Intermediate C Programming

Lesson5

Pointers and Matrices

Today's outline

- Pointers and Matrices
- malloc()
- sturctures
- Exercise

Last Week

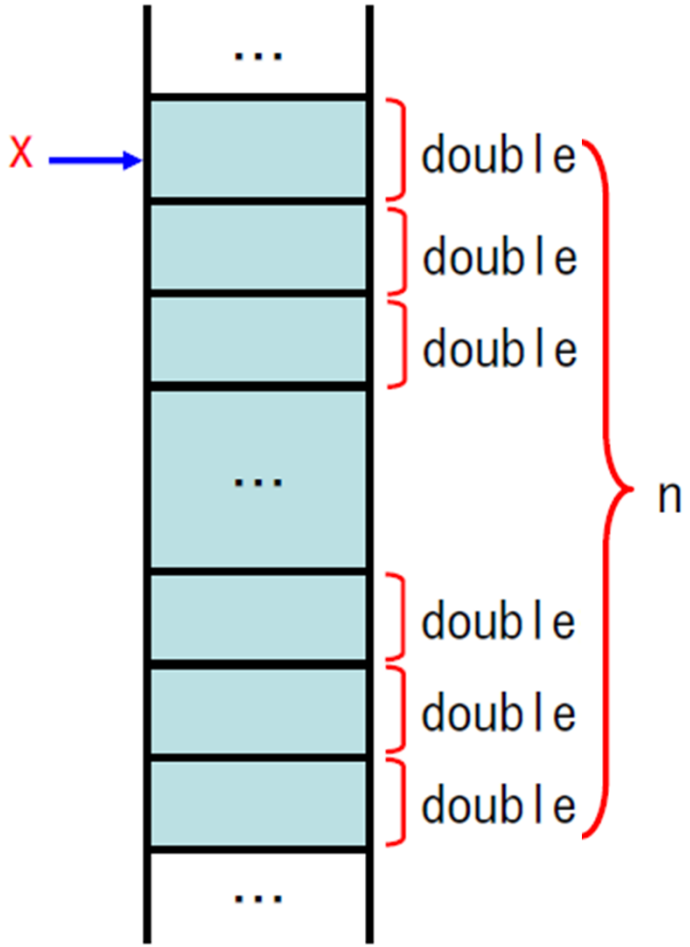
```
#include <stdio.h>


int main(void)
{
    int i, j;
    double x[3] = {-33.0, 9.0, 6.0};
    double y[3];
    double a[3][3] = {{2.0, 4.0, 6.0},
                      {3.0, 8.0, 7.0},
                     {5.0, 7.0, 21.0}};
    double *ai;
```

```
    ai = a;
      for (i=0; i<3; i++) {
          *(y + i) = 0.0;
          for (j=0; j<3; j++) {
              *(y + i) += *(ai + j) * *(x + j);
          }
          ai += 3;
      }


      printf("y = ¥n");
      for (i=0; i<3; i++) {
          printf("%6.2f¥n", *(y + i));
      }


      return 0;
  }
```

malloc(size)

```
double *x;
x = (double *)malloc(n*sizeof(double));
```

```
sizeof(double) → 8 byte
sizeof(int)    → 4 byte
sizeof(char)   → 1 byte
```



malloc(size)

- x

```
x = (double *)malloc(n*sizeof(double));
```

- (double)

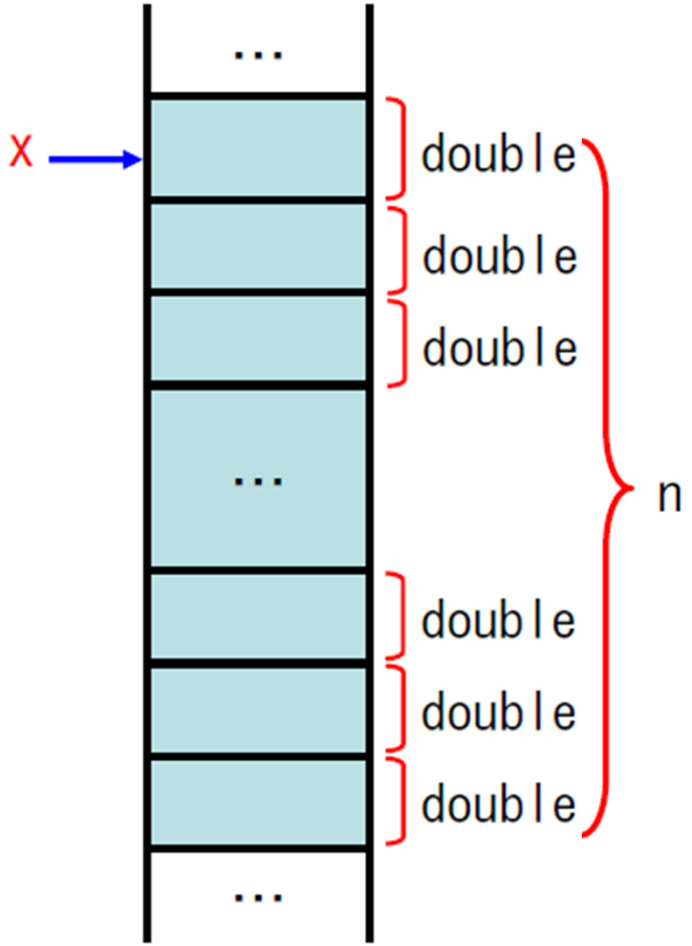
```
int a, b;
double c, d;

a = 1.0;
b = 3.0;
c = a / b;          →  0.000000
d = (double)a / b;  →  0.333333
```

