

# 情報処理実習

## 第10回: 文字列

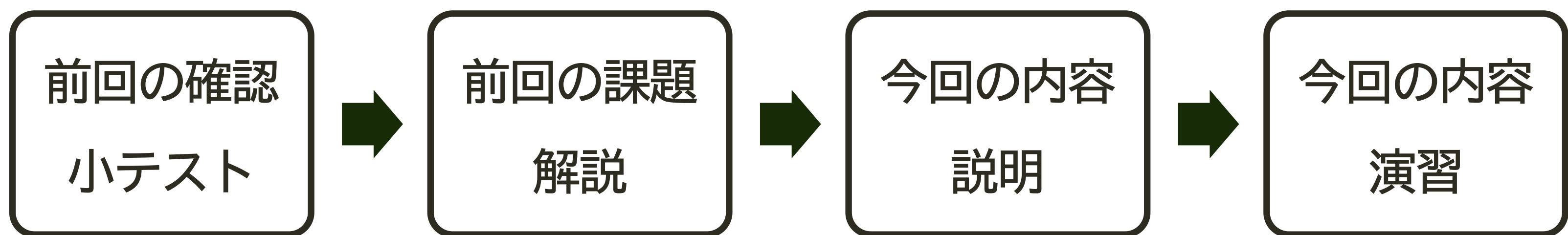
| 担当: 佐藤

2018年11月26日(月)

# アナウンス

- 授業前にやること
  - ・計算機にログオン
  - ・CoursePowerにログイン
- 注意事項
  - ・教室内飲食禁止
  - ・原則として、授業中ではなく休み時間にトイレにいきましょう

# 授業の進め方



# 基本事項の確認小テスト

制限時間: 5分

- ① CoursePowerにログインして、すぐに解答が始められるよう準備する
- ② 開始の合図があるまで、解答を始めずに待機する

注意

「閉じる」ボタンは決して押さないこと！

# 文字列の概要



2018年11月26日(月)

# 文字列

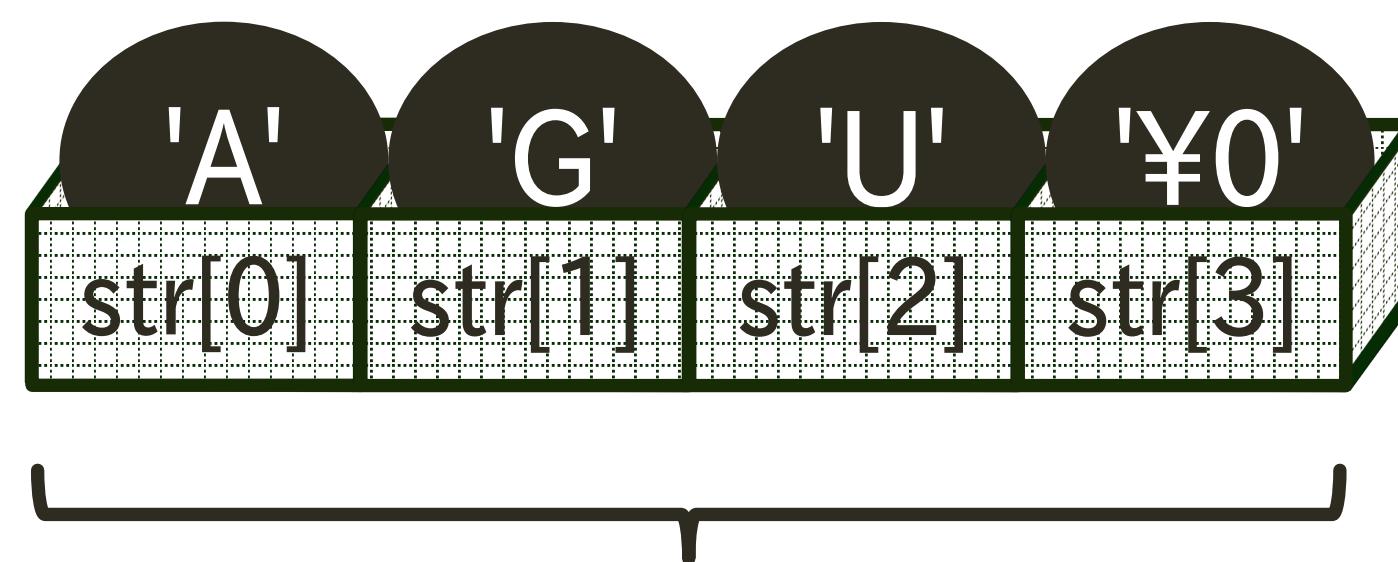
- C言語では、最後の要素が'¥0'(終端文字、ナル文字、ヌル文字)の文字の配列(文字配列)を**文字列**という

