プログラミング演習
文字列操作(2)

阿萬裕久
aman@cs.ehime-u.ac.jp

例題1

- 2 つの文字列(それぞれ改行を含めて 64 文字未満)を読み込み，それらを 1 行に連結した文字列を作りなさい．
  ただし，連結の結果に改行文字は含めないものとする．

例題1:実行例

- 入力

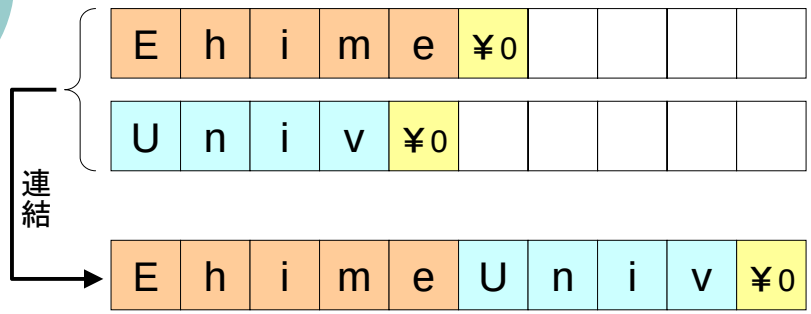
```
Ehime↵
Univ↵
```

- 出力

```
EhimeUniv
```


例題1:解説

- 文字列の**連結**を具体的に描くと



例題1：解説

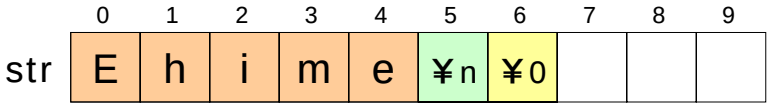
Web ページのコーディング例を参照

- まず，1つ目の文字列を読み込む

16行目 `fgets(str, 64, stdin);`

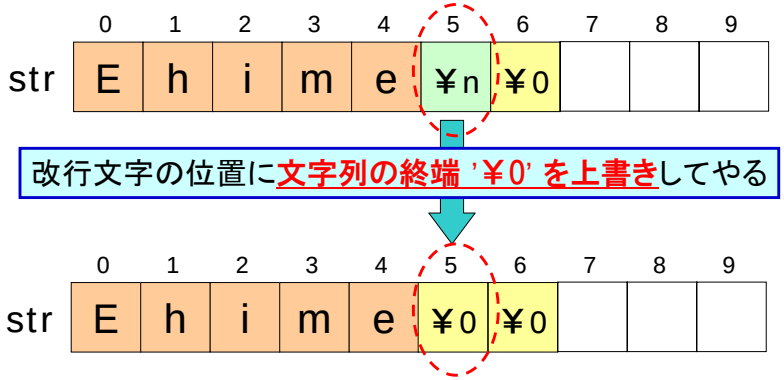
※問題では64文字未満なので

入力 Ehime ↵



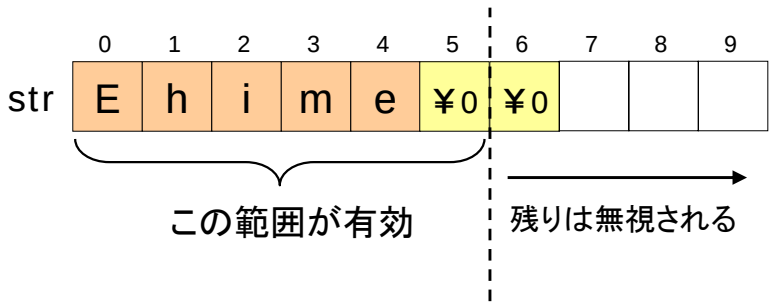
例題1：解説

- 連結の前に改行文字は消してしまいたい



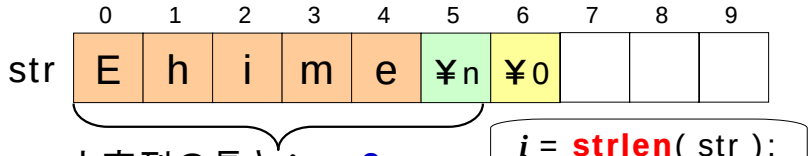
例題1：解説

- 文字列は**最初に '¥0' が登場するまで**有効なので，2つあっても大丈夫



例題1：解説

- 実際には文字列の長さを使って改行文字の位置を特定



文字列の長さ i ＝ **6**
（**strlen** 関数を使って調べる）

i = **strlen**(str);

改行文字は**後ろから2番目**：str[**5**]

