C言語勉強会

お品書き

- 前回のおさらい
- 演算子
- 型
- 配列
- ポインタってなーに(さわり)
- 次回への布石。

1.前回のおさらい

おまじないってなんだろう？

```c
#include <stdio.h>

int main (void)
{

    printf("Hello, World!");

    return (0);
}
```

```c
/*参照するライブラリを指定*/
#include <stdio.h>


/*cファイルの中での関数の定義*/
/*戻り値の型　関数名　(引数(関数に渡さなくてはいけない値))*/
int main (void)
{

    /*printf-出力関数-*/
    /*関数名(引数);の形で使う。*/
    printf("Hello, World!");


    /*戻り値の返し方*/
    /*return (戻り値);として返す。*/
    //宣言時の型とあっていないと怒られるので注意。
    //実は必ずしも値を返す必要はない。
    return (0);
}
```

解答

前回出てきた出力関数と繰り返し構文

- **printf**
- **for文**
- **While文**
- **do…while文**

前回出てきた出力関数と繰り返し構文

printf ←標準出力関数、第一引数を出力

for文 ←繰り返す回数がわかってるときに使う

While文 ←繰り返す回数がわからないときに使う

do…while文←1回以上繰り返したいときに使う

無限ループは絶対に回避してね!!!!

2.演算子

算術演算子(四則演算や、剰余)

代入演算子(前回ちょっぴり触れたイコールの話)

論理演算子(かつ(論理積)、または(論理和)、排他的論理和)

比較演算子(代入演算子とごっちゃにならないように…)

三項演算子(多用しないように注意だけど、他の言語でも似たものがあるので扱いますー)

算術演算子

演算子	意味
+	足し
-	引き
*	掛け
/	割り
%	剰余

代入演算子

演算子	意味
x=y	**xをyに代入**
x+=y	x ＋ yをxに代入
x-=y	x － y をxに代入
x/=y	x ／ y をxに代入
x%=y	x ％ y をxに代入

論理演算子

演算子	意味
A && B	A AND B
A || B	A OR B
!	NOT

比較演算子(1)

演算子	意味
x==y	**xとyが等価であるか？**
x!=y	xとyは等しくないか？
x<y	xはyより小さい(未満)
x>y	xはyより大きい

比較演算子(2)

演算子	意味
x <= y	x はy以下である
x >= y	xはy以上である

間違えやすい演算子まとめ

演算子	意味
x=y	xにyを代入
x==y	xとyは等しいか？
x>=y	xはy以上
x(算術演算子)=y	x(算術演算子)yの結果をxに代入

インクリメント・デクリメント演算子

演算子	意味
x++	?
++x	?
x--	?
--x	?

インクリメント・デクリメント演算子

演算子	意味
x＋＋	(後置演算)xに**1足してxを評価**
＋＋x	(前置演算)xを**評価してからxに1足す**
x−−	(後置演算)xから**1引いてxを評価**
−−x	(前置演算) xを**評価してからxから1引く**

後置演算と前置演算の例

- Q.右のコードでは出力結果はどうなるだろう？

```c
/*参照するライブラリを指定*/
#include <stdio.h>

/*戻り値の型　関数名　(引数の型)*/
int main(void)
{
    int i = 0;

    /*後置演算子*/
    printf("increment after\n");
    while (i++ < 5)
        printf("Hi!");

    /*変数の初期化。これをしないとiは5のまま*/
    i = 0;

    /*前置演算子*/
    printf("\nincrement before\n");
    while (++i < 5)
        printf("Hi!");


    /*main関数の戻り値*/
    return 0;
}
```

後置演算と前置演算の例

- A.以下のような出力結果になるよ

```
increment after
Hi!Hi!Hi!Hi!Hi!
increment before
Hi!Hi!Hi!Hi!
```

```c
/*参照するライブラリを指定*/
#include <stdio.h>

/*戻り値の型　関数名　(引数の型)*/
int main(void)
{
    int i = 0;

    /*後置演算子*/
    printf("increment after\n");
    while (i++ < 5)
        printf("Hi!");

    /*変数の初期化。これをしないとiは5のまま*/
    i = 0;

    /*前置演算子*/
    printf("\nincrement before\n");
    while (++i < 5)
        printf("Hi!");


    /*main関数の戻り値*/
    return 0;
}
```