例

ヌル文字  
分も考慮!

```
char str[4];  
  
str[0] = 'A';  
str[1] = 'G';  
str[2] = 'U';  
str[3] = '¥0';
```

文字列ではない!



文字列

# 文字配列の初期化(1)

- "(二重引用符, ダブルクォーテーション)で囲まれた文字の並びを 文字列定数(文字列リテラル)という
  - ・文字列定数は, 末尾にヌル文字が自動で付加される
- 文字列定数を用いて文字配列を初期化することができる

**文法**

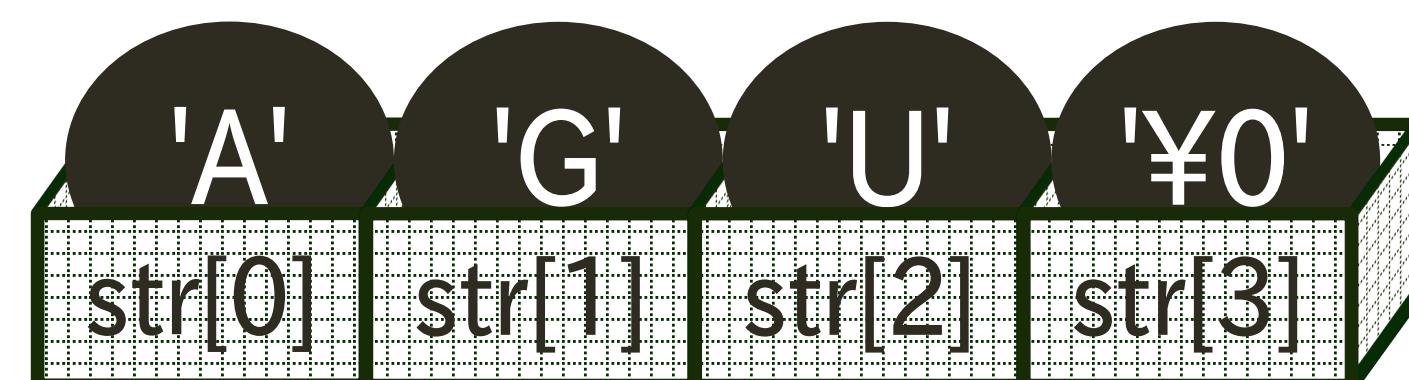
省略可

