

第8回C言語講習(文字列)

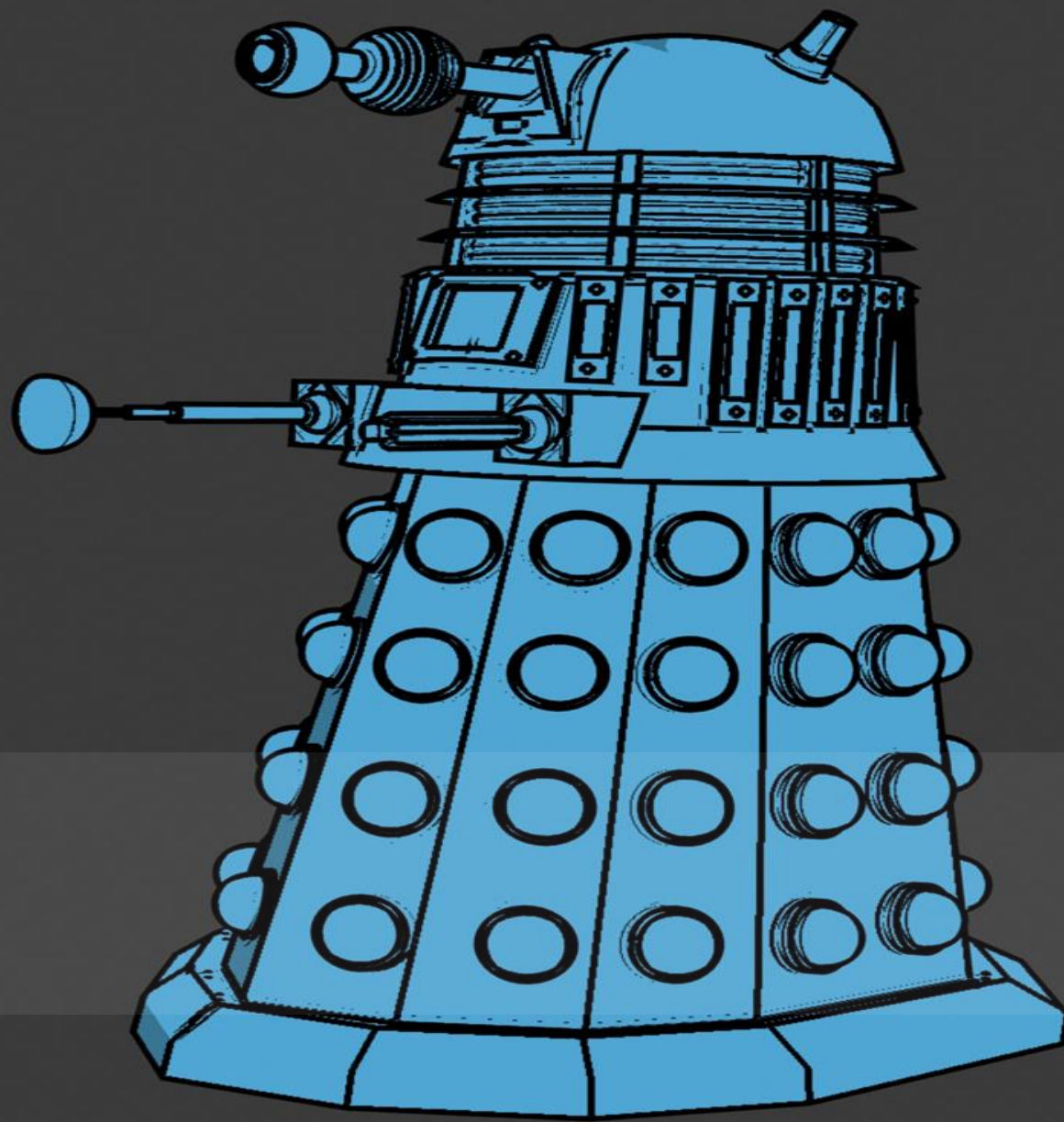
情報研究会 CACTUS



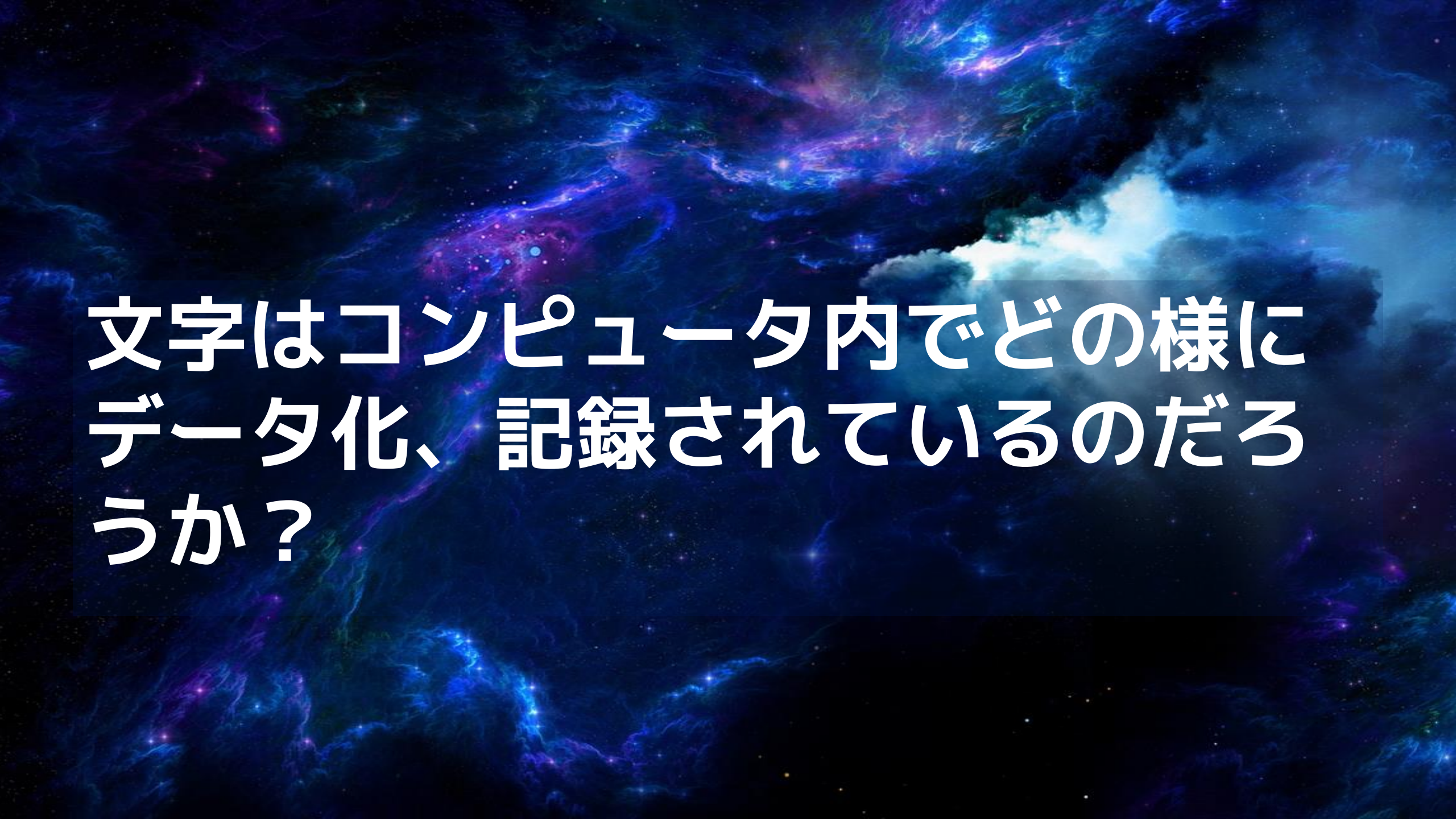
本日の内容

- 文字コード
- 文字
- 文字列(NULL文字)

文字コード



John



文字はコンピュータ内でどの様に
データ化、記録されているのだろ
うか？

A. 文字コードというものをを用いて記録される

- 文字コードとは、コンピュータで使用する文字に番号(コード)を割り振ったものである。いわば、文字の出席番号のようなもの。

**文字コードについて、コンピュータの
気分になってみよう！**

文字化け

- 文字コードは、文字と数値が1:1で対応していれば良いので、その組み合わせはいくらでもある。そこで、文字コードは目的に合わせて様々な種類がある。
- 別の文字コードで書かれた文章を読もうとすると、別の文字として認識されてしまうので、異なる文字として扱われてしまう。

