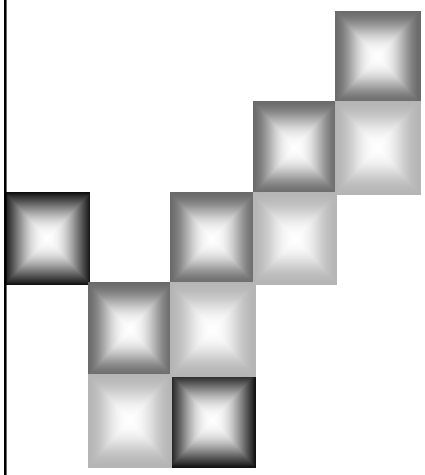


KHÁI NIỆM KDL- CẤU TRÚC DL

Nội dung

- Định nghĩa Kiểu dữ liệu
- Các kiểu dữ liệu cơ bản
- Các kiểu dữ liệu có cấu trúc
- Một số kiểu dữ liệu có cấu trúc cơ bản
- Mảng
- Chuỗi ký tự
- Struct

Định nghĩa kiểu dữ liệu

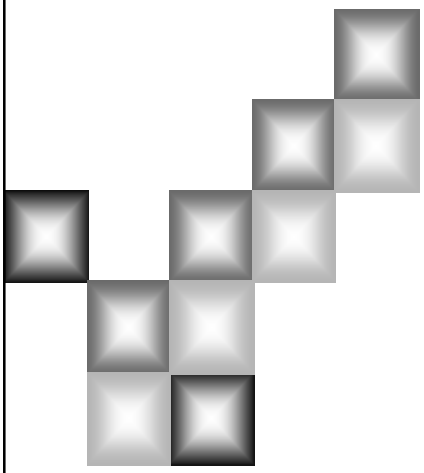
Kiểu dữ liệu **T** được xác định bởi một bộ <**V**,**O**> với:

- **V**: tập các giá trị hợp lệ mà một đối tượng kiểu **T** có thể lưu trữ.
- **O**: tập các thao tác xử lý có thể thi hành trên đối tượng kiểu **T.**

Ví dụ: kiểu dữ liệu số nguyên= <V_i, O_i> với

V_i={-32768.. 32767}; O_i={+, -, *, /, %}.

Như vậy, muốn sử dụng một KDL cần nắm vững cả nội dung DL đươc phép lưu trữ và các xử lý tác động trên nó. Các thuộc tính 1 KDL gồm(Tên KDL, Miền giá trị, Kích thước lưu trữ, tập các toán tử tác động lên KDL).



Kiểu dữ liệu cơ sở trong C

Các kiểu dữ liệu cơ sở

- **Kiểu có thứ tự rời rạc**: Số nguyên.
- **Kiểu có thứ tự không rời rạc**: Số thực.

Các kiểu số nguyên của C

- C hỗ trợ khá nhiều kiểu số nguyên
- Các giá trị lớn nhất và nhỏ nhất được định nghĩa trong thư viện "`limits.h`"

Kiểu	định dạng	kích thước	nhỏ nhất	lớn nhất
char	%c	1	CHAR_MIN	CHAR_MAX
unsigned char	%c	1	0	UCHAR_MAX
short [int]	%hi	2	SHRT_MIN	SHRT_MAX
unsigned short	%hu	2	0	USHRT_MAX
int	%i	**2 or 4**	INT_MIN	INT_MAX
unsigned int	%u	**2 or 4**	0	UINT_MAX
long [int]	%li	4	LONG_MIN	LONG_MAX
unsigned long	%lu	4	0	ULONG_MAX

Ví dụ về số nguyên

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

int  main()
{
     unsigned long   big = ULONG_MAX;

     printf("minimum int = %i, ", INT_MIN);
     printf("maximum int = %i\n", INT_MAX);
     printf("maximum unsigned = %u\n", UINT_MAX);
     printf("maximum long int = %li\n", LONG_MAX);
     printf("maximum unsigned long = %lu\n", big);

     return 0;
}
```

```
minimum int = -32768, maximum int = 32767
maximum unsigned = 65535
maximum long int = 2147483647
maximum unsigned long = 4294967295
```

Ví dụ kiểu ký tự

In ra mã ASCII của ký tự

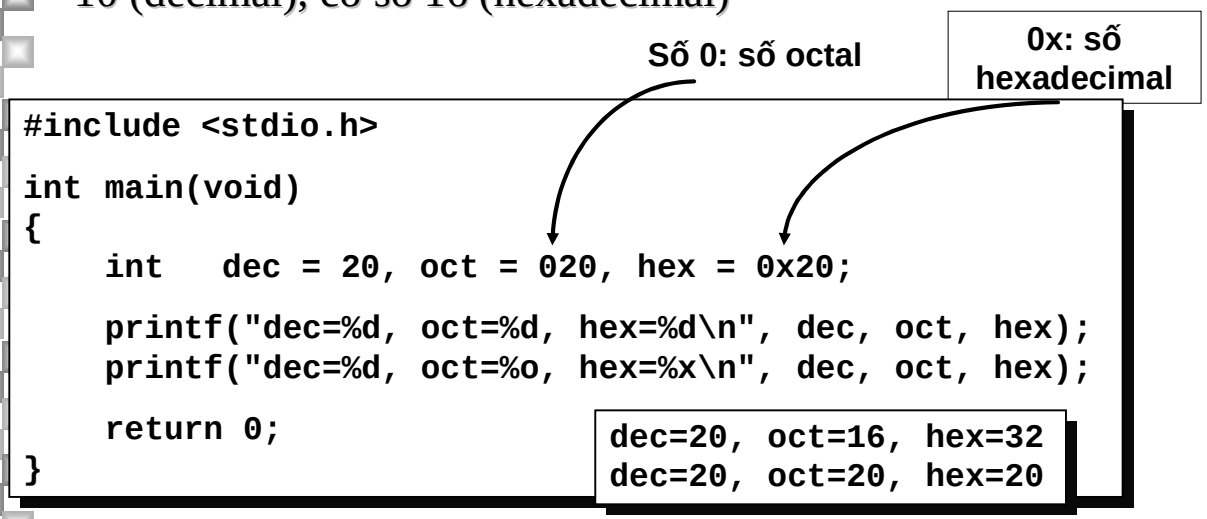
```
#include <stdio.h>
#include <limits.h>

