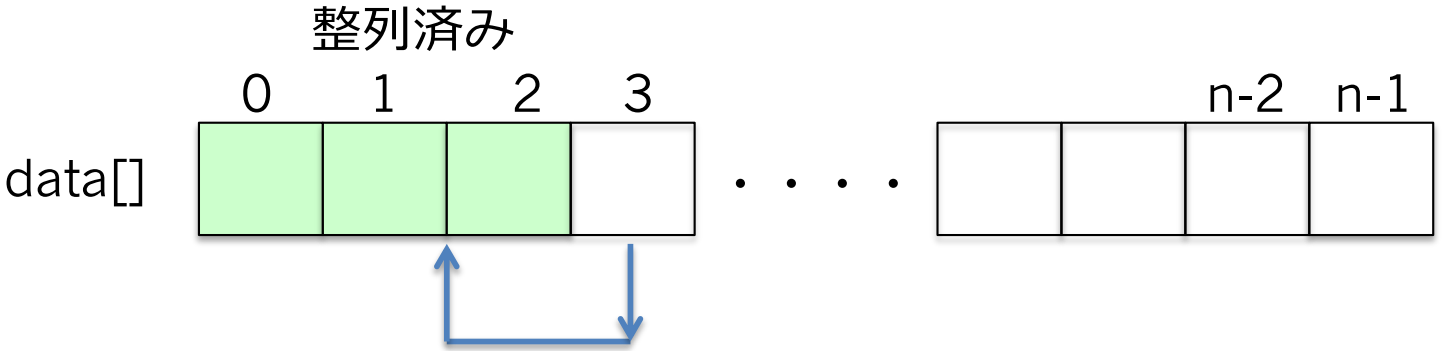
交換法によるソート

Bubble ソート



1. 末尾(n-1番目)まで比較して、順次、必要に応じて交換し、n-1番目が最大値とする。
2. 再度、0番目からn-2番目まで交換を繰り返し、その最大値をn-2番目とする。
3. 0番目と1番目の確認終了で、終了。

挿入法によるソート



1. 先頭２データを比較、順序化
2. 3番のデータを、整列済みの適切な位置へ挿入する。以降は順次移動させる。
3. n-1まで挿入、移動を繰り返す。