機種依存文字

- 文字の中には、そのマシンにのみ搭載されている特殊な文字がある場合がある。これを機種依存文字という。
- こういったものは、別のマシンでどうやっても表すことができないので、どうしても利用した場合は自分のマシンだけで利用すること。

文字



Sample program 1

```
1 #include <stdio.h>
2
3 void main(){
4     char a,b;
5     printf("文字を入力:");
6     scanf("%c",&a);
7     b='a';
8     printf("変数aに保存された文字: '%c'\n",a);
9     printf("'%c'の文字コード: %d\n",a,a);
10    printf("変数bに保存された文字: '%c'\n",b);
11    printf("'%c'の文字コード: %d\n",b,b);
12 }[EOF]
```

※注意

[EOF]は入力しないこと

文字型 char

- C言語で文字を扱うときは、基本的にchar型を使う。
- Char型一つに半角英数字が1つ入る。
- 文字を表示、入力するときは書式指定子%cを使用する。

型名	サイズ	表現範囲
char	1 Byte	-128 ~ 127 (環境によっては0~255)

‘シングルクォート’

- プログラム中にデータとして文字を扱うときにはシングルクォート’で囲う。

どうして囲ってやる必要があるのだろうか？
サンプル中の7行目のクォートを外して実行してみよう。

確認問題1

文字を1文字入力

a

入力文字は'a'ですよん

文字を1文字入力

b

入力文字は'a'ではないね

入力された文字列によって、表示内容を変えるプログラムを作成せよ。

‘a’が入力された時とそれ以外が入力された時で分ける。

文字列



Sample program2

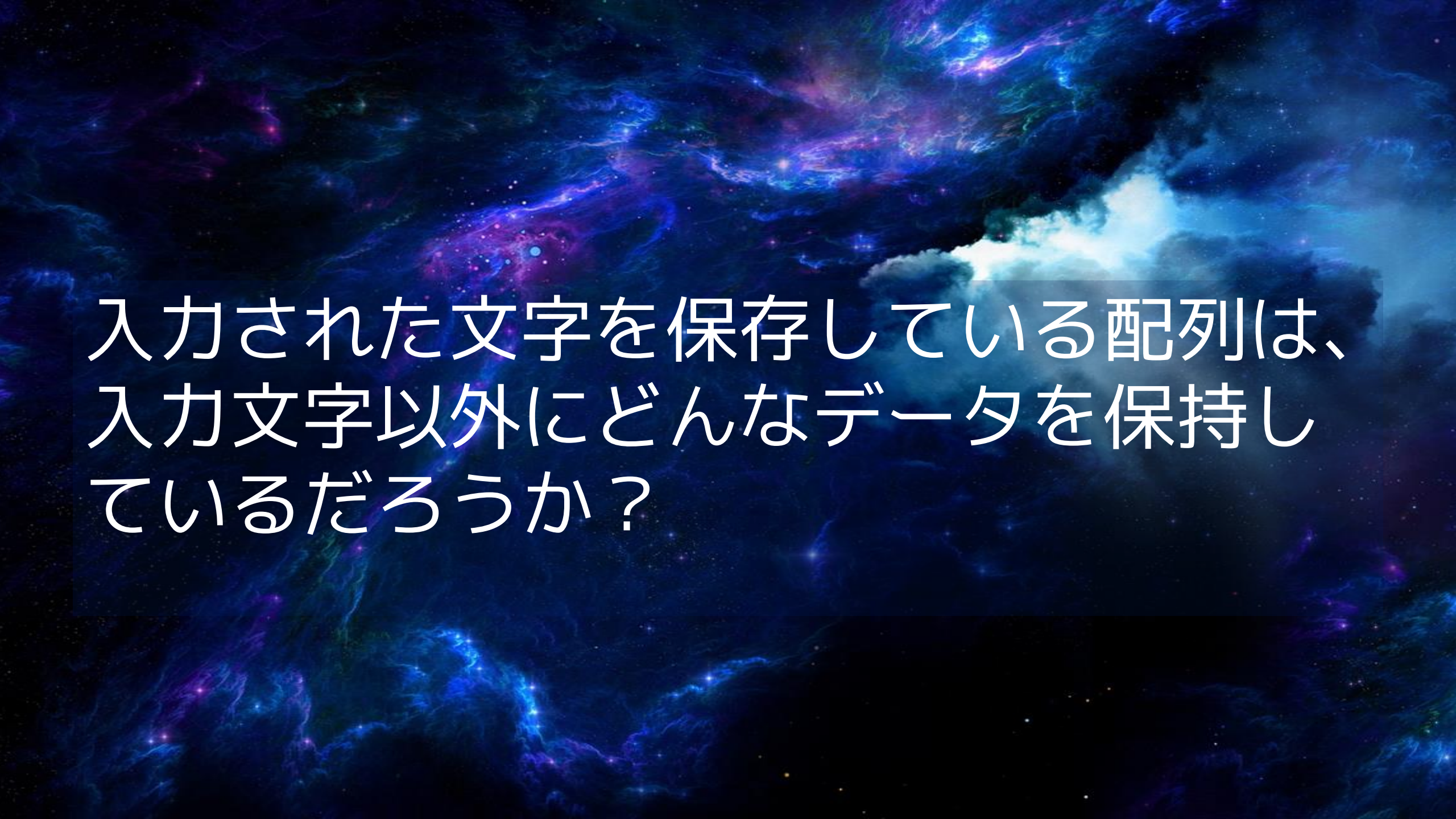
```
1 #include <stdio.h>
2 #define STR_NUM 256
3
4 void main(){
5     char str[STR_NUM];
6     printf("文字列を入力してください\n");
7     scanf("%s",str);
8     printf("入力文字列:%s\n",str);
9 }[EOF]
```

