プログラミング基礎演習

第9回　ポインタ配列を用いたソート

担当：藤本　雄一郎（10-414室）
y_fuji@cc.tuat.ac.jp

1．前回のおさらい

ポインタのポインタ（イメージ）

メモリ初期状態
（何も定義されていない
＝Undefined）

アドレス	1	2	・・・	6	・・・	573	・・・	765	・・・	1024
中身	Undef.	Undef.	・・・	Undef.	・・・	Undef.	・・・	Undef.	・・・	Undef.

List =
(char**) malloc(sizeof(char*)*2);

char型変数へのポインタ
2個分を確保（メモリの2
～5番目，6～9番目）

1	2～5 List[0]	6～9 List[1]	・・・	573	・・・	765	・・・	1024
Undef.	Undef. （アドレス）	Undef. （アドレス）	・・・	Undef.	・・・	Undef.	・・・	Undef.

1．前回のおさらい

ポインタのポインタ（イメージ）

1	2～5 List[0]	6～9 List[1]	・・・	573	・・・	765	・・・	1024
Undef.	Undef. （アドレス）	Undef. （アドレス）	・・・	Undef.	・・・	Undef.	・・・	Undef.

List[0] =(char*) malloc(50);
List[1] =(char*) malloc(50);

573あるいは765番目から
50バイト分のメモリを確保し
その先頭アドレスを
List[0]やList[1]に格納

1	2～5 List[0]	6～9 List[1]	・・・	573 List[0][0]	・・・	622 List[0][49]	・・・	765 List[1][0]	・・・	814 List[1][49]	・・・	1024
Undef.	573 （アドレス）	765 （アドレス）	・・・	Undef. （char型の値）	・・・	Undef. （char型の値）	・・・	Undef. （char型の値）	・・・	Undef. （char型の値）	・・・	Undef.

復習: 文字列に関する関数

strlen(文字列)関数 : string length

文字列の文字数を返す関数

```
char  String[8] = "1234";
printf("Len=%d ¥n", strlen( String));
```

Len = 4

終端文字は
カウントされない

strcmp(文字列1, 文字列2)関数: string compare (comparison)

文字列同士を比較し、同じならば0、
文字列1が大きければ正(1など)、小さければ負(-1など)を返す（この大小は
文字コード順による）

```
char  StringA[8] = “ABD";
char  StringB[8] = “ABC";
if( strcmp(StringA,StringB) > 0)
      printf(“StringA > StringB ¥n” );
```

StringA > StringB

本日の内容

ポインタ配列を用いて
文字列のソートを行います

2. 文字列のソート

文字列のソートでは・・

```
void SelectionSort(char Data[DataSize][StringLength])
{
    for(SortPos = 0; SortPos < DataSize; SortPos++)
    {
        最小値を探す処理
        入れ替えを行う処理

    }
}
```

先頭から順番に最小値を探して入れ替えを行う

アルゴリズムの枠組みは変わらない

2. 文字列のソート

最小値を探す処理

```
MinID  = SortPos;

/* 先頭の次から探索をする */
for(CheckID = SortPos+1; CheckID < DataSize; CheckID++)
{
        if(strcmp(Data[MinID],Data[CheckID]) > 0)
                MinID  = CheckID;
}
/* 探索終了
```

大小比較部分に、関数を用いた

2. 文字列のソート

入れ替えを行う処理

```
/* 最小文字列をバックアップ */
char MinStr[StringLength];
strcpy(MinStr,Data[MinID]);

/* 先頭を最小文字列の場所にコピー */
strcpy(Data[MinID],Data[SortPos]);
/* 先頭は最小文字列に設定 */
strcpy(Data[SortPos],MinStr);
```

値の代入に関数を利用

2. 文字列のソート

文字列のソートでは・・

```
/* 最小文字列をバックアップ */
char Min
strcpy(M

/* 先頭
strcpy(D
/* 先頭
strcpy(D
```

文字列のコピーは必要だろうか？

- 文字列のコピーというのは、手間のかかる処理
- データ自体は場所が移動しているだけで内容に変化はない

内容を入れ替えるのではなく、場所の変化を表現したい

2. 文字列のソート

■データの動きを考える（strcpy関数）

Data[MinID]

Data[SortPos]

MinStr[StringLength]

