

情報研究会CACTAS

第七回 講義資料

今週の内容

- ポインタ

前回の練習問題の解答

```
1  #include <stdio.h>
2  #define N 60
3
4  typedef struct {
5      int math;
6      int jap;
7      int eng;
8      double mathave;
9      double japave;
10     double engave;
11 } grade;
12
13 grade ave(grade student[], int);
14
15 int main(void) {
16     grade student[N] = { 0 }, result;
17     int i = 0;
18
19     printf("生徒の成績(数学、国語、英語)を入力してください。(入力終了するときは-1を入力する)\n");
20     while (1) {
21         printf("%d:\n", i + 1);
22         scanf("%d%d%d", &student[i].math, &student[i].jap, &student[i].eng);
23         if (student[i].math == -1 || student[i].jap == -1 || student[i].eng == -1) {
24             break;
25         }
26         else {
27             i++;
28         }
29     }
```

```

25     }
26     else {
27         i++;
28     }
29 }
30 result = ave(student, i);
31 printf("数学の平均点:%.2f 国語の平均点:%.2f 英語の平均点:%.2f", result.mathave, result.japave, result.engave);
32
33
34     return 0;
35 }
36
37 grade ave(grade student[], int i) {
38     int k;
39     int mathsum = 0, japsum = 0, engsum = 0;
40     grade result = { 0 };
41
42     for (k = 0; k<i; k++) {
43         mathsum += student[k].math;
44         japsum += student[k].jap;
45         engsum += student[k].eng;
46     }
47     result.mathave = (double)mathsum / i;
48     result.japave = (double)japsum / i;
49     result.engave = (double)engsum / i;
50     return result;
51 }

```

```

1  #include <stdio.h>
2  #include <math.h>
3  #define _USE_MATH_DEFINES
4
5  typedef struct {
6      double x;
7      double y;
8  } bect;
9
10 typedef struct {
11     double area;
12     double theta;
13 } multi;
14
15 multi create(bect[], multi);
16
17 int main(void) {
18     bect direct[2];
19     multi bector = { 0 };
20     int i;
21
22     printf("2つの2次元ベクトルの x、y 成分を入力してください\n");
23     for (i = 0; i < 2; i++) {
24         printf("x:");
25         scanf("%lf", &direct[i].x);
26         printf("y:");
27         scanf("%lf", &direct[i].y);
28     }
29     bector = create(direct, bector);
30     printf("2つのベクトルがつくる面積:%2f 角度差:%.2f° %n", bector.area, bector.theta);
31     return 0;
32 }
33
34 multi create(bect direct[], multi bector) {
35     double length1, length2, naiseki, cos;
36     length1 = sqrt(pow(direct[0].x, 2) + pow(direct[0].y, 2));
37     length2 = sqrt(pow(direct[1].x, 2) + pow(direct[1].y, 2));
38     naiseki = direct[0].x*direct[1].x + direct[0].y*direct[1].y;
39     printf("長さ1:%.2f 長さ2:%.2f 内積:%.2f\n", length1, length2, naiseki);
40     cos = naiseki / (length1*length2);
41     bector.theta = acos(cos);
42     bector.area = 1.0 / 2.0*sqrt(pow(length1, 2)*pow(length2, 2) - pow(naiseki, 2));
43     bector.theta *= 180.0 / M_PI;
44     return bector;
45 }