$i-1$

例題1：解説

Web ページのコーディング例を参照

- つまり，文字列 str の長さが *i* の時，
  改行文字は str[*i*−1] にある

19,20行目

```
i = strlen(str);
str[i-1] = '¥0';
```

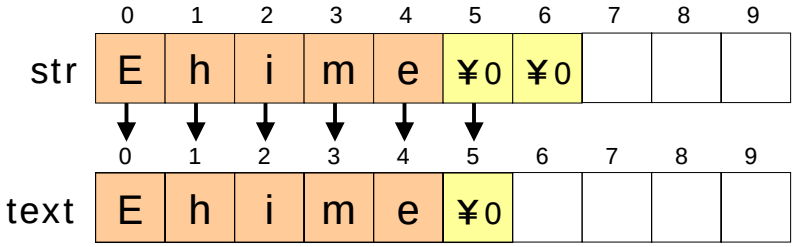

例題1：解説

Web ページのコーディング例を参照

- 次に，2つ目の入力に備えて，str の中身を別の配列へコピーしておく

23行目

```
strcpy(text, str);
```



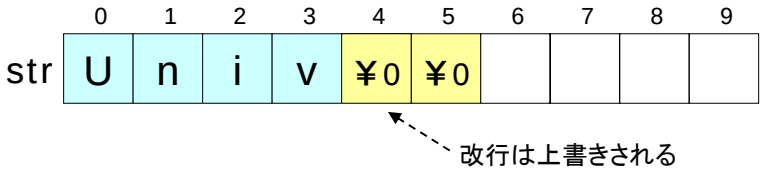
例題1：解説

Web ページのコーディング例を参照

- 再び，2つ目の文字列を読み込み，改行文字を消しておく

27～29行目

```
fgets(str, 64, stdin);
i = strlen(str);
str[i-1] = '¥0';
```

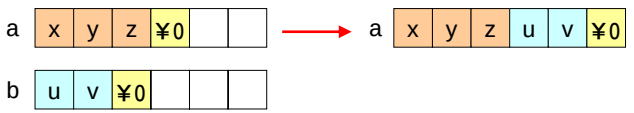


例題1：解説

- 文字列の連結
  1文字ずつコピーしていっても良いが，専用の関数が用意されている

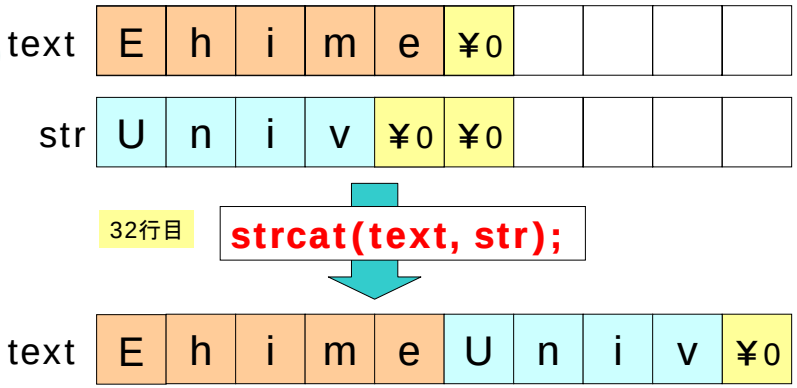
```
strcat( 連結先, 連結元 );
```

（例）strcat(a, b);



例題1：解説

Web ページのコーディング例を参照



例題2

- 64 文字未満の文字列（途中に空白を含む．最後に改行を含む．）を読み込み，空白を区切り文字として入力文字列を複数の部分文字列へ分解しなさい．
  ただし，空白文字は部分文字列に含めないものとする．

例題2：実行例

- 入力

```
This is a test data
```

- 出力

```
This
is
a
test
data
```


例題2：解説

- 基本方針

```
This is a test data
```

1文字ずつ見ていき

①空白以外なら用意した文字配列に追加していく

空 → T → Th → Thi → This

②空白に遭遇した時点で表示し，再び空にする

This → 空

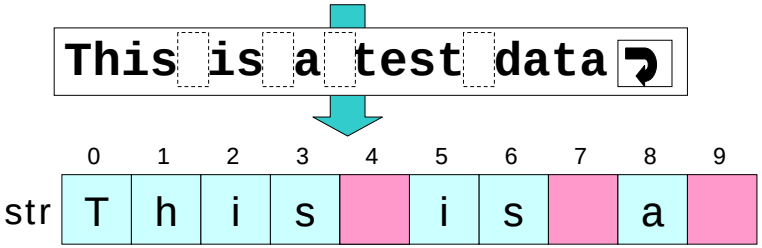
例題2：解説

Web ページのコーディング例を参照

- まずは文字列を読み込む

16行目

```
fgets(str, 64, stdin);
```



例題2：解説

Web ページのコーディング例を参照

- 1文字ずつ見ていきながら空白かどうかチェックする

27～36行目

```
for ( i = 0; i < strlen(str); i++ ){
    if ( str[i] != ' ' ){
        空白でない場合，別の文字配列へ追加
    }                          ここでは word とする
    else{
        空白の場合，文字配列を表示してリセット
    }
}
```


例題2：解説

Web ページのコーディング例を参照

- 空白でない場合：word の後ろに追加

28～31行目

```
if ( str[i] != ' ' ){
    buf[0] = str[i];
    strcat(word, buf);
}
```

