# 平成 24 年度 アドバンスト・プログラミング
# (Advanced Programming)

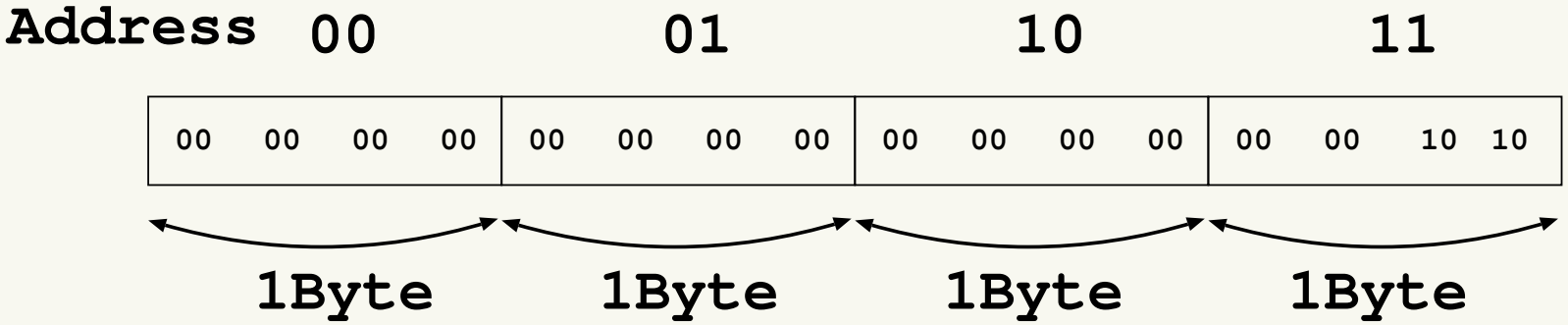
小澤　一文，陳　国躍，中村真輔

木曜 2 限 K323 教室

# 変数とメモリ

変数に値が代入されると，それを格納する場所が決められ，そこに値が格納される。

例えば，int a=10; とすれば，a のための (先頭) 番地が決められ




という形で 4Byte のメモリに格納される。メモリの番地を記憶しておく変数が必要になる。それが **ポインタ** である。

# ポインタについて (1)

&変数名 → 変数の値が格納されている番地 (アドレス演算子)
*番地 → その番地の値 (間接演算子)
値そのものでなく，番地を格納する変数を**ポインタ**と呼ぶ

```
#include <stdio.h>
main() {
   int i;
   int *j; /* 整数型変数へのポインタ */
   i=3;
   j=&i; /*  変数 i のアドレス  */
   printf(" i のアドレスは %p 番地 \n",j);
   printf(" i の値は %d \n",*j);
   printf(" i の値は %d \n",i);
}
```

実行結果

```
 i のアドレスは 0xbf82df68 番地
 i の値は 3
 i の値は 3
```

# ポインタについて (2)

```
#include <stdio.h>
main () {
   int i;
   int *j,*k;
   i=5;
   j=&i;
   k=&*j;
   printf("  i= %d \n",i);
   printf("  j= %p \n",j);
   printf(" *j= %d \n",*j);
   printf(" *k= %d \n",*k);
}
```

実行結果

```
   i= 5
   j= 0xbfe7f134
  *j= 5
  *k= 5
```

# ポインタについて (3)

```
#include <stdio.h>
main() {
   int i,j;
   int *a,*b;
   i=3; j=2;
   a=&i; b=&j;
   i=*b;
   j=*a;
   printf ("%d %d \n",i,j);
}
```

実行結果

```
2 2
```

# データを交換する関数

```
#include <stdio.h>
void swap(int a, int b);
main() {
  int a,b,c;
  scanf("%d",&a); scanf("%d",&b);
  printf(" a = %d, b = %d \n",a,b);
  swap(a,b);
  printf(" a = %d, b = %d \n",a,b);
}
void swap(int a, int b) {
  int c;
  c=a; a=b; b=c;
  return;
}
```