int main()
{
    char  lower_a = 'a';
    char  lower_m = 'm';

    printf("minimum char = %i, ", CHAR_MIN);
    printf("maximum char = %i\n", CHAR_MAX);

    printf("Sau '%c' la '%c'\n", lower_a, lower_a + 1);
    printf("Ky tu in hoa '%c'\n", lower_m - 'a' + 'A');

    return 0;
}
```

Trong NNLT C, ký tự chính là số nguyên

```
minimum char = -128, maximum char = 127
Sau 'a' la 'b'
Ky tu in hoa 'M'
```

Số nguyên trong các cơ số khác

- Các hệ cơ số có thể thực hiện được: cơ số 8 (octal), cơ số 10 (decimal), cơ số 16 (hexadecimal)



Số thực

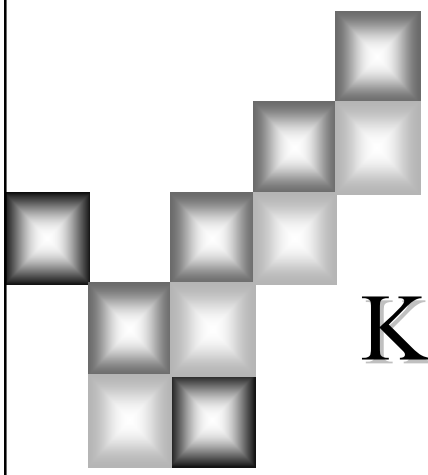
- C hỗ trợ nhiều kiểu số thực lưu trữ dấu chấm động.
- Các giá trị lớn nhất và nhỏ nhất được định nghĩa trong thư viện "`float.h`"

Kiểu	định dạng	kích thước	nhỏ nhất	lớn nhất
`float`	`%f %e %g`	`4`	`FLT_MIN`	`FLT_MAX`
`double`	`%lf %le %lg`	`8`	`DBL_MIN`	`DBL_MAX`
`long double`	`%Lf %Le %Lg`	`10`	`LDBL_MIN`	`LDBL_MAX`

Ví dụ số thực:

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

int  main(void)
{
     double f = 3.1416, g = 1.2e-5, h = 5000000000.0;

     printf("f=%lf\tg=%lf\th=%lf\n", f, g, h);
     printf("f=%le\tg=%le\th=%le\n", f, g, h);
     printf("f=%lg\tg=%lg\th=%lg\n", f, g, h);

     printf("f=%7.2lf\tg=%.2le\th=%.4lg\n", f, g, h);

     return 0;
}
```

```
f=3.141600          g=0.000012          h=5000000000.000000
f=3.141600e+00      g=1.200000e-05      h=5.000000e+09
f=3.1416            g=1.2e-05           h=5e+09
f=   3.14           g=1.20e-05          h=5e+09
```



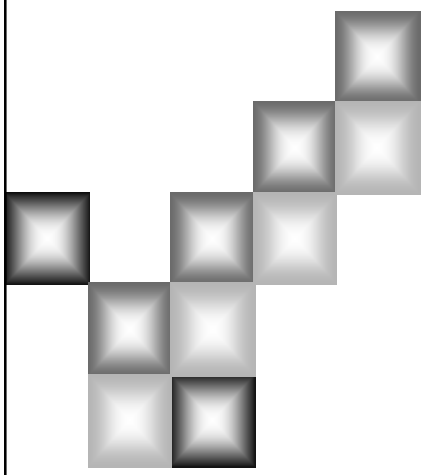
Kiểu dữ liệu có cấu trúc

Định nghĩa kiểu dữ liệu có cấu trúc

Kiểu dữ liệu được xây dựng mới dựa trên việc tổ chức liên kết các thành phần dữ liệu có kiểu dữ liệu đã được định nghĩa. Những KDL được xây dựng như thế gọi là KDL có cấu trúc.

Một số kiểu dữ liệu cấu trúc

- Kiểu mảng
- Kiểu chuỗi ký tự
- Kiểu cấu trúc mẩu tin



Mảng - Array

Mảng – Array

- Một số tính chất
- Khai báo mảng trong C
- Truy xuất các thành phần
- Truyền tham số kiểu mảng cho hàm
- Một số thao tác cơ sở
- Mảng nhiều chiều

Mảng – Một số tính chất

- Mảng là một kiểu dữ liệu có cấu trúc do người lập trình định nghĩa
- Dùng biểu diễn các đối tượng dữ liệu ở dạng một dãy các thành phần có cùng kiểu với nhau – kiểu cơ sở
- NNLT C luôn chỉ định một khối nhớ liên tục cho một biến kiểu mảng
- Kích thước của mảng được xác định ngay khi khai báo và không bao giờ thay đổi

Mảng – Khai báo trong C

```
typedef kiểucơsở Tênkiểu[Sốthànhphần];
```


```
typedef    int    AINT[100];
```