malloc(size)

```
double *x;
x = (double *)malloc(n*sizeof(double));
```

```
x + i
*(x + i)
  ↕
 x[i]
```

```
scanf("%lf", &x[i]);      →
printf("%.2f¥n", x[i]);   →
```



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

int main(void)
{
    int i, n;
    double *x;


    printf("  n : ");
    scanf("%d", &n);


    x = (double
*)malloc(n*sizeof(double));
    if (x==NULL) {
        printf("Can't allocate
memory.¥n");
        exit(1);
    }
```

```
for (i=0; i<n; i++) {
        printf("  x[%d] = ", i);
        scanf("%lf", x + i);
    }


    printf("¥n  x =¥n");
    for (i=0; i<n; i++) {
        printf("    %.2f¥n", *(x + i));
    }


    free(x);


    return 0;
}
```

malloc(size)

```
double *a, *ai;

a = (double *)malloc(n*n*sizeof(double));
if (a==NULL) {
  printf("Can't allocate memory.¥n");
  exit(1);
}
```

- Poniter ai

```
ai = a;
```

- Change line

```
ai += n;
```

malloc(size)

```
double *a, *ai;

a = (double *)malloc(n*n*sizeof(double));
if (a==NULL) {
  printf("Can't allocate memory.¥n");
  exit(1);
}
ai = a;
for (i=0; i<n; i++) {
  for(j=0; j<n; j++) {
    printf("a[%d][%d] = ", i, j);
    scanf("%lf", &ai[j]);
  }
  ai += n;
}
```

Exercise

- For

$$A=\begin{pmatrix}2 & 4 & 6\\ 3 & 8 & 7\\ 5 & 7 & 21\end{pmatrix},\quad x=\begin{pmatrix}-33\\ 9\\ 6\end{pmatrix}$$

- How to write the program to calculate $(A\cdot x)$ with pointers and malloc(size)?

```
$ ./a.out↵
n = 3↵
a[0][0] = 2.0↵
a[0][1] = 4.0↵
a[0][2] = 6.0↵
a[1][0] = 3.0↵
...
x[2] = 6.0↵