※注意

[EOF]を入力したら殺す

文字列を扱うには

- C言語で文字列を扱うには、char型の配列を使う。
- 文字列を入力、表示するときは書式指定子%sを使用する。
- 配列のサイズは、入力する文字数+1よりも必ず大きくなければいけない。



入力された文字を保存している配列は、
入力文字以外にどんなデータを保持し
ているだろうか？

Sample program2を書き換えよう

```
1 #include <stdio.h>
2 #define STR_NUM 256
3
4 void main(){
5     char str[STR_NUM];
6     int i; //追加
7     printf("文字列を入力してください\n");
8     scanf("%s",str);
9     //printf("入力文字列:%s\n",str);
10    for(int i=0;i<STR_NUM;i++){
11        printf("%c",str[i]);
12    }
13 }[EOF]
```

※注意

[EOF](ry

実行例

文字列を入力してください

tyrael

```
tyrael      . .      . L ^m . w/lz . q . |L . N -l . w      \ .      -      x .      E1ж~Mλ= . .  
 . . ^m . w'ə2v l?v . N      ≈@ = 2 B42v 42v⌋      ≈ .      -1@ [ ⌋      X .      r:@ ⌋      2  
      -3@ .      [ @ ⌋      .      才 @ ⌋      .      . @ ⌋      ≈@ |      -3@ X .      ;I@ ʒ3@      ⌋  
⌋      ≈@      ㇿ@      ト≈@      . ~h A |
```


実行結果より、データ以外には、初期化されていないゴミが入っている事が分かる。

では、何故 %s で表示するときには、これらのゴミデータが表示されないのだろうか？

A. 文字列を区切る文字がデータ中に存在する

- 入力された文字列には、null文字という文字列が有効な範囲を示すための文字が末尾に付与される。
- この文字は、表示する際には無視されるが、他の文字と同様に配列の1要素を利用して記録される。
- null文字の値(文字コード)をプログラム中で使用する場合は'¥0'とする。

t	y	r	a	e	l	¥0	ゴミ	...
---	---	---	---	---	---	----	----	-----

Sample program3

```
1 #include <stdio.h>
2 #define STR_NUM 256
3
4 void main(){
5     char str1[STR_NUM];
6     char str2[STR_NUM]="ArthurScherbius";
7
8     printf("文字列を入力:");
9     scanf("%s",str1);
10    if(str1==str2) printf("文字列は%sと等しいです",str2);
11    else printf("文字列は%sではない?こいつはコトだ!",str2);
12 }[EOF]
```


“ダブルクォート”

- 文字列をデータとして扱うときは、ダブルクォート”で囲う。
- 囲わなければならない理由は大体文字の時と同様。



A vibrant cosmic background featuring a complex network of blue and purple nebulae, with bright star clusters and distant galaxies visible against the dark void of space.