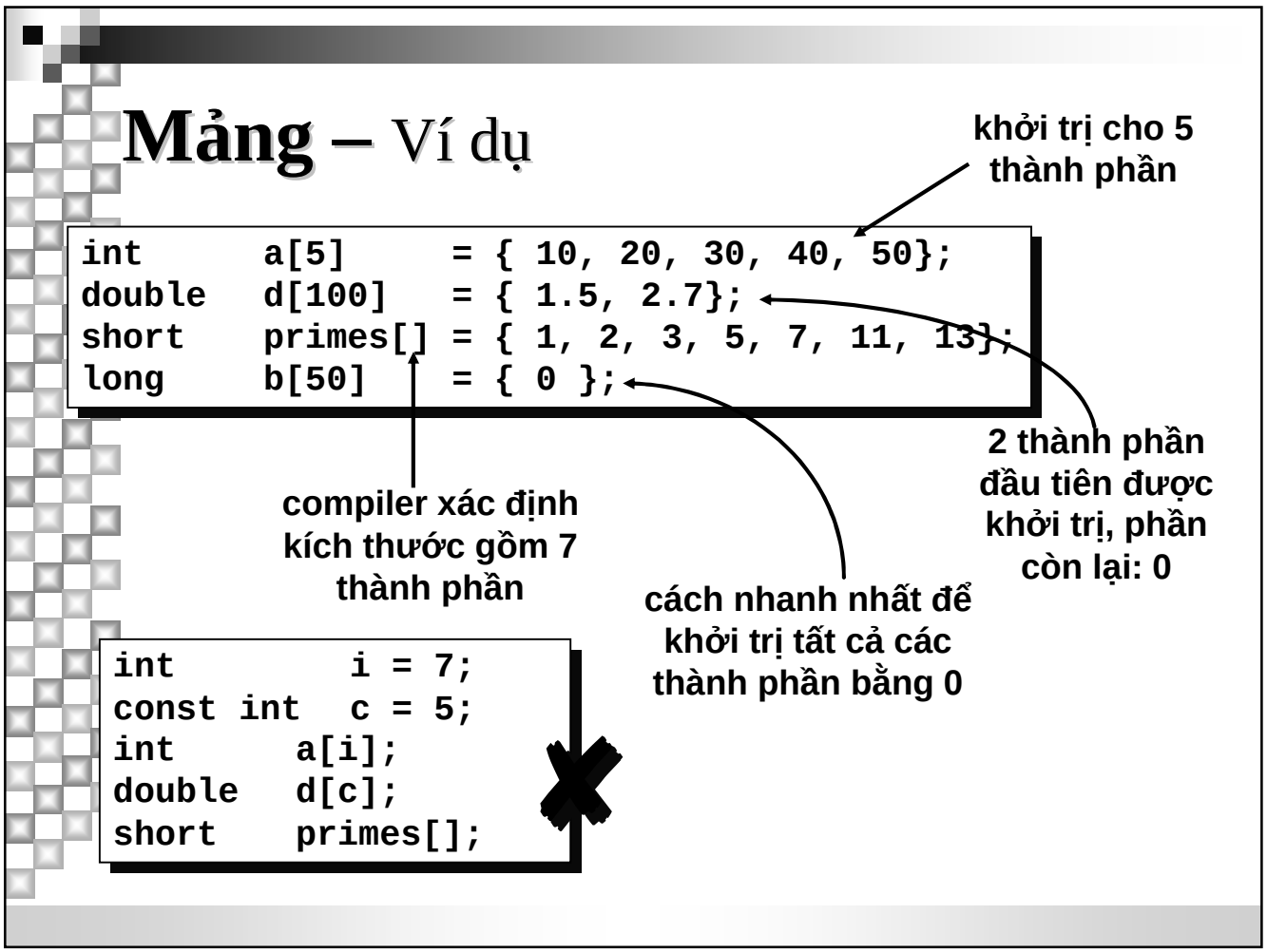
//AINT là kiểu mảng biểu diễn dãy gồm 100 thành phần int

```
AINT   a;   //a: biến kiểu AINT
```

Mảng – Ví dụ

```
#define  SIZE     10
int      a[5];      // a dãy gồm 5 số nguyên
long int big[100];// big: chiếm 400 bytes!
double   d[100];   // d: chiếm 800 bytes!
long double  v[SIZE];// v:10 long doubles
```

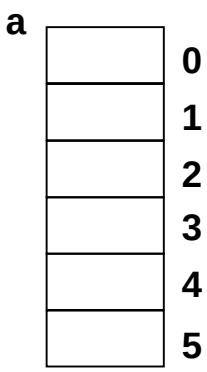
Mảng – Ví dụ



Mảng – Truy xuất các phần tử

- Các thành phần của mảng được truy xuất thông qua chỉ số của chúng 0..size-1
- Thao tác truy xuất không kiểm tra giới hạn của chỉ số

```
int main()
{
    int   a[6];
    int   i = 7;
    a[0] = 59;
    a[5] = -10;
    a[i/2] = 2;
    a[6] = 0;
    a[-1] = 5;
    return 0;
}
```



Truyền tham số Mảng cho hàm

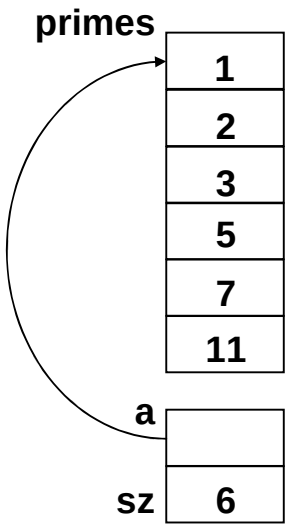
- Tham số kiểu mảng được truyền cho hàm chính là địa chỉ của phần tử đầu tiên trên mảng
- Số thành phần trong tham số mảng có thể để trống.
- Số thành phần thực sự được sử dụng phải truyền qua một tham số khác (vd: size)

```
int add_elements(int a[], int size)
{
```

```
int add_elements(int *p, int size)
{
```

Ví dụ

```
#include <stdio.h>
void sum(long [], int);
int  main(void) {
     long  primes[6] = { 1, 2,
                3, 5, 7, 11 };
     sum(primes, 6);
     printf("%li\n", primes[0]);
     return 0;
}

void sum(long a[], int sz) {
     int   i;
     long  total = 0;
     for(i = 0; i < sz; i++)
          total += a[i];
     a[0] = total;
}
```



dùng để kiểm tra giới hạn chỉ số

tổng được lưu vào phần tử đầu tiên

Một số thao tác cơ sở

- Nhập
- Xuất
- Thêm một thành phần dữ liệu
- Loại bỏ một thành phần dữ liệu
- Tìm kiếm
- Sắp xếp

Mảng – Nhập dữ liệu

```
void ReadData(int a[], int size)
{
      int i;

      for(i = 0; i < size; i++)
      {
          printf("Nhap thanh phan %d: ", i);
          scanf("%d", &a[i]);
      }
}
```

duyệt qua tất cả các phần tử

nhập dữ liệu cho a[i]