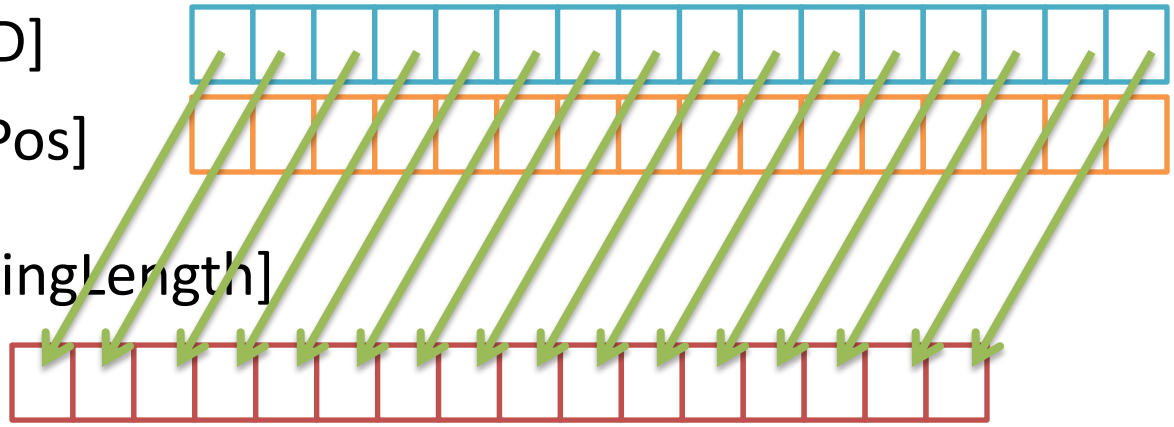
2. 文字列のソート

■データの動きを考える（strcpy関数）

Data[MinID]

Data[SortPos]

MinStr[StringLength]



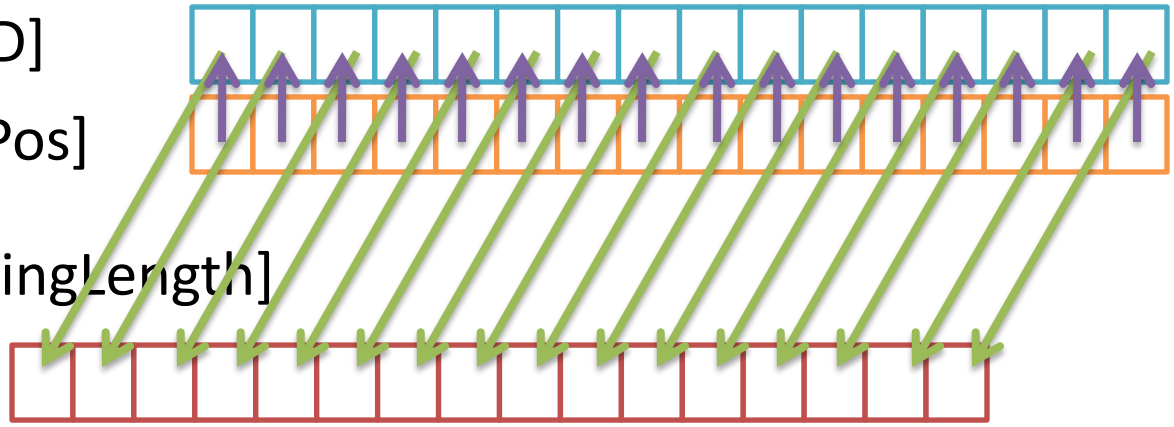
2. 文字列のソート

■データの動きを考える（strcpy関数）

Data[MinID]