後置演算と前置演算の例

A.以下のような出力結果になるよ！

```
increment after
Hi!Hi!Hi!Hi!Hi!
increment before
Hi!Hi!Hi!Hi!
```

←0から1ずつ増えて5になって条件を満たさないから処理をしないで終わる

↑1から1ずつ増えて5になって条件を満たさないから処理をしないで終わる

```c
/*参照するライブラリを指定*/
#include <stdio.h>

/*戻り値の型　関数名　(引数の型)*/
int main(void)
{
    int i = 0;

    /*後置演算子*/
    printf("increment after\n");
    while (i++ < 5)
        printf("Hi!");

    /*変数の初期化。これをしないとiは5のまま*/
    i = 0;

    /*前置演算子*/
    printf("\nincrement before\n");
    while (++i < 5)
        printf("Hi!");


    /*main関数の戻り値*/
    return 0;
}
```

おまけ

- **sizeof**は演算子だよ！

演算子	意味
sizeof(型名や変数)	型名や変数の大きさを返す！

3.型

- 型の種類
- データ型の種類(今回はやらんでおこう)
- 型修飾子

型の種類

実行環境によって異なる

型名	意味	最大値	大きさ
void	型なし	-	-
char	文字型	-128 ～ 128	1
int	整数型	-2147483648~2147483647	4
float	単精度浮動小数点型	-3.402823e+38 ~ 3.402823e+38	4
double	倍精度浮動小数点型	-1.797693e+308~1.797693e+308	8

型修飾子の種類(頻出物のみ)

型名	意味
unsigned	符号なし
const	定数(キャストしないと中身を変えられない)

おまけ

sizeofで大きさを確認してみよう！

printfの使い方もちょっと拡張してみよう

printfの使い方

- **定義**

型-> int型 :出力した文字数をバイト数で出力

引数->可変長引数。第一引数はconst char*型

- 主な使い方

書式指定子を第一引数の間に配置し、そこに対応した第二引数以降を表示する。

書式指定子

型に対応している。

書式指定子	対応する型	説明
%c	char	一文字を出力
%s	char *	文字列を出力
%d	int	整数を10進数で出力
%u	unsigned int	符号なし整数を10進数で出力
%x	unsigned int, int	整数を16進数で出力
%f	float, double	実数を出力

ためしにやってみよう

- 名前と年齢を出力してみよう！
- ヒント: ‘¥(バックスラッシュ)n’で出力文字を改行できるよ！

ためしにやってみよう(解答)

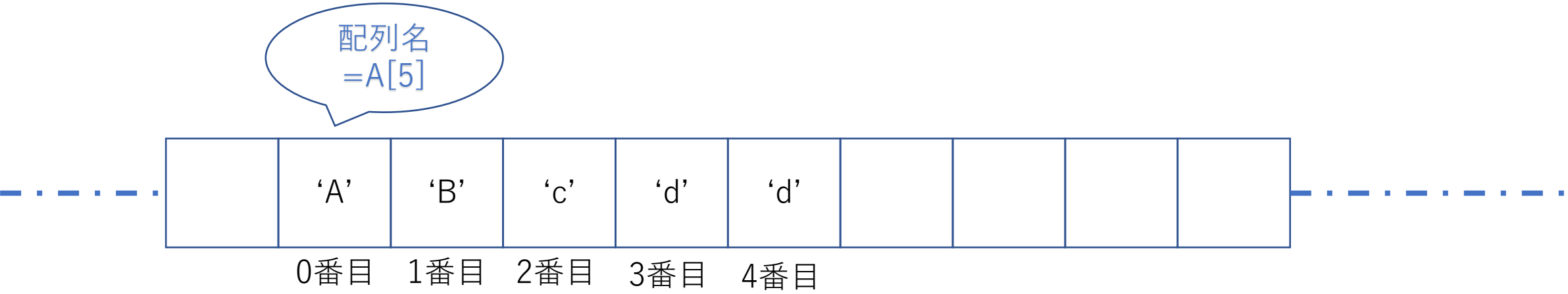
```c
/*参照するライブラリを指定*/
#include <stdio.h>

/*戻り値の型　関数名　(引数の型)*/
int main(void)
{
    /* printf */
    /*%s = 文字列を出力*/
    /*%d = 数字を10進数で出力*/
    printf("My Name is = %s\nMy Age = %d", "Hinata Kikuchi", 17);

    /*main関数の戻り値*/
    return 0;
}
```