実行結果

```
2
3
 a = 2, b = 3
 a = 2, b = 3
```

# ポインタの応用 (1)

## ポインタを使ったデータの交換

```
#include <stdio.h>
main() {
   int a,b,c;
   scanf("%d",&a);
   scanf("%d",&b);
   printf(" a = %d, b = %d \n",a,b);
   c=a; a=b; b=c;
   printf(" a = %d, b = %d \n",a,b);
}
```

実行結果

```
2
3
 a = 2, b = 3
 a = 3, b = 2
```

# ポインタの応用 (2)

ポインタを使えばうまく行く！

```
#include <stdio.h>
void swap(int *x, int *y);
main() {
  int a,b,c;
  scanf("%d",&a);
  scanf("%d",&b);
  printf(" a = %d, b = %d \n",a,b);
  swap(&a,&b);
  printf(" a = %d, b = %d \n",a,b);
}
void swap(int *x, int *y) {
  int z;
  z=*x; *x=*y; *y=z;
  return;
}
```

**問題** 変数 a，b のうち大きい方を x，小さい方を y とする関数を書け。

# ポインタの応用 (3)

直交座標 $(x, y)$ を極座標 $(r, \theta)$ へ変換する関数

```
#include <stdio.h>
#include <math.h>
void polar(double x, double y, double *r, double *theta);
main() {
  double x,y,r,theta;
  scanf("%lf",&x); scanf("%lf",&y);
  polar(x,y,&r,&theta);
  printf("r = %lf, θ  = %lf\n",r,theta);
}
void polar(double x, double y, double *r, double *theta)
{
  *r=sqrt(x*x+y*y); *theta=atan(y/x);
  return;
}
```

実行例

```
1
1
r = 1.414214, θ  = 0.785398
```

# ポインタと配列 (1)

```
#include <stdio.h>
int main() {
  int i;
  int a[5];
  int *p;
  p=a;    /*  p は a[0] の先頭番地 &a[0]  */

  for (i=0; i<5; i++)
    printf(" &a[%d] = %p, p+%d = %p\n",i,
            &a[i],i,p+i);
}
```

実行結果

```
 &a[0] = 0xbf826980, p+0 = 0xbf826980
 &a[1] = 0xbf826984, p+1 = 0xbf826984
 &a[2] = 0xbf826988, p+2 = 0xbf826988
 &a[3] = 0xbf82698c, p+3 = 0xbf82698c
 &a[4] = 0xbf826990, p+4 = 0xbf826990
```

a は &a[0] を表すので，p+i と &a[i] は同じもの

# ポインタと配列 (2)

```
#include <stdio.h>
int main()
{
  int i,a[5];

  for (i=0; i<5; i++) a[i]=i;
  for (i=0; i<5; i++)
    printf(" a[%d] = %d, *(a+%d) = %d\n",i,a[i],i,*(a+i));
}
 a[0] = 0, *(a+0) = 0
 a[1] = 1, *(a+1) = 1
 a[2] = 2, *(a+2) = 2
 a[3] = 3, *(a+3) = 3
 a[4] = 4, *(a+4) = 4
```

# ポインタと配列 (3)

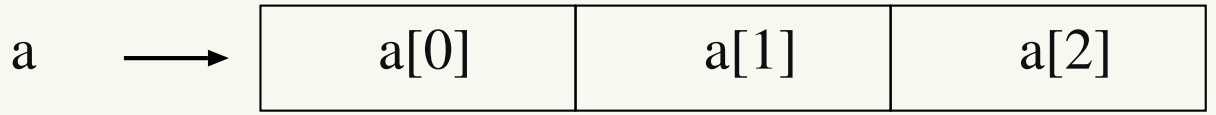
```
#include <stdio.h>
#include <stdlib.h>
#define n 100000

void max_min(int a[], int *max, int *min, int N);

main()
{
  int a[n],max,min,i;

  srand(51);
  for (i=0; i<n; i++) a[i]=rand();

  max_min(a, &max, &min, n);
  printf(" max of a[] = %10d, min of a[] = %10d
                  \n",max,min);
}
```

