C言語 夏期講習 （動的確保）

情報研究会 CACTUS

動的確保とは

- 通常のメモリの確保(変数宣言)では、コンパイル時(ビルド)にサイズが決まっていなければならない
- しかし、プログラムを実行してから配列のサイズを決めたい時などがある

nameは¥0含め最大10文字（決まっている）

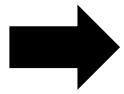
```c
#include <stdio.h>
#include <stdlib.h>

int main(void) {

    char name[10];

    scanf("%s", name);
    printf("Name:%s¥n", name);

    return 0;
}
```



実行してからサイズを決めたい

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

int main(void) {

    int n;
    scanf("%d", n);

    char name[n];

    scanf("%s", name);
    printf("Name:%s¥n", name);

    return 0;
}
```

動的確保とは

例　通常の配列 char[8]の場合

W	I	N	D	O	W	S	¥0

M	A	C	¥0				

L	I	N	U	X	¥0		

- 呼び出されてからメモリを確保する
- メモリのムダ使いを無くす
- 動的確保では通常とは違う領域にメモリ確保される

check! “ヒープ領域”,”スタック領域”

サンプルプログラム1

N個のデータの和をとるプログラム

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

int main(void) {

    int n, i, sum = 0;
    printf("データの個数を入力してください:");
    scanf("%d", &n);

    // 動的確保
    int* data = (int*)malloc(sizeof(int)*n);
    if (data == NULL) {
        printf("Allocation error.¥n");
        exit(0);
    }

    printf("データを入力してください¥n");
    for (i = 0; i < n; i++) {
        scanf("%d", &data[i]);
    }

    for (i = 0; i < n; i++) {
        sum += data[i];
    }

    printf("データの合計値は%dです¥n", sum);

    // メモリの開放
    free(data);

    return 0;
}
```

重要

超重要

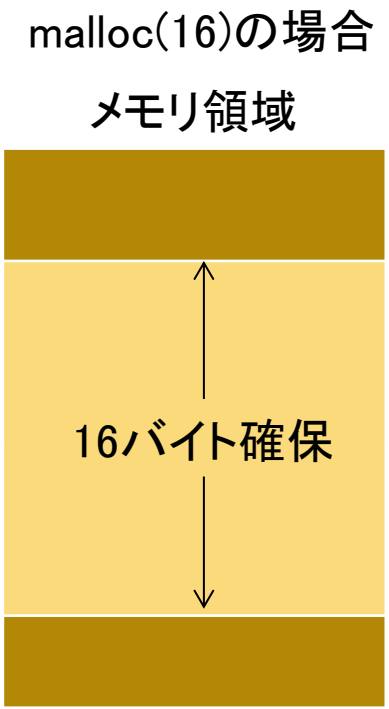
出力結果

```
データの個数を入力してください:6
データを入力してください
8
22
-7
11
-14
19
データの合計値は39です
```

malloc関数

```
// 動的確保
int* data = (int*)malloc(sizeof(int)*n);
if (data == NULL) {
    printf("Allocation error.¥n");
    exit(0);
}
```

malloc(16)の場合



malloc

書式	void* malloc(size_t size)
引数	size : 割り当てるバイト数
戻り値	確保した領域の先頭ポインタを返します。 メモリが不足している場合、NULLを返します。
説明	メモリブロックを割り当てます
要件	<stdlib.h>または<malloc.h>のインクルード

Microsoft C ランタイムライブラリ　リファレンス より要約

sizeof演算子

- 渡された変数や型のメモリサイズを返すもの

- 例
- sizeof(int)　　= 4(バイト)
- sizeof(double)　　= 8(バイト)
- sizeof(char)　　= 1(バイト)
- sizeof(char *)　　= 4(バイト)

free関数

```
// メモリの開放
free(data);
```

- malloc等で動的確保したメモリ領域を開放してくれる
- 開放された領域は次回の動的確保に使用できる

free

書式	void free(void* memblock)
引数	memblock：以前割り当てられ、解放されるメモリ ブロック
説明	メモリ ブロックを割り当て解除または解放します。
要件	<stdlib.h>または<malloc.h>のインクルード

サンプルプログラム2

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>

#define MAXLENGTH 64

int main(void) {

    int n, i;
    char stnc[MAXLENGTH]; // 一時的に文字列を入れる変数

    printf("行数を入力してください:");
    scanf("%d", &n);