```
char 文字配列名[要素数] = 文字列定数;
```

**例**

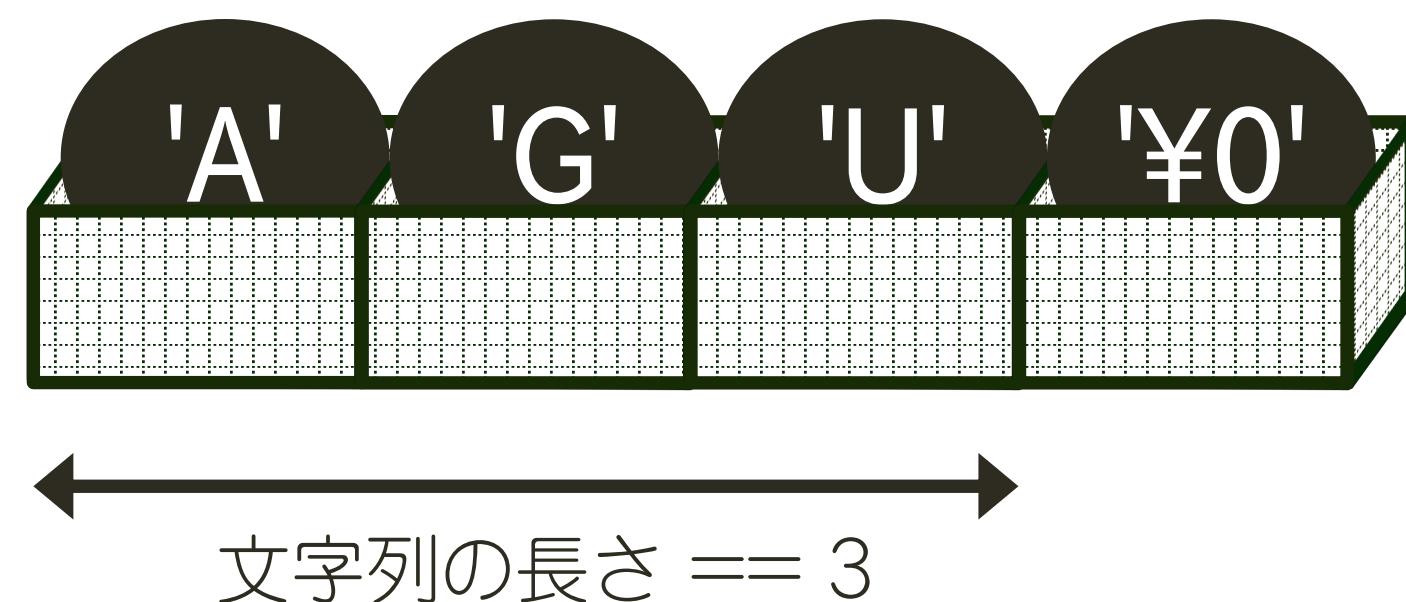
```
char str[4] = "AGU";
```

末尾のヌル文字は書かなくてよい!



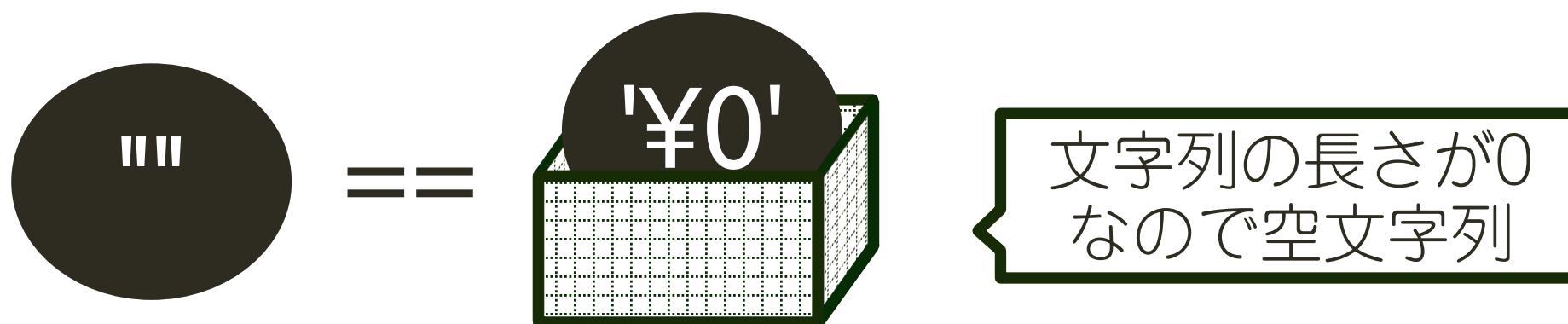
# 文字列の長さ

- 文字列の先頭の文字からヌル文字まで(ヌル文字を除く)の文字の個数を**文字列の長さ**という
- **文字列の要素数 == 文字列の長さ + 1**
- 文字配列を宣言する際、配列の要素数には文字列の長さではなく文字列の要素数を設定すること