Mảng – Xuất dữ liệu ra màn hình

```
void WriteData(int a[], int size)
{
      int i;

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

Mảng – Nhập xuất dữ liệu

```
#include <stdio.h>
void  ReadData(int [], int );
void  WriteData(int [], int );
void  main()
{
      int a[100], n;
      clrscr();
      printf("Nhap so thanh phan cua day: ");
      scanf("%d", &n);
      printf("Nhap cac thanh phan cua day: ");
      ReadData(a, n);
      printf("Day vua nhap: \n");
      WriteData(a, n);
}
```

Mảng – Tìm vị trí X trong dãy

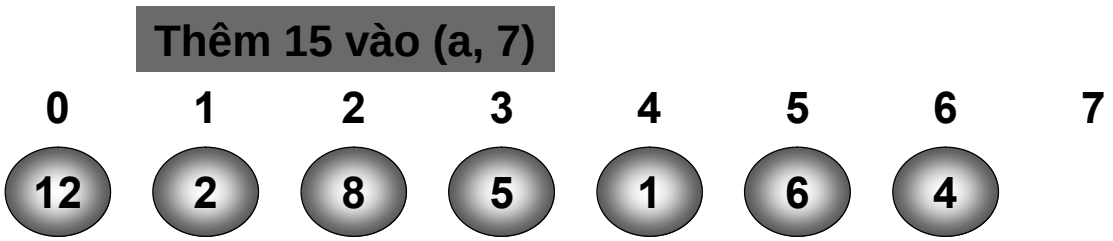
- Bài toán: Tìm vị trí X trên mảng a đang có N thành phần.
- Giải pháp: Tìm tuần tự

```
//input: dãy (a, N), X
//output: Vị trí của X, -1 nếu không có

int     Search(int a[], int N, int X)
{
   for (int i = 0; i < N; i ++)
         if (a[i] == X)
               return i;
   return -1;
}
```

Mảng – Thêm một thành phần dữ liệu

- Bài toán: cần thêm thành phần dữ liệu X vào mảng a đang có N thành phần.
- Hai trường hợp cần xem xét:
  - Dãy chưa có thứ tự
    - → Thêm X vào cuối a.
  - Dãy đã có thứ tự
    - → Tìm vị trí thích hợp, chèn X vào

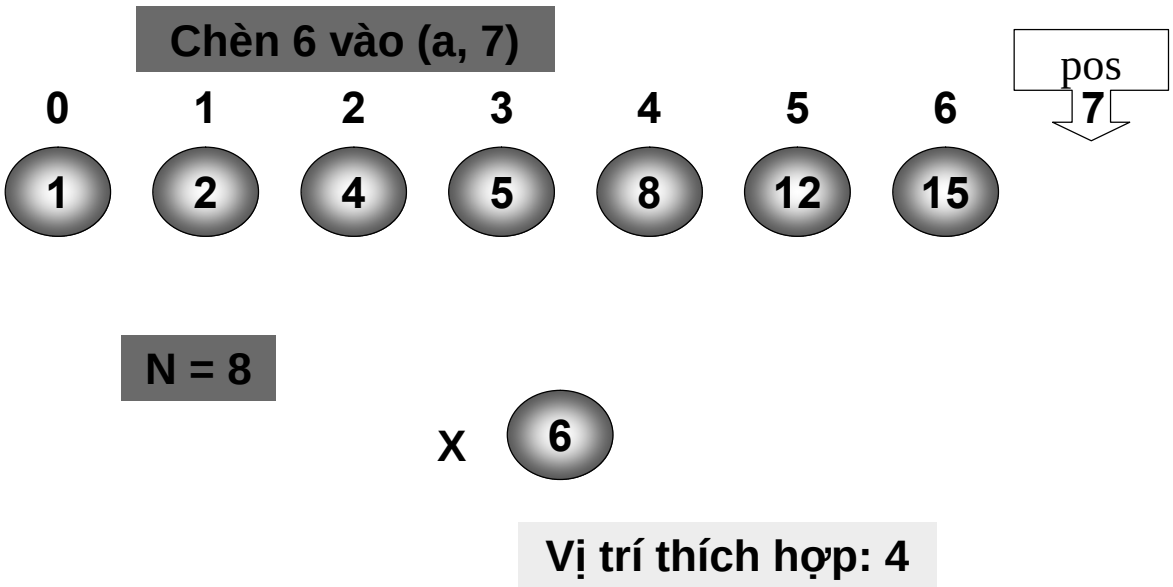
Mảng – Thêm X vào cuối dãy



N = 8