A*x =
   6.00
  15.00
  24.00
```

Homework

- For

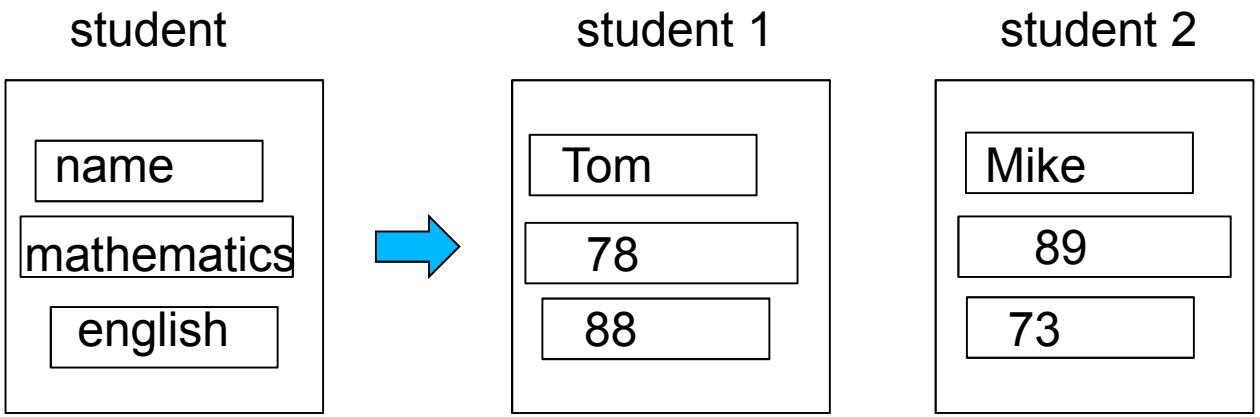
$$A=\begin{pmatrix}3 & -1 & 2\\ 1 & 2 & 3\\ 2 & -2 & -1\end{pmatrix},\quad B=\begin{pmatrix}8 & 1 & -1\\ -1 & 7 & -2\\ 2 & 1 & 9\end{pmatrix}$$

- How to write the program to calculate A*B with pointers and malloc(size)?

Structures

- A structure is a collection of one or more variables, possibly of different types, grouped together under a single name for convenient handling

- For example, student record



Structures

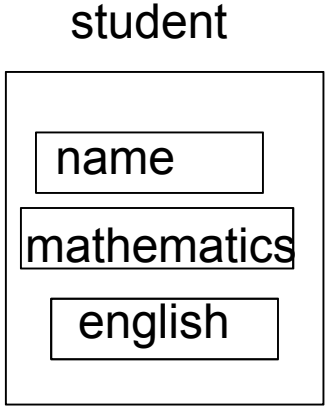
```
struct structure-tag{
 member1;
 member2;
};
```

An optional name called a structure-tag may follow the word struct

The variables named in a structure are called members

- For example, student record

```
struct student {
    char name[20];
    int eng;
    int math;
};
```



Structures

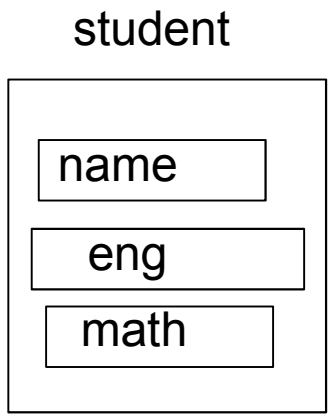
A struct declaration defines a type.

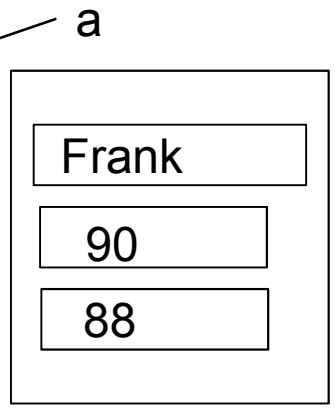
```
#include <stdio.h>
struct student {
    char name[20];
    int eng;
    int math;
};

int main(void)
{
    struct student a, b, *pa;

    ...
}
```

Structures

A struct declaration defines a type.

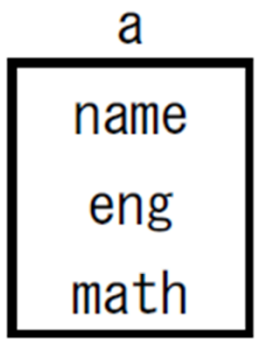
```
#include <stdio.h>
#include <string.h>
struct student {
    char name[20];
    int eng;
    int math;
};

int main(void)
{
    struct student a, b, *pa;

    strcpy(a.name, "Frank");
    a.eng = 90;
    a.math = 88;
    ...
}
```





Structures

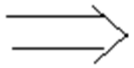
```
struct student a, b, *pa;
```




```
pa = &a;
strcpy(pa->name, "Thomas");
pa->eng = 85;
pa->math = 94;
```

Structures

```
#include <stdio.h>
#include <string.h>

struct student {
char name[20];
int eng;
int math;
};

int main(void)
{
struct student a, b, *pa;
strcpy(a.name, "Frank");
a.eng = 90;
a.math = 88;
printf(" Name:%s\n", a.name);
printf(" English:%d\n", a.eng);
printf("Mathematics:%d\n\n", a.math);
```



```
b=a;
printf(" Name:%s\n", b.name);
printf(" English:%d\n", b.eng);
printf("Mathematics:%d\n\n", b.math);

pa = &a;
strcpy(pa->name, "Thomas");
pa->eng = 85;
pa->math = 94;
printf(" Name:%s\n", pa->name);
printf(" English:%d\n", pa->eng);
printf("Mathematics:%d\n\n", pa->math);
return 0;
}
```