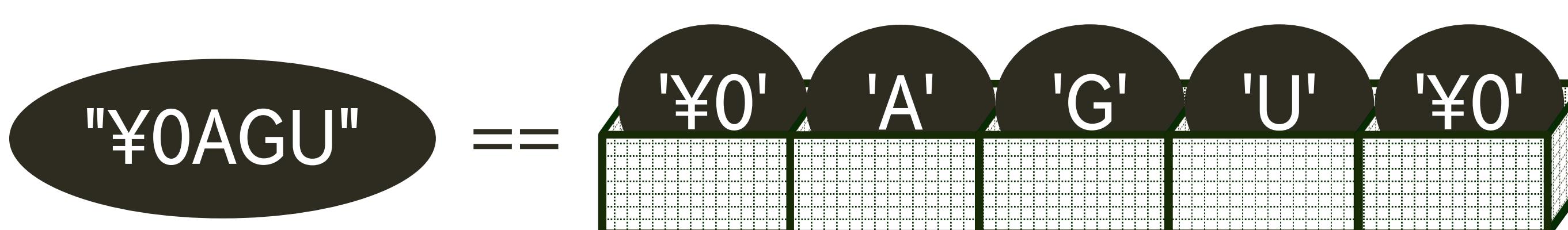


# 空文字列

- 文字列の長さが0の文字列を**空(から、くう)文字列**という



先頭がヌル文字 → 文字列の要素数に関係なく空文字列



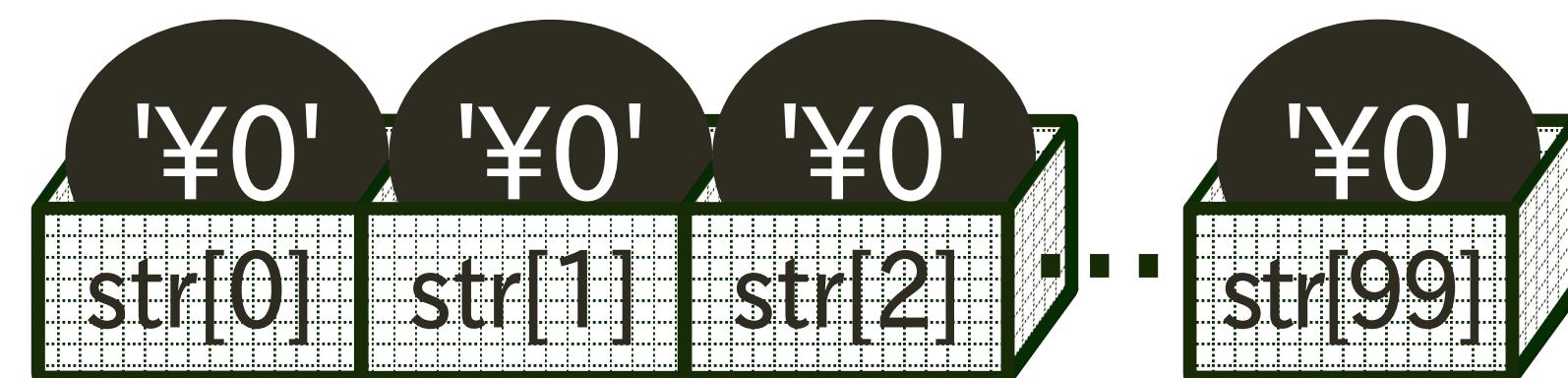
# 文字配列の初期化(2)

文字配列を空文字列で初期化

- 初期値として設定する文字列が未定 → 文字配列は空文字列で初期化しておくこと!
  - ・int型の変数を0で初期化しておくのと同様、ゴミの値に起因する不具合を防止するため

例

```
char str[100] = "";
```



すべての要素に'¥0'が設定される

# 文字列の入出力



2018年11月26日(月)