```
a[N] = X;
N ++;
```

Mảng – Chèn X vào dãy tăng dần



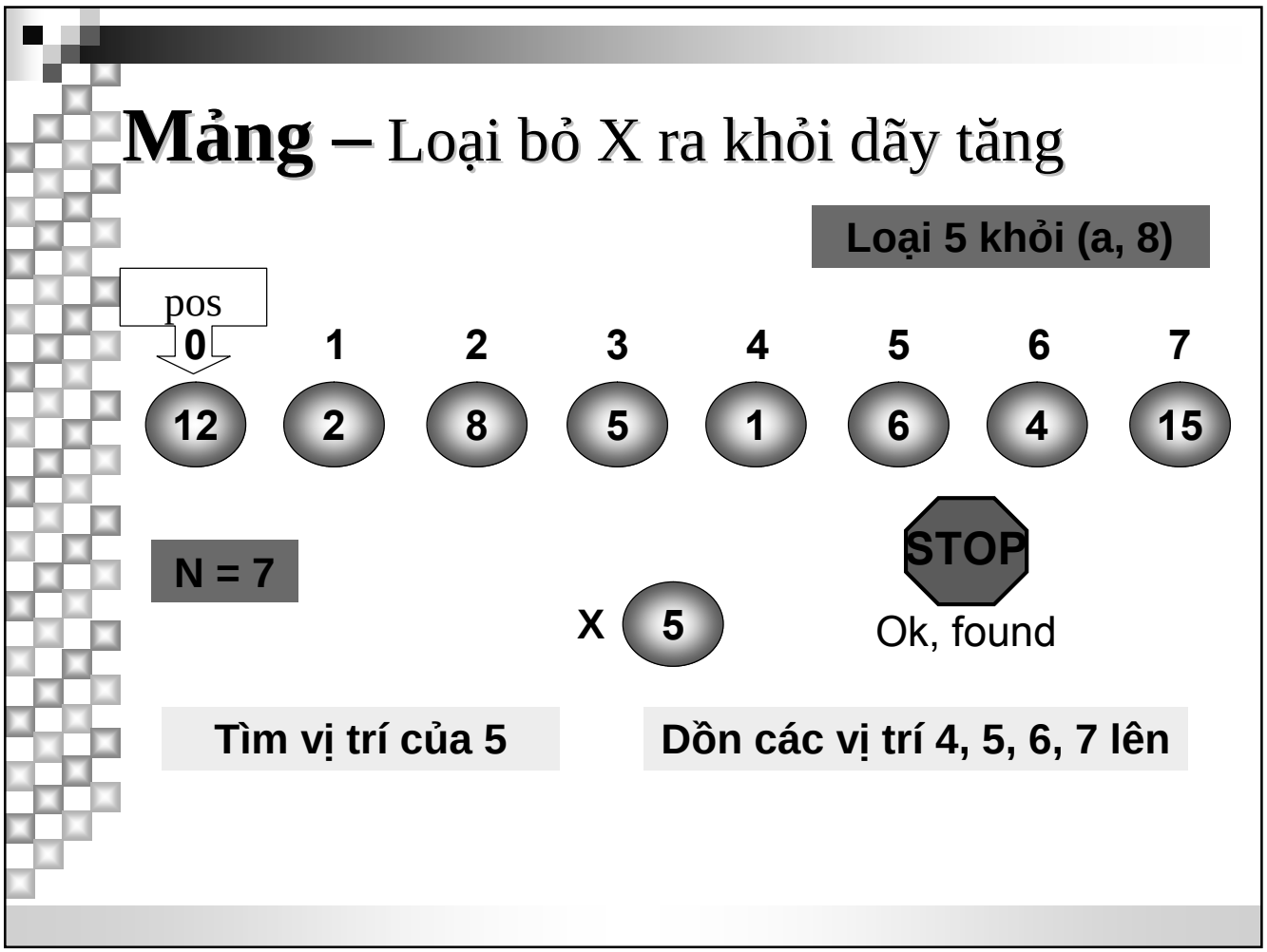
Mảng – Chèn X vào dãy tăng dần

```
//input: dãy (a, N) tăng dần, X
//output: dãy (a, N) đã có X ở đúng vị trí

void    Insert(int a[], int &N, int X)
{
   int pos;

   for (pos = N; (pos>0)&&(a[pos-1]>X); pos --)
         a[pos] = a[pos – 1];
   a[pos] = X;
   N ++;
}
```

Mảng – Loại bỏ một thành phần dữ liệu

- Bài toán: loại bỏ thành phần dữ liệu X ra khỏi mảng a đang có N thành phần.
- Hướng giải quyết: xác định vị trí của X, nếu tìm thấy thì dồn các phần tử ở phía sau lên để lấp vào chỗ trống. 2 trường hợp:
  - Dãy không có thứ tự: lấp phần tử cuối lên
  - Dãy đã thứ tự: dời tất cả các phần tử ở sau ví trí của X lên trước 1 vị trí.

Mảng – Loại bỏ X ra khỏi dãy tăng



Mảng – Loại bỏ X ra khỏi dãy tăng

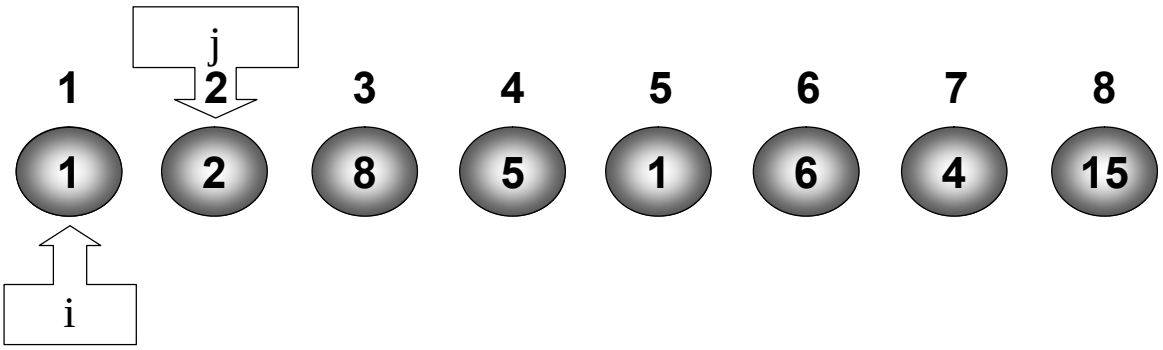
```
//input: dãy (a, N), X
//output: dãy (a, N) đã loại bỏ 1 thành phần X