    // 動的確保 (n行)
    char** data = (char**)malloc(sizeof(char*)*n);
    if (data == NULL) {
        printf("Allocation error.¥n");
        exit(0);
    }

    printf("文字列を入力してください(max 63)¥n");
    for (i = 0; i < n; i++) {
        scanf("%s", stnc);

        // 動的確保(m文字)
        data[i] = (char *)malloc(strlen(stnc) + 1);
        if (data[i] == NULL) {
            printf("Allocation error.¥n");
            exit(0);
        }

        // 確保したメモリ領域に文字列をコピー
        strcpy(data[i], stnc);
    }

    // 先頭の文字を大文字へ変換
    for (i = 0; i < n; i++) data[i][0] = toupper(data[i][0]);


    printf("----------全文表示----------¥n");
    for (i = 0; i < n; i++) printf("%s¥n",data[i]);

    // メモリの開放
    for(i = 0; i < n;i++) free(data[i]);
    free(data);

    return 0;
}
```

文字列の先頭を
大文字にするプログラム

出力結果

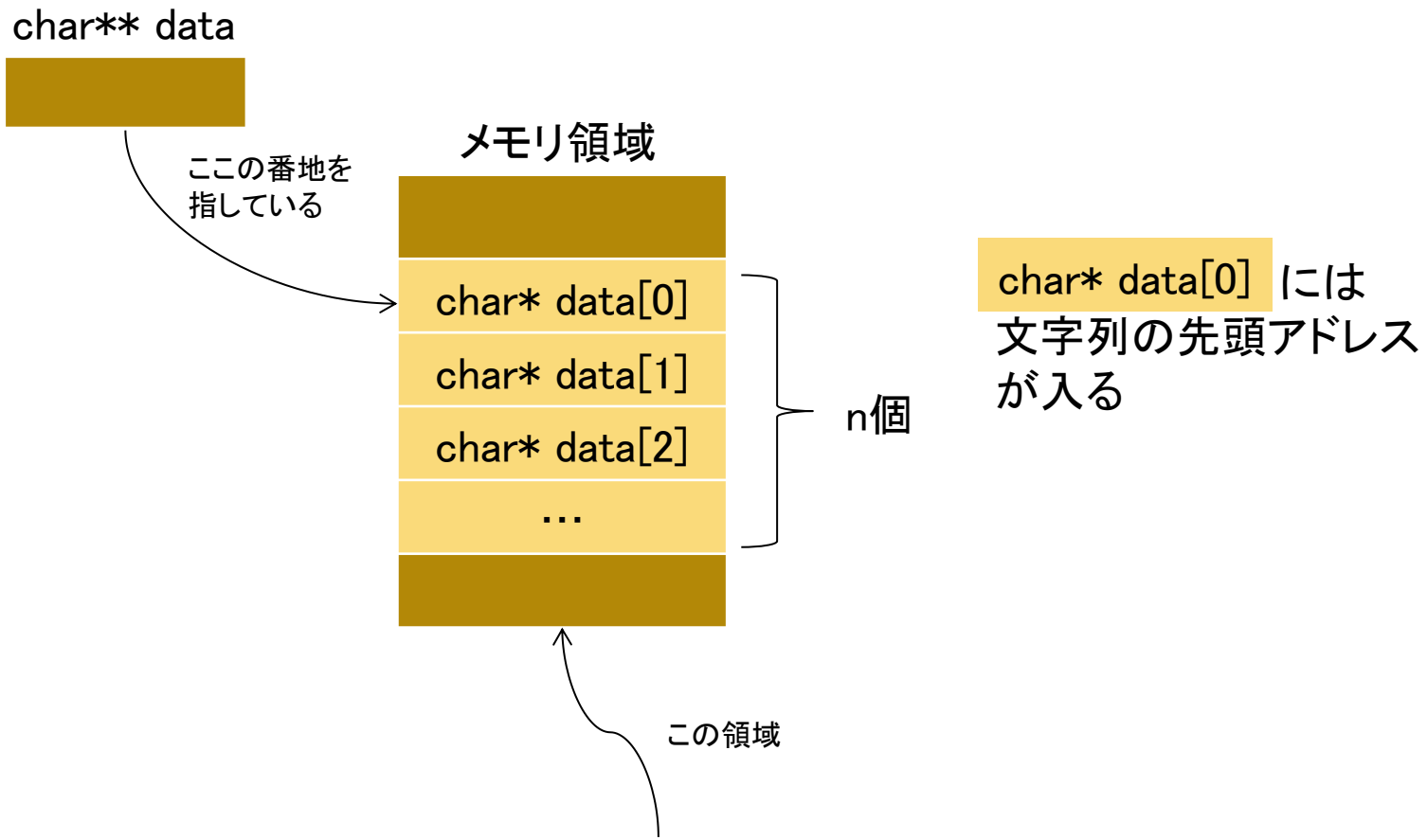
```
行数を入力してください:9
文字列を入力してください(max 63)
there
is
white
dog,
his
tails
is
white
too.
----------全文表示----------
There
Is
White
Dog,
His
Tails
Is
White
Too.
```

2次元でのmalloc