- 名前と年齢を出力してみよう！
- ヒント: ‘¥(バックスラッシュ)n’で出力文字を改行できるよ！

4.配列について

- **配列**（**array**）とは，同一の型のデータを（メモリ上に）一列に（隙間をあけずに）並べたものである。
- 違う型のデータを混在して並べて配列とすることはできない。
- **配列中の各データを，配列の要素**（element）という。
- **配列には**，その配列全体を指すための名前（**配列名**）がついている。

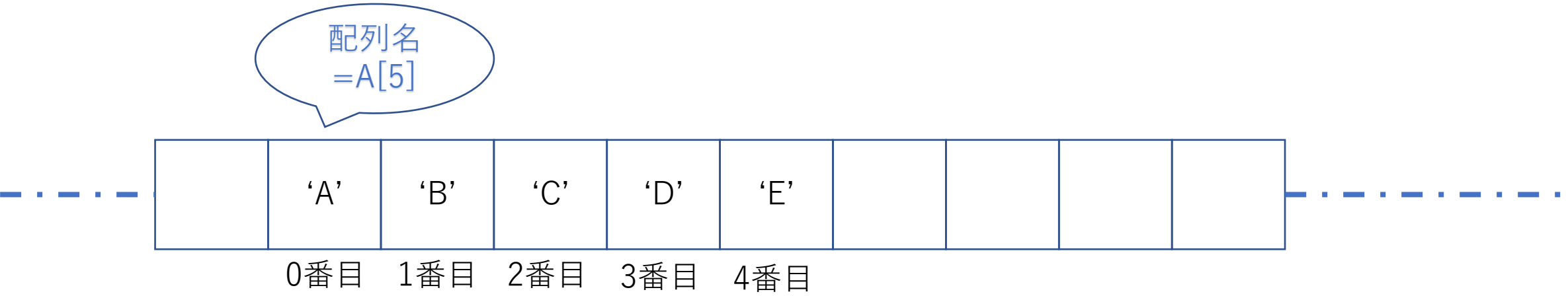
イメージ



じゃあ文字列って…？

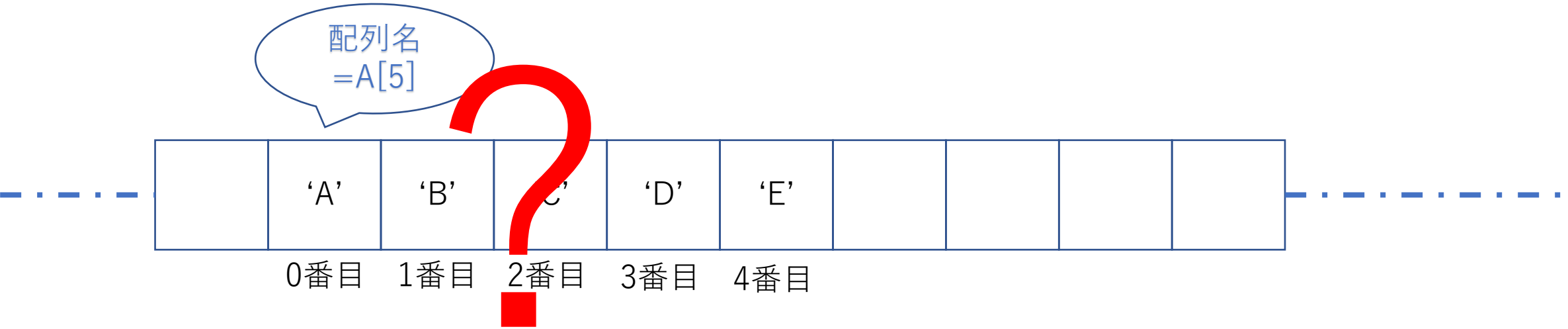
文字列も、文字型配列の一部なのだ！！！！

(例)
char Alphabet[7] = “ABCDEFG”;



あれでも…char *って何…？

(例)
char *Alphabet = “ABCDEFG”;



つづく…

参考文献

- [1]IT 専科 C言語入門演算子 (http://www.itsenka.com/contents/development/c/operator.html)
- [2]SAMURAI ENGINEER【C言語入門】型と変数の一覧(サイズ、範囲、宣言について解説)

(https://www.sejuku.net/blog/25517)

- [3]Wakaba Programming Schoolフォーマット指定子一覧

(https://www.k-cube.co.jp/wakaba/server/format.html)