int    Remove(int a[], int &N, int X)
{
   int pos = Search(a, N, X);
   if (pos == -1) //không có X trong dãy
        return 0;
   N --;
   for (; (pos < N); pos ++)
        a[pos] = a[pos + 1];
   return 1;
}
```

Mảng – Sắp xếp

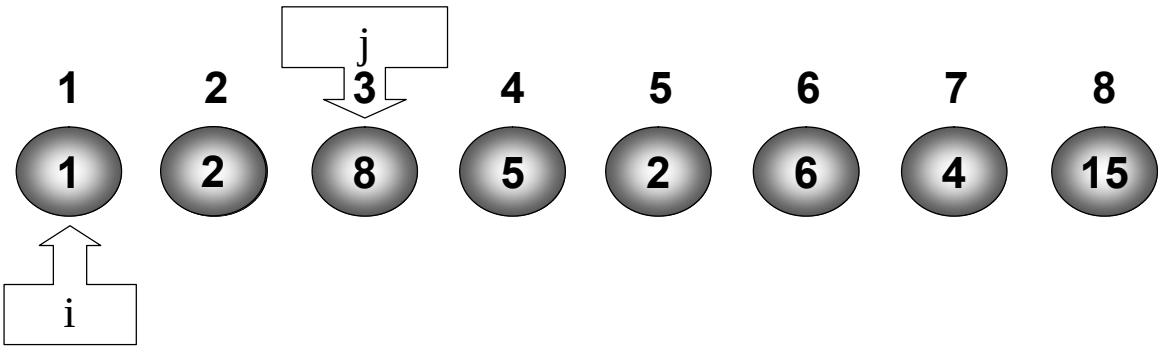
- Bài toán: Sắp xếp các thành phần của (a, N) để thu được dãy tăng dần
- Giải pháp: Tìm cách triệt tiêu tất cả các nghịch thế của dãy

  → Thuật toán sắp xếp Đổi chỗ trực tiếp

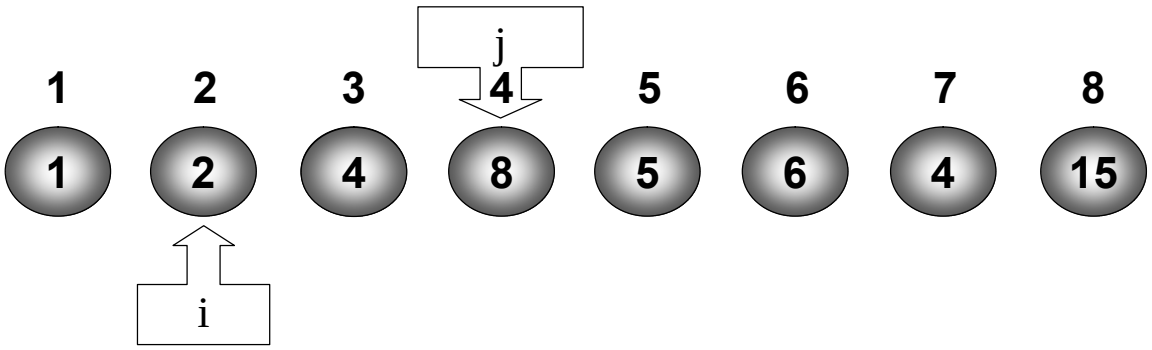
Mảng – Sắp xếp đổi chỗ



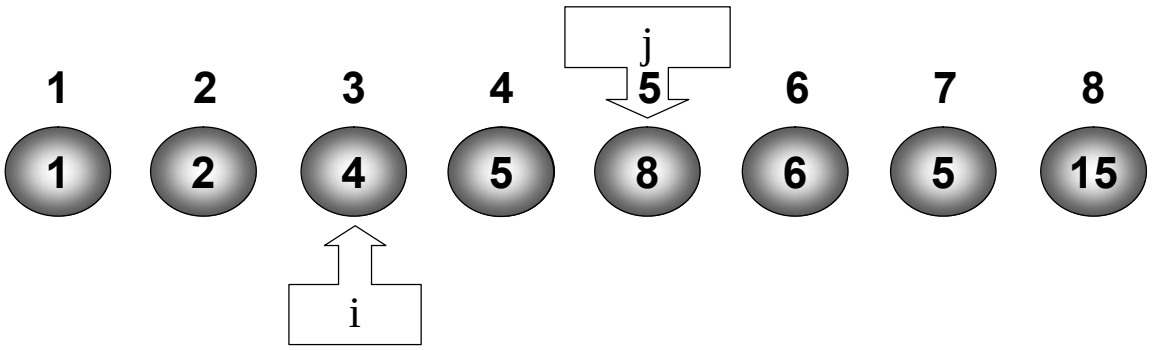
Mảng – Sắp xếp đổi chỗ



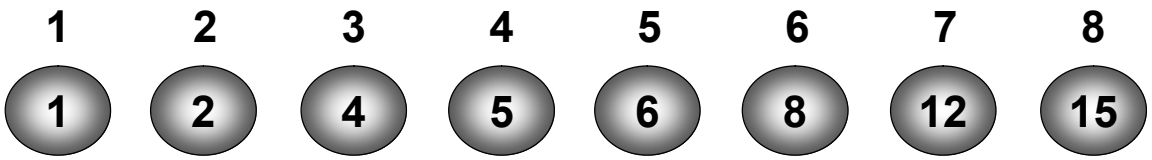
Mảng – Sắp xếp đổi chỗ



Mảng – Sắp xếp đổi chỗ



Mảng – Sắp xếp đổi chỗ



Mảng – Sắp xếp đổi chỗ

```
void Swap(int &x, int &y)
{
   int t = x; x = y; y = t;
}

void InterchangeSort(int a[], int N)
{
   int  i, j;
   for (i = 0 ; i<N-1 ; i++)
       for (j =i+1; j < N ; j++)
          if(a[j]< a[i])
             Swap(a[i],a[j]);
}
```

Mảng nhiều chiều

- C không hỗ trợ mảng nhiều chiều. Tuy nhiên có thể tiếp cận theo hướng: Mảng 2 chiều là mảng một chiều mà mỗi thành phần của nó là một mảng một chiều.