# 文字列の入力(1)

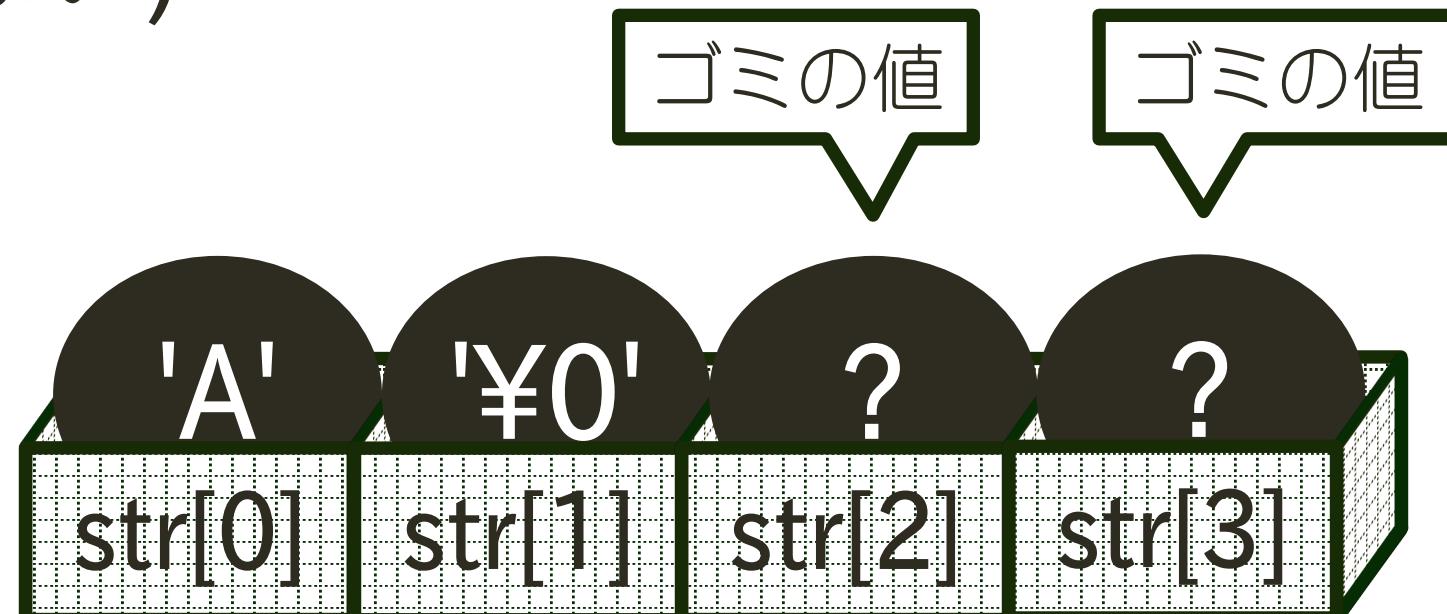
## ● scanf()を使った文字列の入力



・空白類文字(※)に出会うと読み込みを終了し、**最後にヌル文字('¥0')を付加する**(空白類文字は読み込まれない)

### 例

```
char str[4];  
scanf("%s", str); /* 「A_U」を入力 */
```



説明用に敢えてstrを""で初期化していない。ゴミの値を要素に持つ文字列にならないよう「char str[4] = "";」と書くべき

文字配列は空文字列で初期化する癖をつけよう！

※空白文字(' '\_'), 改行文字('¥n'), タブ文字('¥t')などの総称

# 文字列の入力(2)

- **gets()**を使った文字列の入力

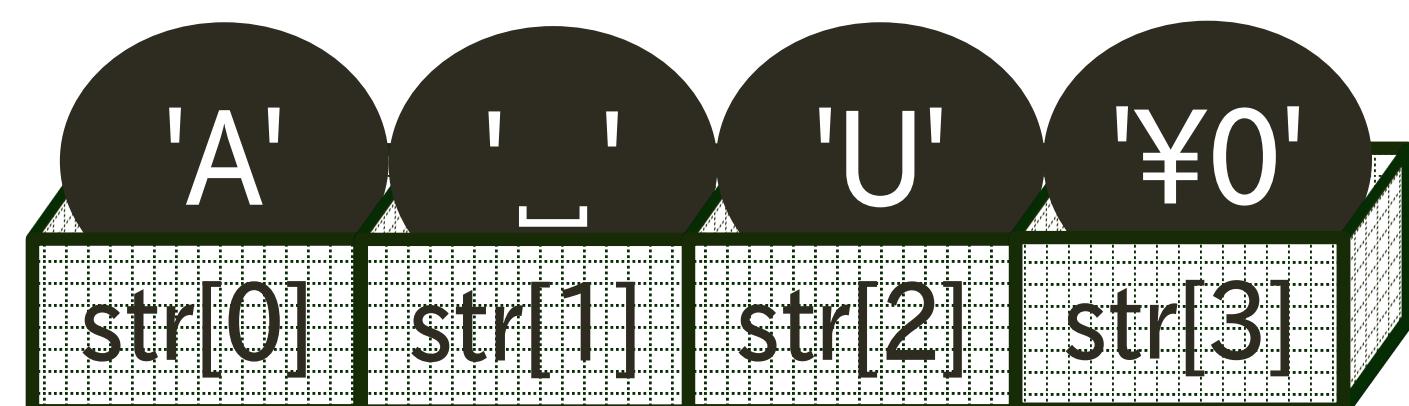
**文法** &は不要!

```
gets(文字配列名);
```

- ・改行文字('¥n')に出会うと読み込みを終了し、最後にヌル文字('¥0')を付加する('¥n'は読み込まれない)
- ・**空白文字(' ')を読み込む!**