```
// 動的確保（n行）
char** data = (char**)malloc(sizeof(char*)*n);
if (data == NULL) {
    printf("Allocation error.¥n");
    exit(0);
}
```

char** data
→ [char*]* data
→ charポインタのポインタ
→ charポインタの配列の先頭アドレス
→ char配列の先頭アドレスの配列の先頭アドレス
→ 文字列の先頭アドレスの配列の先頭アドレス

2次元でのmalloc



char* data[0] には
文字列の先頭アドレス
が入る

data = malloc(sizeof(char*)*n) → charポインタがn個分入る領域を確保

2次元でのmalloc

```
printf("文字列を入力してください(max 63)¥n");
for (i = 0; i < n; i++) {
    scanf("%s", stnc);

    // 動的確保(m文字)
    data[i] = (char *)malloc(strlen(stnc) + 1);
    if (data[i] == NULL) {
        printf("Allocation error.¥n");
        exit(0);
    }

    // 確保したメモリ領域に文字列をコピー
    strcpy(data[i], stnc);
}
```

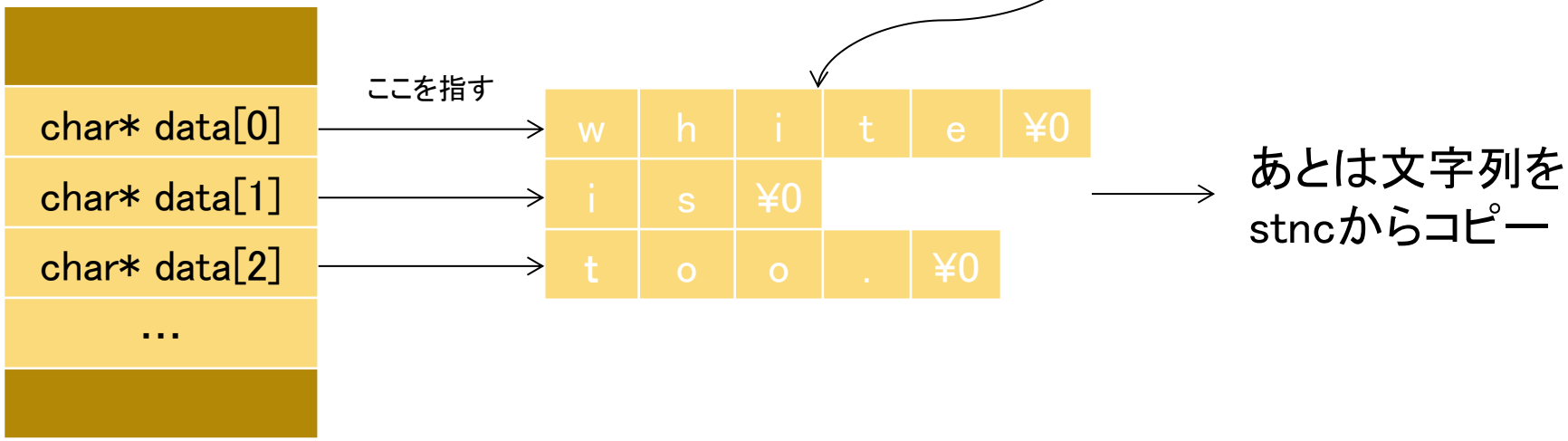
scanfから入力

stnc[MAXLENGTH]

w	h	i	t	e	¥0			

文字列の長さ+1個分の領域を確保

メモリ領域



2次元でのfree

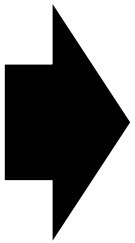
```
// メモリの開放
for(i = 0; i < n;i++) free(data[i]);
free(data);
```

char* data[?] が
指している領域を開放

こいつらを開放

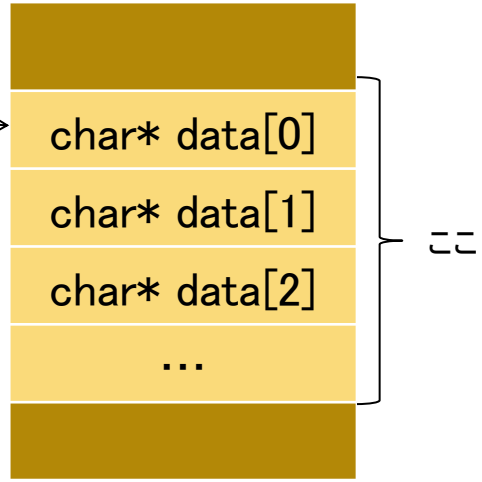
w	h	i	t	e	¥0
i	s	¥0			
t	o	o	.	¥0	

for(i=0;i<n;i++) free(data[i])



char** data

が指している
領域を開放



free(data)

練習問題1

- 文字列を入力し、奇数番目の文字をつなげて表示して下さい。mallocを使うこと。
（日本語に対応しなくても良い）
- 例