正しく比較できないはずです。

何故比較できないんだろう？

思い出してみよう

- 配列名を指定した場合、その配列の先頭番地を指す。
- つまり...今回のサンプルプログラムでは、2つの配列が同じアドレスであるかという比較演算になっているので、常に結果は `false`

str1			
i	n	u	¥n
0	1	2	3

str2			
i	n	u	¥n
211	212	213	214

A deep space image showing a vast cosmic landscape. In the foreground, there are wispy, glowing nebulae in shades of blue, purple, and pink. In the background, several spiral galaxies are visible, some appearing as bright, glowing clouds of light. The overall scene is set against a dark, star-filled sky.

どうしたら比較できるだろう？

A. 一文字ずつ比較する

```
1 #include <stdio.h>
2 #define STR_NUM 256
3
4 int compareString(char*,char*);
5
6 void main(){
7     char str1[STR_NUM];
8     char str2[STR_NUM]="ArthurScherbius";
9
10    printf("文字列を入力:");
11    scanf("%s",str1);
12    if(compareString(str1,str2))
13        printf("文字列は%sと等しいです",str2);
14    else
15        printf("文字列は%sではない?こいつはコトだ!",str2);
16 }
17
18 int compareString(char* str1,char* str2){
19     int i;
20     for(int i=0;i<STR_NUM;i++){
21         if(str1[i]!=str2[i])
22             return false;
23         if(str1[i]=='\0' || str2[i]=='\0')
24             break;
25     }
26     return true;
27 }[EOF]
```

いちいち関数を作るのは
めんどくさいので . . .

string.h

- c言語には文字列を処理するための関数が標準ライブラリとして実装されている。
- string.hには文字列の比較を行う関数のほか、結合を行う関数やコピーする関数などが用意されている。



```
int strcmp(const char*,const char*)
```

- 文字列の比較を行う関数
- 戻り値は
 - 文字列が等しい時: 0
 - 引数1が辞書順で後の時: 正数
 - 引数2が辞書順で前の時: 負数

確認問題2

- sample program3をstrcmp()関数を使って、正しく動く形に書き換えてみよう。

補足：マルチバイト文字

- char型一つで表せる数値は256種類しかない。
- しかし、大小アルファベットと数字、必要最低限の数学記号などに加え、日本語の文字をデータ化するとき、文字コードは256では済まない。
- そこで、char型を複数利用して、文字列を表現する。
- このように、多バイトで表現する文字をマルチバイト文字という。

練習問題1

```
文字列1:Alan  
文字列2:Turing  
AlanTuring
```

文字列を 2 つ入力し、それを結合して出力せよ

(string.hの関数を利用してもよい)

※string.hの関数を利用する場合は
自分で調べること

練習問題2

```
式: 25+11-45  
num: -9
```

自然数と+記号、-記号だけを文字列として入力し、その計算結果を出力せよ。

<ヒント>atoi関数を使うと楽ができる。atoiについては調べること

※オペランドは自然数だけとするので、11+-24のような式はありえない

チャレンジ問題1

```
文字列:dexter  
shift:21  
yzsozm
```

文字列と自然数Sを入力し、シフト数をSとしたシーザー暗号を出力せよ。
(アルファベットが文字コード上で連番であることを利用すると良い)

※シーザー暗号は、シフト数に従って、文字をずらす暗号

※入力する文字列は小文字アルファベットだけで構成されているとする

チャレンジ問題2

```
文字列:captainjack  
shift:23  
暗号化:zxmqxfkgxzh  
復号:captainjack
```

文字列と自然数Sを入力し、Sをシフト数として、文字列をシーザー暗号で復号せよ。

(問題1のものを書き換えて、暗号化したものを復号すると良い)

※復号とは暗号を元の文(平文)に戻すこと

チャレンジ問題3

```
str:raxacoricofallapatorius  
key:isolus  
暗号:zsllwgzaqzzstdoaulwjwfm  
復号:raxacoricofallapatorius
```

文字列1と2を入力し、文字列2を鍵として、文字列1をヴィジユネル暗号で暗号化、またその文字列を復号して平文が得られることを確認せよ。

※ヴィジユネル暗号について調べよ