**例**

```
char str[4];  
gets(str); /* 「A_U」を入力 */
```



# 文字列の入力(3)

- 入力終了を表す文字コード^Zが入力された場合のgets()の戻り値は  
**NULL(ナル, ヌル)**
  - ・コマンドプロンプトで^Zを入力するにはCtrl+Zを押す

## 例

```
char str[4] = "";

if (gets(str) != NULL) {
    printf("Yes¥n");
} else {
    printf("No¥n");
}
```

## 実行結果

AGU↙  
Yes

## 実行結果

^Z↙  
No

# 文字列の出力(1)

- printf()を使った文字列の出力

## 文法

文字配列名または文字列定数

```
printf("%s", 文字列);
```

- ・文字列の先頭の文字からヌル文字('¥0')の1つ前の文字まで出力

## 例

```
char str[4] = "AGU";  
  
printf("%s¥n", str);  
printf("%s¥n", "AGU"); /* printf("AGU¥n"); */
```

## 実行結果

```
AGU  
AGU
```

# 文字列の出力(2)

- `puts()`を使った文字列の出力

## 文法

```
puts(文字列);
```

- ・文字列の先頭の文字からヌル文字('¥0')の1つ前の文字まで出力し、**最後に改行文字('¥n')**を付加する

## 例

```
char str[4] = "AGU";  
  
puts(str);  
puts("AGU");
```

## 実行結果

```
AGU  
AGU
```

printf()と異なり、'¥n'を書かなくても勝手に改行される

# 文字列入力の詳細

- エンターキー(リターンキー)を押すと2つの効果がある

① 改行文字('¥n')の入力

② 入力内容の確定

・②だけでなく①の効果もあることに注意!

## 例

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

```
int main() {
    char str1[6] = "", str2[6] = "";

    gets(str1);           /* "A G U"を入力 */
    puts(str1);           /* "A G U"を出力 */
    scanf("%s", str2);    /* "A G U"を入力 */
    printf("%s¥n", str2); /* "A"を出力 */

    return 0;
}
```

gets()は'A'→' '→'G'→' '→'U'と先頭から順に文字をstr1に代入し、最後の文字である'¥n'に出会うと'¥n'の代わりに'¥0'を代入して読み込みを終える。したがって、str1には「A' ' 'G' ' 'U' '¥0' == "AGU"」という文字列が設定される

gets()とscanf()の読み込み対象は「A' ' 'G' ' 'U' '¥n'」という文字配列(末尾が'¥0'ではないため、この時点では文字列ではない)

scanf()もgets()同様、先頭から順に文字を(str2に)代入していくが、こちらは空白文字(' ')を読み込めないため'A'の次に' 'に出会うと読み込みを終え、「 'の代わりに'¥0'を代入する。したがって、str2には、「A' '¥0' == "A"」という文字列が設定される

# 文字列操作関数

2018年11月26日(月)

# string.h

- 標準ヘッダファイルのひとつ `string.h` をインクルードしておくことにより、あらかじめ用意された便利な **文字列操作関数** が使えるようになる
  - `#include <string.h>`
- 代表的な文字列操作関数は次の通り

関数	関数の機能
<code>strcpy()</code> (ストリングコピー)	文字列の代入
<code>strcat()</code> (ストリングキャット)	文字列の連結
<code>strcmp()</code> (ストリングコンプ)	文字列の比較
<code>strlen()</code> (ストリングレン)	文字列の長さ

catはconcatenation(連結)の略

# strcpy()

- 文字列は配列なので代入演算子(=)を使って文字配列に代入することはできない。文字列の代入はstrcpy()で行う
  - ・(文字)配列の初期化の文法における「=」は代入演算子ではない！初期化の文法に「=」記号が使われているだけ
- コピー先の文字配列の要素数  $>=$  コピー元の文字列の要素数

## 文法

```
strcpy(コピー先の文字配列名, コピー元の文字列);
```

## 例

コピー元の文字  
列の要素数以上

```
char str1[4] = "", str2[] = "AGU";  
  
strcpy(str1, str2);  
puts(str1);
```

