C言語の基礎5

文字列

- C言語での文字列の扱いは、他言語とは異なり、慣れが必要。
- 文字列を扱うための専用の型は存在しない。
    - 文字を扱う `char` を利用し、配列として文字列を扱う。
- 文字列は、終端文字（実際は `\0` だが、以降 `NULL` と表記）で終わる0文字以上の文字の集まり

【str1.c】

```c
#include <stdio.h>

int main(int argc, char const *argv[]) {
    char ch = 'a';    // 文字はシングルクォート
    char str[] = "a"; // 文字列はダブルクォート

    printf("文字：%c\n", ch);
    printf("文字列：%s\n", str);

    return 0;
}
```

【str2.c】

```c
#include <stdio.h>

int main(int argc, char const *argv[]) {
    char str[10] = "abcde";

    for (int i = 0; i < 10; i++) {
        printf("文字：%c 文字コード：0x%02x\n", str[i], str[i]);
    }

    return 0;
}
```

確認

- 以下のプログラムの場合、要素数はいくつか？

【str3.c】

```c
#include <stdio.h>

int main(int argc, char const *argv[]) {
    char str1[] = "abcde";
    char str2[][7] = {"abcde", "012", "z1"};

    printf("sizeof str1:%ld\n", sizeof(str1));
    printf("sizeof str2:%ld\n", sizeof(str2));

    return 0;
}
```

【str1】

	[0]	[1]	[2]	[3]	[4]	[5]
str1	'a'	'b'	'c'	'd'	'e'	NULL

【str2】

	[0]	[1]	[2]	[3]	[4]	[5]	[6]
str2[0]	'a'	'b'	'c'	'd'	'e'	NULL	NULL
str2[1]	'0'	'1'	'2'	NULL	NULL	NULL	NULL
str2[2]	'z'	'1'	NULL	NULL	NULL	NULL	NULL

printf での%s

- 文字列を出力する場合、`printf` では、`%s` を使用する。
    - これは、どのように動作するのだろうか？

【printf1.c】

```c
#include <stdio.h>

int main(int argc, char const *argv[]) {
    char str[] = "Hello World";
```

```
    // 1文字ずつ出力
    for (int i = 0; i < sizeof(str); i++) {
        printf("str[%02d] : (0x%02x) %c\n", i, str[i], str[i]);
    }
    // 文字列として出力
    printf("str : %s\n", str);

    return 0;
}
```

【printf2.c】

```
#include <stdio.h>

int main(int argc, char const *argv[]) {
    char str[] = "Hello World";

    // 1文字ずつ出力
    for (int i = 0; i < sizeof(str); i++) {
        printf("str[%02d] : (0x%02x) %c\n", i, str[i], str[i]);
    }
    // 文字列として出力
    printf("str : %s\n", str);

    // データを変更
    str[4] = '\0';

    // 1文字ずつ出力
    for (int i = 0; i < sizeof(str); i++) {
        printf("str[%02d] : (0x%02x) %c\n", i, str[i], str[i]);
    }
    // 文字列として出力
    printf("str : %s\n", str);

    return 0;
}
```

文字列の操作

- 文字列とは、文字配列の要素が `\0` （NULL）であるところまでのデータを意味する！

文字列の代入

- 文字列を他の文字配列に代入するにはどうすればよいだろうか？

配列の直接代入は出来ない

```
    char str1[] = "Hello";
    char result[];  // サイズが指定されていない
    result = str1;  // このような操作は出来ない
```

【set.c】（練習）

- for / while で代入する。

```
#include <stdio.h>

int main(int argc, char const *argv[]) {
    char str1[] = "Hello";
    char result[100];  // 結果を入れる配列（誤字修正：reslut → result）
    int  j = 0;        // result[j]の添字

    // str1 を ループで1個ずつ代入
    // 以下作成




    printf("result : %s\n", result);  // 誤字修正：reslut → result

    return 0;
}
```

【実行結果】

```
$ ./a.out
result : Hello
```

文字列の結合

- 文字列1と文字列2を結合するプログラムを作成してみよう。

```
    char str1[] = "Hello";
    char str2[] = "World";
  char result[];
  result = str1 + str2; // このような操作は出来ない
```

【concat.c】（練習）

```c
#include <stdio.h>

int main(int argc, char const *argv[]) {
    char str1[] = "Hello";
    char str2[] = "World";
    char concat[100];  // 結果を入れる配列
    int  j = 0;        // concat[j]の添字

    // str1 を concat にコピー



    // str2 を concat にコピー



    // 終端記号を付加
    concat[j] = '\0';
    printf("concat : %s\n", concat);

    return 0;
}
```

【実行結果】

```
$ ./a.out
concat : HelloWorld
```

文字列の比較

- 文字列1と文字列2を比較して、同じ文字列かを判定するプログラムを作成してみよう

	str1	str2	結果
パターン1	ABC	ABC	一致
パターン2	ABCD	ABC	不一致
パターン3	ABC	ABCDE	不一致

【compare.c】（練習）

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

int main(int argc, char const *argv[]) {
    char str1[] = "ABC";
    char str2[] = "ABCD";
    bool match = true;  // bool型を使用