```

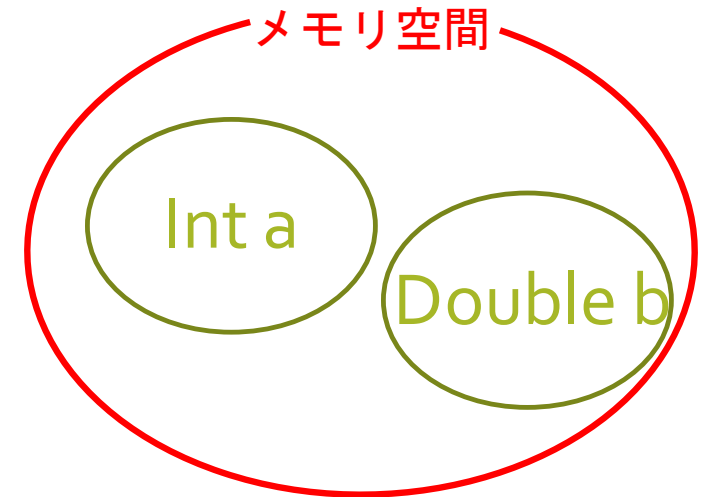
前回のおさらい問題

クラスの各生徒が読んだ本の数、見た映画、ドラマの数を入力し、
クラス全体で読まれた本の数、見た映画、ドラマの数を表示するプログラム

```
C:\WINDOWS\system32\cmd.exe
クラスの生徒1人1人の読んだ本の数、見た映画、ドラマの数を入力してください。(最大60人まで)
1人目:10 5 15
入力が続けますか?(Yes:1/No:-1):1
2人目:9 10 2
入力が続けますか?(Yes:1/No:-1):1
3人目:4 10 18
入力が続けますか?(Yes:1/No:-1):1
4人目:15 0 0
入力が続けますか?(Yes:1/No:-1):-1
クラス全体で読まれた本の数は38冊、映画は25本、ドラマは35本でした。
E:\Shamit\C言語\bccdev1221\プログラム\プロジェクト\kousyukakunin\Debug>_
```

メモリ空間

- 普段当たり前の如く使用している「int a」。
- int型の変数aを宣言しているのだが、コンピューター内ではどのような処理が行われているか？？
- メモリ空間上にint型の変数aの領域を確保した！！
ということ！！



バイトについて

思い出してみよう

int型	4バイト
float型	4バイト
double型	8バイト
char型	1バイト

256個の数字を扱える単位をバイトという。

例) 155→155

それより大きい数は2バイト用意して表せばいい。

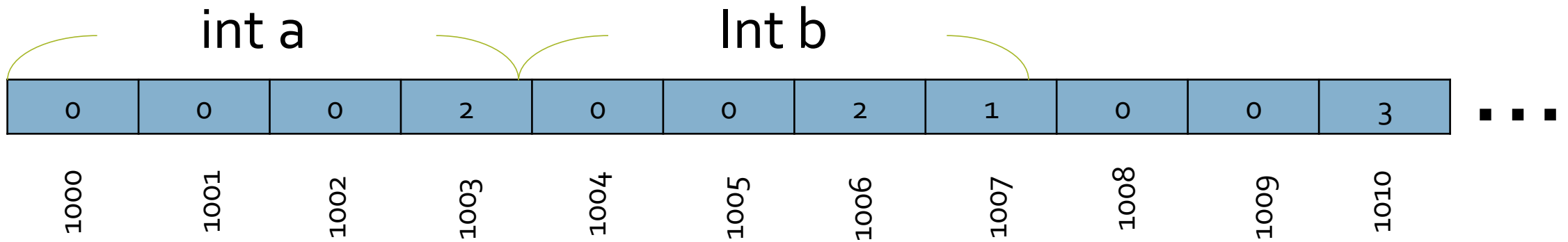
例) 356→1、100



Int型は4バイトなので、

$256^4 = 4294967296$
の数字を表すことができる。

メモリ空間



上図の様に1バイト毎に区切られた記憶領域がメモリ空間
その一つ一つに**アドレス**がついている

4バイトで区切る意味とは？

→メモリ空間にint型の変数の領域(4バイト)を確保する

例) int a = 2 → 4バイトで2を表す

構文

例) `int a = 5, b = 1;`

- ポインタを宣言するときは* を付ける

```
int *p;    double *nasubi;
```

- 変数のアドレスを知りたいときは& を付ける

```
p = &a;
```

- ポインタが指す変数を知りたいときは* を付ける

```
b = *p;
```

- ポインタが指す変数を上書きする

```
*p = b
```

確認問題

穴を埋めていきましょう

```
1 #include<stdio.h>
2
3 int main(void){
4     int a=5, b=1;
5     int ___, ___; //ポインタp1,p2の宣言
6     p1=__a; ... //p1にaの先頭アドレスを入れる
7     p2=__b; ... //p2にbの先頭アドレスを入れる
8     ___p2=a; ... //p2が指す変数にaを代入する
9     ___p1=b; ... //p1が指す変数にbを代入する
10
11     printf("%d,%d\n", ___p1, ___p2); ... //p1とp2が指す変数をそれぞれ出力する
12     printf("%d,%d\n", ___a, ___p1); ... //aとp1それぞれを使ってアドレスを出力する
13     return 0;
14
15 }[EOF]
```