## 実行結果

```
AGU
```

# strcat()

- 連結先の文字配列の**最後のヌル文字の位置から**連結元の文字列を連結(ヌル文字は連結元の文字列の先頭の文字で上書きされる)
- 文字配列の要素数  $>=$  連結後の文字列の要素数

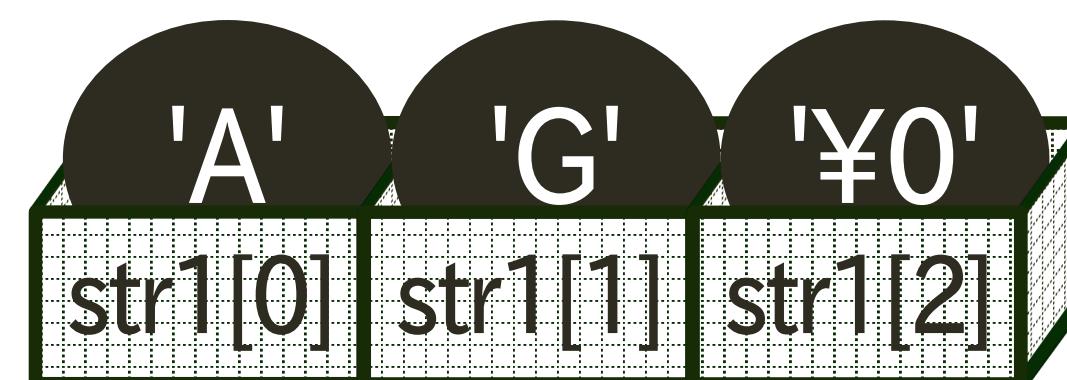
## 文法

```
strcat(連結先の文字配列名, 連結元の文字列);
```

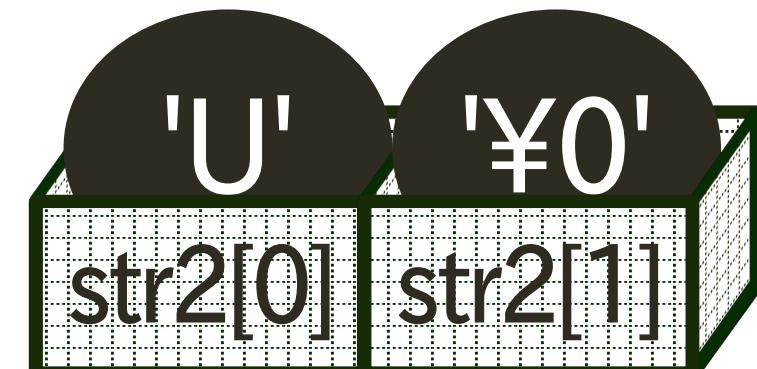
### 例

連結後の  
文字列の要素数

```
char str1[4] = "AG", str2[] = "U";  
  
strcat(str1, str2);  
puts(str1);
```



↑ 上書き



# strcmp()

- 2つの文字列が同じならば、戻り値として**0**が返される
  - ・if文の条件式で用いられる

## 文法

```
strcmp(文字列1, 文字列2);
```

## 例

```
char str1[4] = "AGU", str2[] = "AGU";  
  
if(strcmp(str1, str2) == 0) {  
    puts("str1とstr2は同じ文字列");  
} else {  
    puts("str1とstr2は違う文字列");  
}
```

## 実行結果

```
str1とstr2は同じ文字列
```

# strlen()

- 戻り値として文字列の長さを返す(文字列の要素数ではない!)

## 文法

```
strlen(文字列);
```

## 例

```
char str1[] = "AGU", str2[] = "", str3[] = "¥0AGU";  
  
printf("%d¥n", strlen(str1));  
printf("%d¥n", strlen(str2));  
printf("%d¥n", strlen(str3));
```

## 実行結果

```
3  
0  
0
```

# 課題



2018年11月26日(月)

# レポートの作成

① レポートの冒頭に以下を適切なレイアウトで書く

- 「情報処理実習第10回課題レポート」というタイトル
- 学生番号
- 氏名

② 課題ごとに以下を載せる

- 作成したプログラムのソースコード
- 作成したプログラムの実行結果を示すコマンドプロンプトのスクリーンショット

③ 完成したレポートをCoursePower上で提出する



提出〆切: 18:30