# ポインタと配列 (3)(続き)

```
void max_min(int a[], int *max, int *min, int N)
{
  int i;
  *min=a[0]; *max=a[0];
  for (i=1; i<N; i++) {
    if ( a[i] > *max ) *max=a[i];
    if ( a[i] < *min ) *min=a[i];
  }
  return;
}
```

**問題** このプログラムに平均を計算する部分を付け加えよ。

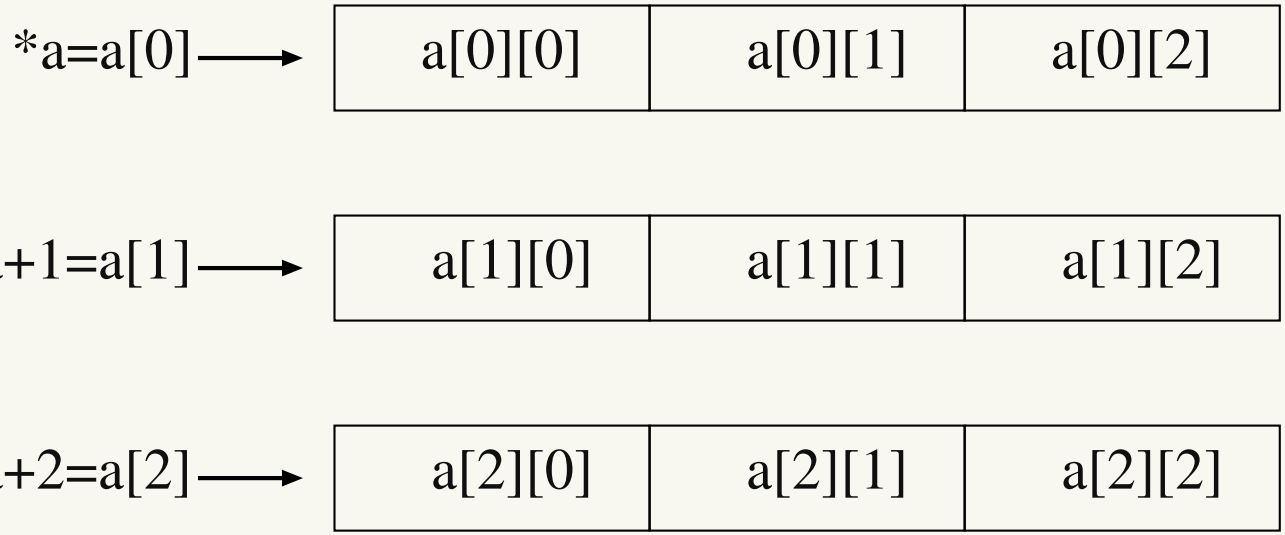
# 一次元配列とポインタ




int a[3] を宣言すると

`a` で配列 `a[0]` の先頭アドレスを表す約束なので，`*a` は `a[0]` を表し，`a[i]` は `*(a+i)` を表していることになる。

# 二次元配列とポインタ




`int a[3][3]` を宣言したとき

例えば a[0][0] の先頭番地を 0 とすると

a[0][0] の先頭番地= 0，a[0][1] の先頭番地= 4，a[0][2] の先頭番地= 8，
a[1][0] の先頭番地=12，a[1][1] の先頭番地=16，a[1][2] の先頭番地=20，
a[2][0] の先頭番地=24，a[2][1] の先頭番地=28，a[2][2] の先頭番地=32

となる。したがって，a[0]=0, a[1]= 12, a[2] = 24 となる。

`a[i][j]` は `*(a[i]+j)=*(*(a+i)+j)` を表している。