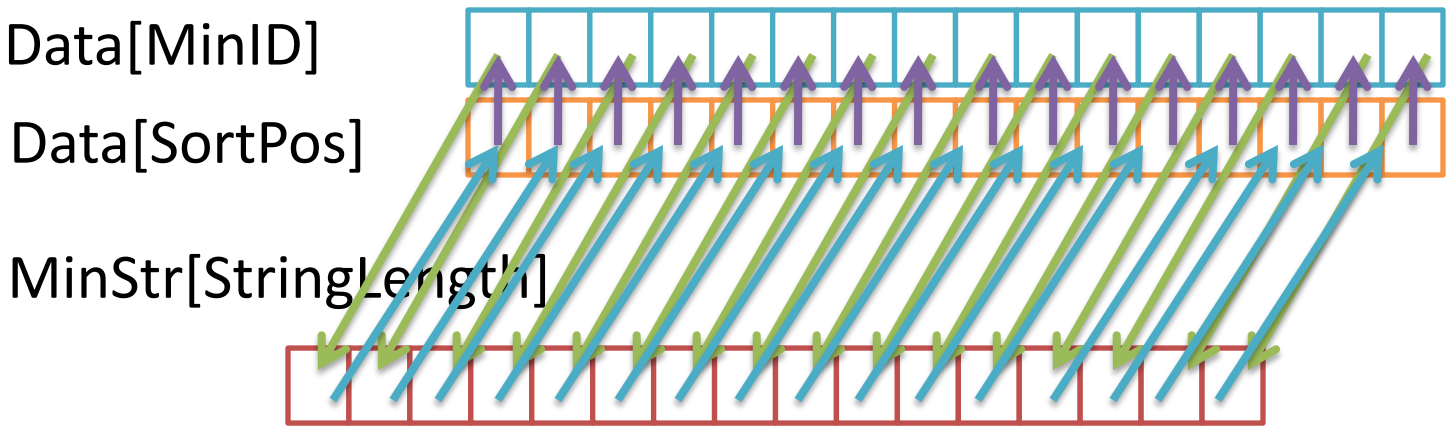
Data[SortPos]

MinStr[StringLength]