```
float   rainfall[12][365];
```

"rainfall" là mảng gồm 12 thành phần, mỗi thành phần là mảng gồm 365 số float

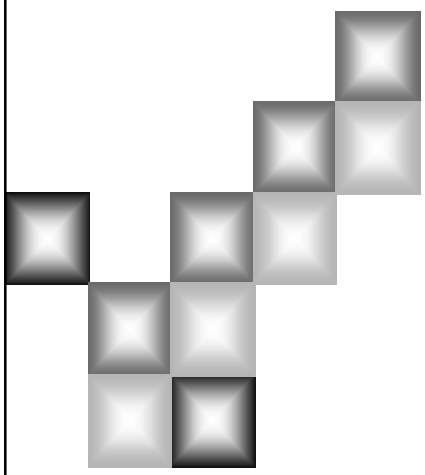
```
short   exam_marks[500][10];
```

"exam_marks" là mảng gồm 500 thành phần, mỗi thành phần là mảng 10 số short

```
const int brighton = 7;
int day_of_year = 238;

rainfall[brighton][day_of_year] = 0.0F;
```

Tóm lược

- Khai báo mảng trong C
- Truy xuất các phần tử
- Truyền tham số kiểu mảng cho hàm
- Các thao tác: nhập, xuất, thêm/hủy 1 thành phần, tìm kiếm, sắp xếp
- Mảng nhiều chiều



Chuỗi ký tự - String

Chuỗi ký tự – Strings

- Một số qui tắc
- Nhập / xuất
- Con trỏ và chuỗi ký tự
- Một số hàm thư viện

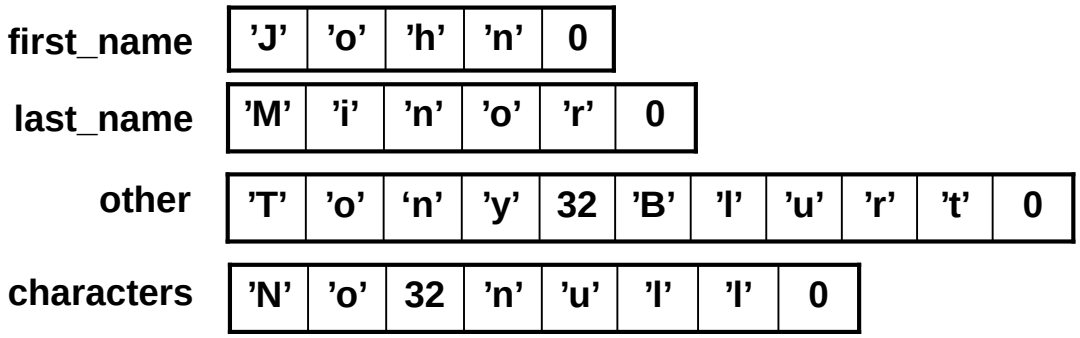
Chuỗi ký tự - Một số qui tắc

- Chuỗi ký tự là mảng một chiều có mỗi thành phần là một số nguyên được kết thúc bởi số 0.
- Ký tự kết thúc (0) ở cuối chuỗi ký tự thường được gọi là ký tự null (không giống con trỏ NULL). Có thể ghi là 0 hoặc '\0' (không phải chữ o).
- Được khai báo và truyền tham số như mảng một chiều.

```
char           s[100];
unsigned char  s1[1000];
```

Chuỗi ký tự - Ví dụ

```
char first_name[5] = { 'J', 'o', 'h', 'n', '\0' };

char last_name[6]  = "Minor";

char other[]       = "Tony Blurt";

char characters[7] = "No null";
```



Chuỗi ký tự - Nhập / xuất

- Có thể nhập / xuất chuỗi ký tự s bằng cách nhập từng ký tự của s
- Hoặc sử dụng các hàm `scanf` và `printf` với ký tự định dạng "`%s`"

```
char  other[] = "Tony Blurt";
printf("%s\n", other);
```

- Nhập chuỗi có khoảng trắng dùng hàm gets

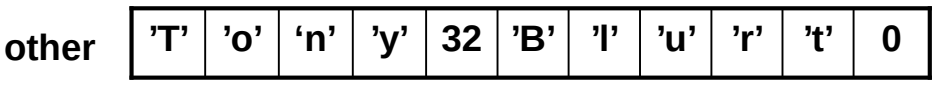
```
char  name[100];
printf("Nhap mot chuoi ky tu %s: ");
gets(name);
```

Lưu ý: kết thúc chuỗi

```
#include <stdio.h>

int  main()
{
     char other[] = "Tony Blurt";

     printf("%s\n", other);

     other[4] = '\0';

     printf("%s\n", other);

     return 0;
}
```

"Blurt" sẽ không được in ra

```
Tony Blurt
Tony
```



Chuỗi ký tự – Một số hàm thư viện

- Lấy độ dài chuỗi

```
l = strlen(s);
```

- Đổi toàn bộ các ký tự của chuỗi thành IN HOA

```
strupr(s);
```

- Đổi toàn bộ các ký tự của chuỗi thành **in thường**

```
strlwr(s);
```

Chuỗi ký tự – Một số hàm thư viện

- So sánh chuỗi: so sánh theo thứ tự từ điển

Phân biệt IN HOA – in thường:

```
int strcmp(const char *s1, const char *s2);
```

Không phân biệt IN HOA – in thường:

```
int stricmp(const char *s1, const char *s2);
```

Chuỗi ký tự – ví dụ `strcmp`

```
#include <stdio.h>

int main()
{
    char  s1[] = "Minor";
    char  s2[] = "Tony";
    int cmp = strcmp(s1, s2);
    if (cmp < 0)
        printf("%s < %s", s1, s2);
    else
        if (cmp == 0)
            printf("%s = %s", s1, s2);
        else
            printf("%s > %s", s1, s2);
    return 0;
}
```

```
Minor < Tony
```

Chuỗi ký tự – Một số hàm thư viện

- Gán nội dung chuỗi:
  - *Chép toàn bộ chuỗi source sang chuỗi dest:*

```
int strcpy(char *dest, const char *src);
```

  - *Chép tối đa n ký tự từ source sang dest:*

```
int strncpy(char *dest,
            const char *src, int n);
```

- Tạo chuỗi mới từ chuỗi đã có:

```
char *strdup(const char *src);
```

Chuỗi ký tự – ví dụ `strcpy`

```
#include <stdio.h>

int main()
{
    char s[] = "Tony Blurt";
    char s2[100], *s3;

    strcpy(s2, s);
    printf("%s\n", s2);
    strncpy(s2 + 2, "12345", 3);
    printf("%s\n", s2);
    s3 = strdup(s + 5);
    printf("%s\n", s3);
    free(s3);
    return 0;
}
```

```
Tony Blurt
To123Blurt
Blurt
```

Chuỗi ký tự – M ột số hàm thư viện

- Nối chuỗi:

```
char *strcat(char *dest,
                    const char *src);
```

- Tách chuỗi:

```
char *strtok(char *s,
                    const char *sep);
```

Trả về địa chỉ của đoạn đầu tiên. Muốn tách đoạn kế tiếp tham số thứ nhất sẽ là NULL

Chuỗi ký tự – ví dụ `strtok`

```
#include <stdio.h>

#define SEPARATOR "., "

int main()
{
    char s[]= "Thu strtok: 9,123.45";
    char *p;

    p = strtok(s, SEPARATOR);
    while (p != NULL)   {
        printf("%s\n", p);
        p = strtok(NULL, SEPARATOR);
    }
    return 0;
}
```

```
Thu
strtok:
9
123
45
```

Chuỗi ký tự – Một số hàm thư viện

- Tìm một ký tự trên chuỗi:

```
char *strchr(const char *s, int c);
```

- Tìm một đoạn ký tự trên chuỗi:

```
char *strstr(const char *s1,
             const char *s2);
```

Chuỗi ký tự – ví dụ tìm kiếm

```
#include <stdio.h>

int main()
{
    char s[]= "Thu tim kiem chuoi";
    char *p;

    p = strchr(s, 'm');
    printf("%s\n", p);
    p = strstr(s, "em");
    printf("%s\n", p);
    return 0;
}
```

```
m kiem chuoi
em chuoi
```

Chuỗi ký tự – chèn một đoạn ký tự

```
#include <stdio.h>

void StrIns(char *s, char *sub)
{
    int  len = strlen(sub);
    memmove(s + len, s, strlen(s)+1);
    strncpy(s, sub, len);
}
int main()
{
    char s[]= "Thu chen";

    StrIns(s, "123");       printf("%s\n", s);
    StrIns(s + 8, "45");    printf("%s\n", p);
    return 0;
}
```

```
123 Thu chen
123 Thu 45chen
```

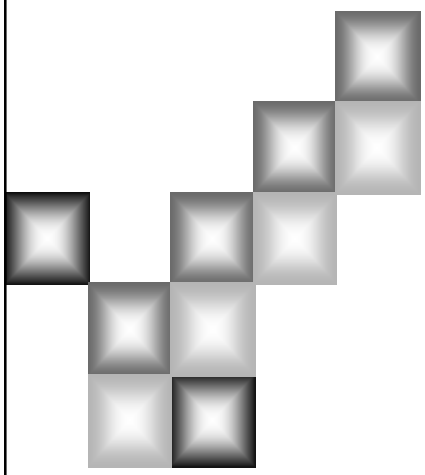
Chuỗi ký tự – xóa một đoạn ký tự

```
#include <stdio.h>

void StrDel(char *s, int n)
{
    memmove(s, s + n, strlen(s+n)+1);
}
int main()
{
    char s[]= "Thu xoa 12345";

    StrDel(s, 4);       printf("%s\n", s);
    StrDel(s + 4, 3);   printf("%s\n", p);
    return 0;
}
```

```
xoa 12345
xoa 45
```

Tóm lược

- Khai báo
- Nhập / xuất
- Con trỏ và chuỗi ký tự
- Một số hàm thư viện
- Chèn / loại bỏ một đoạn con



Cấu trúc - Struct

Kiểu cấu trúc

- Khái niệm
- Khai báo
- Truy xuất các thành phần
- Cấu trúc & mảng
- Con trỏ đến cấu trúc

Khái niệm

- Cấu trúc là kiểu dữ liệu gồm một nhóm các thành phần có kiểu không giống nhau, mỗi thành phần được xác định bằng một tên riêng biệt.
- Kiểu của mỗi thành phần trong cấu trúclà một kiểu đã được định nghĩa trước, kể cả mảng và các cấu trúc khác.

Cấu trúc – Khai báo trong C

Một kiểu cấu trúc được định nghĩa với từ khóa **`struct`**.

```
typedef struct Tênkiểu
{
    Kiểuthànhphần    Tênthànhphần;
    Kiểuthànhphần    Tênthànhphần;
    Kiểuthànhphần    Tênthànhphần;
    Kiểuthànhphần    Tênthànhphần;
    ...
};
```

Cấu trúc – ví dụ

```
typedef struct TDate
{
   char    day;
   char    month;
   int     year;
};
```

```
typedef struct TBook
{
   char    title[80];
   char    author[80];
   float   price;
   char    isbn[20];
};
```

```
typedef struct TStudent
{
   char    ID[10];
   char    firstname[10];
   char    lastname[20];
   TDate   dob;
   float   marks[10];
};
```

```
//khai báo các biến
TBook       book;
TStudent    list[100];
```

Cấu trúc – Truy xuất các thành phần

- Các thành phần của một biến kiểu cấu trúc được truy xuất thông qua tên biến, dấu "." và tên thành phần.

```
void Print(TStudent m)
{
     printf("Name           : %s %s\n",
                  m.firstname, m.lastname);
     printf("Student ID     : %s\n", m.ID);
     printf("Date of birth  : %hi/%hi/%i",
           m.dob.day, m.dob.month, m.dob.year);
     printf("Marks          : ");
     for (int i=0; i<10; i++)
           printf("%.2f ", m.marks[i]);
}
```

Cấu trúc – Truy xuất các thành phần

```
void ReadInfo(TStudent &m)
{
     printf("Type student ID: ");
     scanf("%s", m.ID);
     printf("Type first name: ");
     gets(m.firstname);
     printf("Type last name: ");
     gets(m.lastname);
     printf("Date of birth (d m y): ");
     scanf("%hi %hi %i", &(m.dob.day),
                  &(m.dob.month), &(m.dob.year));
     printf("Marks (10 floats): ");
     for (int i=0; i<10; i++)
           scanf("%f", &(m.marks[i]));
}
```

Tóm lược

- Định nghĩa KDL
- Kiểu dữ liệu cơ bản: Số nguyên, số thực
- Kiểu dữ liệu cấu trúc: Định nghĩa, một số Kiểu cấu trúc cơ bản
- Mảng
- Chuỗi ký tự
- Struct