サンプルプログラム 1

実行結果

```
1 #include<stdio.h>
2
3 int main(void){
4     int arr[5]={1,5,9,6,10};
5     int *p;
6     int i;
7
8     p=arr; //p=&arr[0]と同じ意味!!
9
10
11     printf("pは、%d\n",p);
12     printf("*pは、%d\n\n",*p);
13
14     //ここに後でコードを書いてください。
15
16     printf("今からpを一個ずつ足していきます。¥n");
17     for(i=1;i<=5;i++){
18         printf("%d回目¥n",i);
19         printf("pは、%d¥n",p);
20         printf("*pは、%d¥n¥n",*p);
21
22         p++;
23     }
24
25     return 0;
26 }[EOF]
```

C:\WINDOWS\system32\cmd.exe

```
pは、 1703728
*pは、 1

今からpを一個ずつ足していきます。
1回目
pは、 1703728
*pは、 1

2回目
pは、 1703732
*pは、 5

3回目
pは、 1703736
*pは、 9

4回目
pは、 1703740
*pは、 6

5回目
pは、 1703744
*pは、 10
```

配列とポインタ

配列を宣言すると連続して領域が確保される

ポインタ

「**変数の先頭アドレス**」を格納する**変数**

更にワンポイント！

ポインタは、**アドレス**と**データ型のサイズ**の二つの情報を持っている

サンプルの //コードを書いてください の部分に

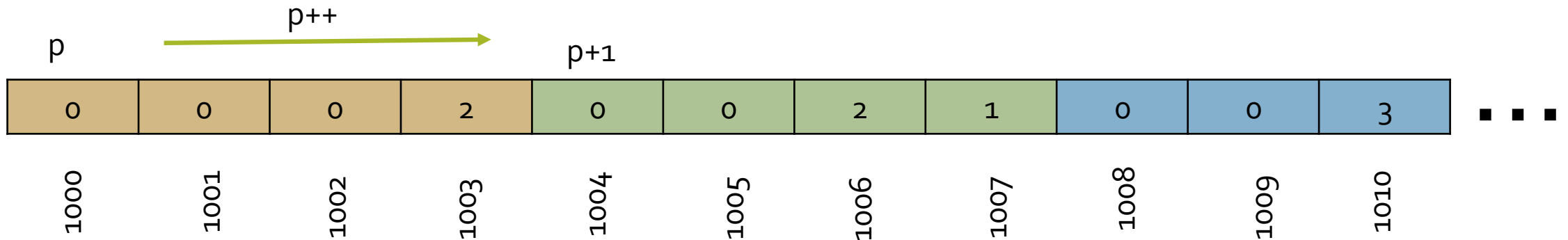
```
//ここに後でコードを書いてください。  
printf("*(p+1)を出力すると、 %d\n",*(p+1));  
printf("*(p+1)を出力すると、 %d\n\n",*(p+1));
```

を付け足してみよう

ポインタ	番地	変数
*p	50	Arr[0]
	51	
	52	
	53	
*(p+1)	54	Arr[1]
	55	
	56	
	57	
*(p+2)	58	Arr[2]
	59	
	60	

・
・
・

p++とは



ポインタが指し示すところ

ポインタは先頭アドレスからサイズ分だけ進めた領域のことを指している。

つまり p++ が指し示す領域とは、
変数の先頭アドレスからデータ型のサイズ分の領域の、次の領域である。

多次元配列

&arr[o] と arr は同じ意味である。したがって、
p = arr; p = &arr[o]; は同じである。

例) int arr[3][2]; int *p; とする

p = arr = arr[o] = &arr[o][o]

arr[1] = &arr[1][o]

arr[2] = &arr[2][o]

ポインタ	番地	変数
*p	50	arr[o][o]
*(p+1)	54	arr[o][1]
*(p+2)	58	arr[1][o]
*(p+3)	62	arr[1][1]
*(p+4)	66	arr[2][o]
*(p+5)	70	arr[2][1]

▪
▪
▪

確認クイズ

- 先ほどはint型でしたが、次はdouble型に書き変えてみましょう。

確認クイズの解説

C:\WINDOWS\system32\cmd.exe

```
pは、1703708
*pは、1.000000

*(p+1)を出力すると、5.000000
*p+1を出力すると、2.000000

今からpを一個ずつ足していきます。
1回目
pは、1703708
*pは、1.000000

2回目
pは、1703716
*pは、5.000000

3回目
pは、1703724
*pは、9.000000

4回目
pは、1703732
*pは、6.000000

5回目
pは、1703740
*pは、10.000000
```