```
文字列を入力してください(max 15)
abcdefghijk
-------------奇数番目の文字を表示-------------
acegik
```

```
文字列を入力してください(max 15)
nky1aU-qn(
-------------奇数番目の文字を表示-------------
nya-n
```

- check! “realloc”,”calloc”

列挙体

- 文字を定数として扱う方法の1つ

```
enum rekkyo {
    zero, first, second, third, forth, fifth
};
```

enum 名前{ 文字列 }と書くことで、文字列に連番が割り当てられ
プログラムで扱えるようになる

rekkyo num;　　のように宣言でき、zero～fifthの値をとる

特に指定がない場合
zero → 0
first → 1
second → 2
・・・
となるが、

```
enum rekkyo {
    zero,
    first,
    third = 3,
    forth
};
```

のように書くと、
zero → 0
first → 1
third → 3
forth → 4
となる

- ゲーム製作におけるフラグ、難易度などに使うと良いでしょう。

列挙体の例

```c
#include <stdio.h>

enum date {
    Sun, Mon, Tue, Wed, Thu, Fri, Sat
};

enum weather {
    Rainy = -1, Cloudy, Sunny
};

typedef struct days {
    enum date youbi;
    enum weather tenki;
}DAY;

int main(void) {

    int n;
    DAY today = { Mon, Sunny }, nextday;

    printf("今日は月曜日です。何日後の天気が知りたいですか？¥n");
    scanf("%d", &n);

    nextday.youbi = (today.youbi + n) % 7;

    switch (nextday.youbi) {
    case Sun:
    case Mon:
    case Wed:
    case Fri:
        nextday.tenki = Sunny;
        break;
    case Tue:
    case Sat:
        nextday.tenki = Rainy;
        break;
    case Thu:
        nextday.tenki = Cloudy;
    }

    printf("%d日後は",n);
    switch (nextday.youbi){
    case Sun:
        printf("日曜日");
        break;
    case Mon:
        printf("月曜日");
        break;
    case Tue:
        printf("火曜日");
        break;
    case Wed:
        printf("水曜日");
        break;
    case Thu:
        printf("木曜日");
        break;
    case Fri:
        printf("金曜日");
        break;
    case Sat:
        printf("土曜日");
    }

    printf("です。天気は");

    switch (nextday.tenki){
    case Rainy:
        printf("雨です¥n");
        break;
    case Cloudy:
        printf("曇りです¥n");
        break;
    case Sunny:
        printf("晴れです¥n");
    }

    return 0;
}
```

```
今日は月曜日です。何日後の天気が知りたいですか？
4
4日後は金曜日です。天気は晴れです
```