できれば strcat(word, **str[i]**); と書きたい
しかし，str[i] は**文字であって文字列でない**！

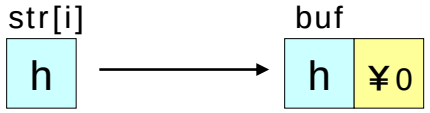
例題2：解説

Web ページのコーディング例を参照

- 1文字でも，いったん文字列（文字配列）へ変換しないと strcat は使えない
- そこで buf という配列を経由させる

```
buf[0] = str[i];
```

```
buf[1] = '¥0';
```
← これはあらかじめやっておく



例題2：解説

- 空白の場合：**word を出力してリセット**

32～35行目

```
else{
    printf("[%s]¥n", word);
    strcpy(word, "");
}
```

空文字列で上書き

※文字列の表示で [ ] を付けているが，
出力に空白が含まれていないことを確認しやすいようにしているだけで他意は無い

例題3

- 複数行の英文（1 行は改行も含めて 256 文字未満）を読み込み，その中で "software" という単語の登場回数を答えなさい．

例題3：解説

- 基本方針
  ①読み込んだ文字列を単語に分解し，
  ②その単語が “software” に等しいかどうかチェックする

※①に関しては例題2と同じ

例題3：解説

Web ページのコーディング例を参照

- 複数行の読み込みを行うため，fgets を繰り返し実行することになる

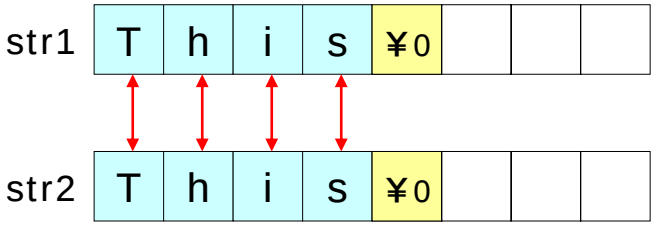
19～35行目

```
while ( fgets(str, 256, stdin) != NULL ){
    .......
}
```

最終行まで繰り返し読み込む：
入力が終了すると fgets の戻り値が NULL になるよう設計されている

例題3：解説

- 2つの文字列の比較
  原理的には1文字ずつ一致するか見ていく



例題3：解説

- しかし，文字列の比較はワンパターンなプログラムになるので，標準関数が存在する

strcmp(文字列1, 文字列2);

strcmp 関数の戻り値が重要
- 2つが同じなら，strcmp の戻り値は 0
- str1 の方が辞書式順序で先ならば負の数
- str1 の方が辞書式順序で後ならば正の数

※英和辞書に載せるとした場合の登場順

例題3：解説

Web ページのコーディング例を参照

- したがって，文字列 word の内容が
  “software” の時だけカウントするには

29～31行目

```
if ( strcmp(word,"software") == 0 ){
    count++;
}
```


なぜ strcmp を使うのか？

- なぜ 2 つの文字列 str1 と str2 の比較は
  if (**str1 == str2**){ …
  ではダメなのか？

直感的な説明

これは配列どうしの比較になるから．
例えば，int 型配列 a と b があったとき，
2つの配列の内容がすべて同じかどうか
を調べるのに if (a == b) では無理がある

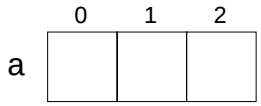
より正確な説明

ただし，書き換えは不可なので，厳密には定数

- 実は，**配列の名前**は**ポインタ変数**
- よく，配列の宣言

  int a[3];

  に対して次のイメージ図が使われる

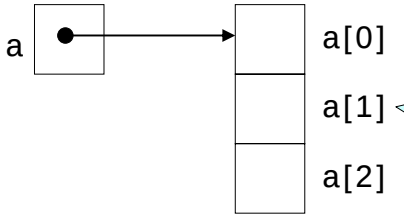


これは配列を直感的にとらえるには有効だが，
残念ながら正確ではない

より正確な説明

ただし，a の内容の書き換えは認められない

- 本当は **a という名前のポインタ変数**が作られ，その参照先に int 型変数が連続して3つ存在している

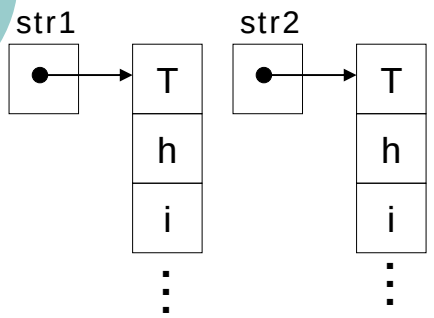


a の参照先から1マスだけずらした位置にあるので a[1] と書かれる

※配列の番号が 0 から始まっている理由でもある

より正確な説明

- 文字列も同じく配列なので



if (str1 == str2) というのは，2つの番地が同じかどうかという意味になってしまう！