ポインタ	番地	変数
p	50,51,52,53,54,55,56,57	arr[0]
p+1	58,59,60,61,62,63,64,65	arr[1]
p+2	66,67,68,69,70,71,72,73	arr[2]

Double型なので必要なメモリは8bit。

従って、p++すると、

先頭アドレス+8bitの領域

の

次の領域を指す

サンプル 2

```
1 #include<stdio.h>
2
3 int main(void){
4     int arr[10] = {1,2,3,4,5,6,7,8,9,0};
5     int *p = arr;
6     int i;
7
8     for(i=0; i<=9; i++){
9         printf("%d", *p);
10        p++;
11    }
12    printf("\n");
13
14    p--;
15    for(i=0; i<=9; i++){
16        printf("%d", *p);
17        p--;
18    }
19    printf("\n");
20
21    p+=7;
22    printf("%d\n", *p);
23
24    p-=4;
25    printf("%d\n", *p);
26
27    return 0;
28 }[EOF]
```

実行結果

```
1234567890
0987654321
7
3
```

ポインタと配列

- ポインタを進める時や戻すときはこのようにfor文やwhile文を使ってインクリメントやデクリメントすると便利である

- またある個数の分だけポインタを進めたいときは

`p += ____;`

- またある個数の分だけポインタを戻したいときは

`p -= ____;`

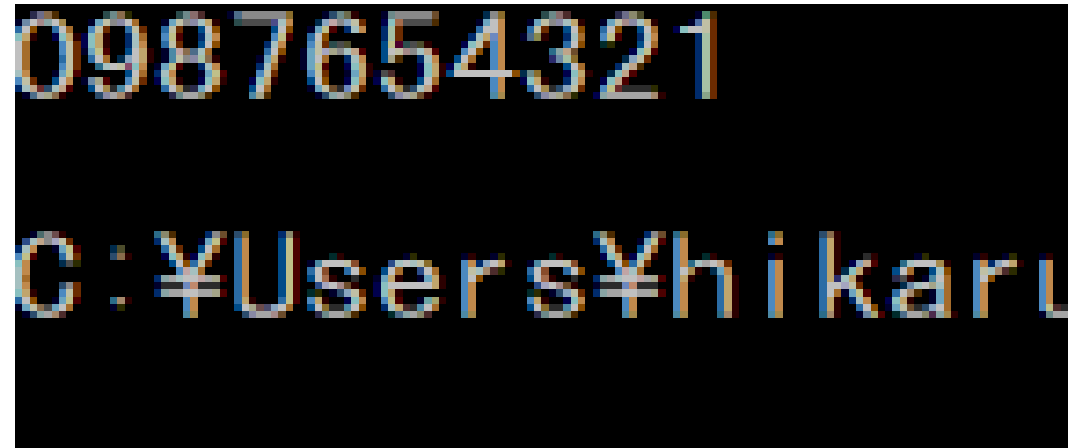
とすることができます

```
for(i=0; i<=9; i++){  
    printf("%d", *p);  
    p++;  
}
```

```
p+=7;  
printf("%d¥n", *p);  
  
p-=4;  
printf("%d¥n", *p);
```

練習問題 1

- 1234567890の10ケタの数字を逆向きに0987654321と出力するプログラム
- 使うもの
 - 配列 一個
 - ポインタ あえて二個



```
0987654321
C:\Users\hikaru
```

練習問題 2

- 整数を10個入力して、何番目の整数を表示したいか入力して、その数を入力するプログラム
- ポインタ使いましょう

```
数字を入力してください
543
634
643
24
976
4
75
534
8
433
何番目の数を入力しますか :
1
1番目の数は543
何番目の数を入力しますか :
7
7番目の数は75
何番目の数を入力しますか :
4
4番目の数は24
何番目の数を入力しますか :
6
6番目の数は4
何番目の数を入力しますか :
```

練習問題 3

- フィボナッチ数列の第 n 項をもとめるプログラム
- ポインタ使いましょう

```
何番目の項を表示しますか : 10  
55
```