2. 文字列のソート

■データの動きを考える（strcpy関数）

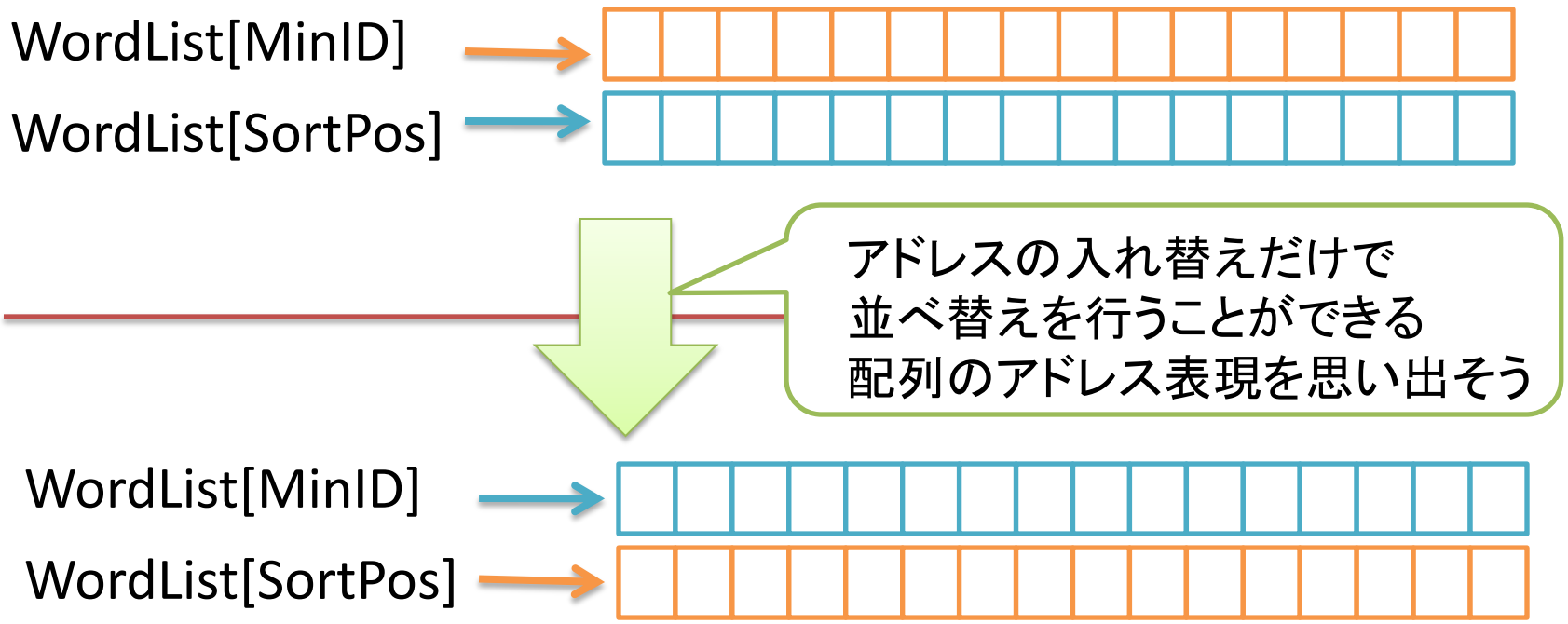


- データのコピーが多い（時間がかかる）
- 配列では他に方法がない

3. ポインタ配列を用いたソート

■動的に確保された配列の場合（場所の変化を表現できる）

```
char **WordList;
```



4. 本日の課題

■単語帳の作成

標準入力により，入力ファイル名と出力ファイル名をスペース区切りで指定する.
次に，単語数とその単語数分の文字列をファイルから読み込む.
入力ファイルの1行目には単語数が記載され，その次の行から入力する文字列が単語数分記載されている.
入力された文字列について，辞書順にソートし，ソートした文字列を新たなファイルに出力するプログラムを作成せよ.
また，その他の条件は次頁に示すものとする.

4. 本日の課題

■単語帳の作成

＜条件＞

- 入力ファイル・出力ファイルの名前は，標準入力で指定する
- 単語数と単語文字列の入出力はすべてファイルを利用する
  (リダイレクション不可)
- ファイルから読み込む各文字列の長さは不定（最大30文字）である
- 入力用バッファ（ char buf[30+1]等）を用いて良い
- ソート方法は問わない
- 単語文字列を記憶するためのメモリの確保にはmalloc関数を使い，
  必要以上に確保しないようにする
- 単語の入れ替えには、ポインタの差し替えを用いる

4. 本日の課題

入力ファイルの形式

入力ファイルは以下の通りである.

n
s_1
s_2
⋮
s_n

nは1以上の整数(実行に支障のない範囲で指定).
s_i($1 \leqq i \leqq n$)は長さ1以上30以下の文字列で, 'a'-'z'と'-'からなる.

4. 本日の課題

標準入力の形式

標準入力は以下の通りである.

f_{in} f_{out}

f_{in}は入力ファイル名を，f_{out}は出力ファイル名を指定する.
なお，f_{in}とf_{out}は，長さ1以上255以下の文字列で，'a'-'z'と'-'と'.'からなる.

4. 本日の課題

出力ファイルの形式

s_{j1}
s_{j2}
⋮
s_{jn}

以下の通りの出力ファイルを作成するようにすること.

入力ファイルから受け取った文字列s_1～s_nを辞書順にソートした新たな文字列s_{j1}～s_{jn}を，各行に1つずつ出力ファイルに書き出せ.
なお，s_{j1}からs_{jn}の各行の後には改行が入る.

4. 本日の課題

入力例・出力例

例.

入力ファイル: input-file.txt

```
3
abrty
y
abbu
```

標準入力:

```
input-file.txt output-file.txt
```

出力ファイル: output-file.txt

```
abbu
abrty
y
```

4. 本日の課題

■参考（ファイル入出力用コード）

- ファイルを開く （rは読込，wは書込モードで開く）

```
FILE *fp_input = fopen(“input_file.txt”, “r”);
FILE *fp_output = fopen(“output_file.txt”, “w”);
```

- ファイルから文字を読み込む

```
fscanf(fp_input, "%d", &n);
fscanf(fp_input, "%s", buf);
```

- ファイルに文字を書き込む

```
fprintf(fp_output, "%s¥n", wordlist[i]);
```

- ファイルを閉じる

```
fclose(fp_input);
fclose(fp_output);
```

4. 本日の課題

■プログラムの作成（第9回講義演習課題）

- ソースコードのファイル名は次の通りにする（すべて小文字）
  （EDENアカウント）_wordlist.c
- 提出用ディレクトリ：
  /home/y_fuji/prokiso/
- コマンドでコピーを行う場合：
  cp （EDENアカウント）_wordlist.c /home/y_fuji/prokiso/

- コピーする前にアクセス権を変更しておくこと

[sxx@xx~/prokiso] chmod 755 （EDENアカウント）_wordlist.c

- 提出期限： 12月12日（月） AM10:30（講義開始時）

5. 次回について

第10回（12月12日）
「構造体」