    // 比較する処理を書いてみよう




    if (match) {
        printf("文字列は一致しています\n");
    } else {
        printf("文字列は一致していません\n");
    }

    return 0;
}
```

関数の受け渡し

- 関数へ文字列を渡すにはどう書けばよいか？

【str_func1.c】

```
#include <stdio.h>

void print(char[]);  // 関数プロトタイプ

int main(int argc, char const *argv[]) {
    char hello[] = "Hello World";

    print(hello);    // 配列名のみ記述する[]は書かない

    return 0;
}

void print(char str[]) {
    printf("%s\n", str);
}
```

練習

以下のプログラムを作成してください。ただしライブラリなどは使用しないものとします。

- 「Hello, World!」という文字列を逆順に出力するプログラムを作成してください。（prog1.c）
    - reverse関数を作成し、利用するようにしてください。
    - reverse関数内で、出力することにします。

- 大文字を小文字に変換するプログラムを作成してください。（prog2.c）
    - 文字を受け取り、小文字に変換して返すlower_c関数を作成する。
    - main内で出力することにする。
      【実行結果】

      ```
      $ ./a.out
      hello, world!
      ```

      【初期値】

      ```
      char str[] = "HELLO, World!";
      ```

- 大文字を小文字に変換するプログラムを作成してください。（prog3.c）
    - 文字列を受け取り、小文字に変換して出力するlower_str関数を作成する。
    - lower_str内で出力することにする。
    - 実行結果はprog2.cと同じ結果。

関数内での文字列の変更

- 渡された文字列（配列）の変更を行った場合、どうなるだろうか？

- 以下のサンプルをトレースし、どのような結果になるかを考えてください。
  【func_str.c】

```c
#include <stdio.h>

void func_str(char[]);  // 関数プロトタイプ

int main(int argc, char const *argv[]) {
    char str[] = "Hello World!";

    printf("%s\n", str);

    func_str(str);

    printf("%s\n", str);

    return 0;
}

/**
 * @brief 文字列を受け取り、なにか処理する
 *
 * @param str 対象の文字列
 */
void func_str(char str[]) {
    int i = 0;
    while (str[i]) {
        str[i++]++;  // 各文字を1つ進める
    }
}
```

練習

- 以下のプログラムを作成してください。(prog4_1.c)

> https://ja.wikipedia.org/wiki/%E3%82%B7%E3%83%BC%E3%82%B6%E3%83%BC%E6%9A%97%E5%8F%B7

  - Caesar暗号（シーザー暗号）のアルゴリズムで暗号化する関数`enc_caesar()`を作成する
    - 関数内で文字列を書き換えるものとする。
    - とりあえず鍵は5とする。
    - 大文字のみ扱えるものとする。
      - アルファベット大文字以外はそのままとする
  - キーボードから文字列を入力する
    - Caesar暗号で暗号化した文字列をmain側で出力する。

```
int main(int argc, char const *argv[]) {
    char str[80];

    printf("文字列を入力してください：");
    scanf("%s", str);

    enc_ceasar(str);
    printf("暗号化完了：%s\n", str);
    return 0;
}
```

- 以下のプログラムを作成してください。(prog4_2.c)

  - Caesar暗号（シーザー暗号）のアルゴリズムで暗号化する関数`dec_caesar()`を作成する
    - 関数内で文字列を書き換えるものとする。
    - とりあえず鍵は5とする。
    - 大文字のみ扱えるものとする。
      - アルファベット大文字以外はそのままとする
  - キーボードからprog4_1.cで生成された文字列を入力する
    - Caesar暗号で復号化した（暗号化される前の）文字列をmain側で出力する。

```
int main(int argc, char const *argv[]) {
    char str[80];
```

```
    printf("文字列を入力してください：");
    scanf("%s", str);

    enc_ceasar(str);
    printf("暗号化完了：%s\n", str);

    dec_ceasar(str);
    printf("復号化完了：%s\n", str);

    return 0;
}
```

資料

- ASCIIコード表は確認しておいてください。

> https://www.coins.tsukuba.ac.jp/~syspro/2005/No2_files/ASCII-ctrl.html

ASCIIコード表

上位3ビット→ ↓下位4ビット	0	1	2	3	4	5	6	7
0	NUL	DLE	SP	0	@	P	`	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAC	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF/NL	SUB	*	:	J	Z	j	z
B	VT	ESC	+	;	K	[	k	{
C	FF	FS	,	<	L	＼	l	|
D	CR	GS	-	=	M	]	m	}
E	SO	RS	.	>	N	^	n	~
F	SI	US	/	?	O	_	o	DEL

制御符号

- **NUL** ヌル(空文字)
- **SOH** ヘディング開始
- **STX** テキスト開始
- **ETX** テキスト終了
- **EOT** 伝送終了
- **ENQ** 問い合わせ
- **ACK** 肯定応答
- **BEL** ベル
- **BS** バックスペース
- **HT** 水平タブ
- **LF/NL** 復帰/改行
- **VT** 垂直タブ
- **FF** 改ページ
- **CR** 復帰
- **SO** シフトアウト
- **SI** シフトイン
- **DLE** データリンクでの拡張
- **DC1** 制御装置1
- **DC2** 制御装置2
- **DC3** 制御装置3
- **DC4** 制御装置4
- **NAC** 否定応答
- **SYN** 同期文字
- **ETB** 伝送ブロック終了
- **CAN** 取消
- **EM** 媒体終端
- **SUB**
- **ESC** (制御コード)拡張
- **FS** ファイルセパレータ
- **GS** グループセパレータ
- **RS** レコードセパレータ
- **US** ユニットセパレータ
- **SP** (半角)スペース
- **DEL** 削除