

ساختار

مبانی برنامه‌نویسی

(۱۳۹۱-۱۳۹۰-۱۳۸۹)

جلسه‌ی بیست و نهم



دانشگاه شهید بهشتی

پاییز ۱۳۹۳

دانشکده‌ی مهندسی برق و کامپیوتر

احمد محمودی ازناوه

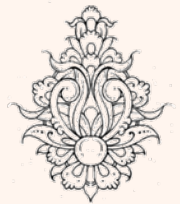
فهرست مطالب

• ساختار

- ساختارهای تودرتو
- ارسال ساختار به تابع
- آرایه‌ای از ساختارها

• union

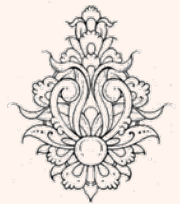
• ثابت‌های شمارشی



– در بسیاری موارد لازم است چند عنصر غیر هم‌نام و با انواع مختلف را تحت عنوان یک نام در حافظه ذخیره نماییم.

VARIABLE DEFINITION	DATA HELD
int empNumber;	Employee number
char name[30];	Employee name
double hours;	Hours worked
double payRate;	Hourly pay rate
double grossPay;	Gross pay

```
struct PayRoll{  
    int empNumber;  
    char name[30];  
    double hours,  
           payRate,  
           grossPay;  
};
```



تعریف ساختار

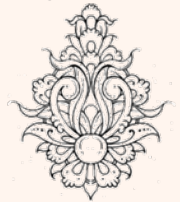
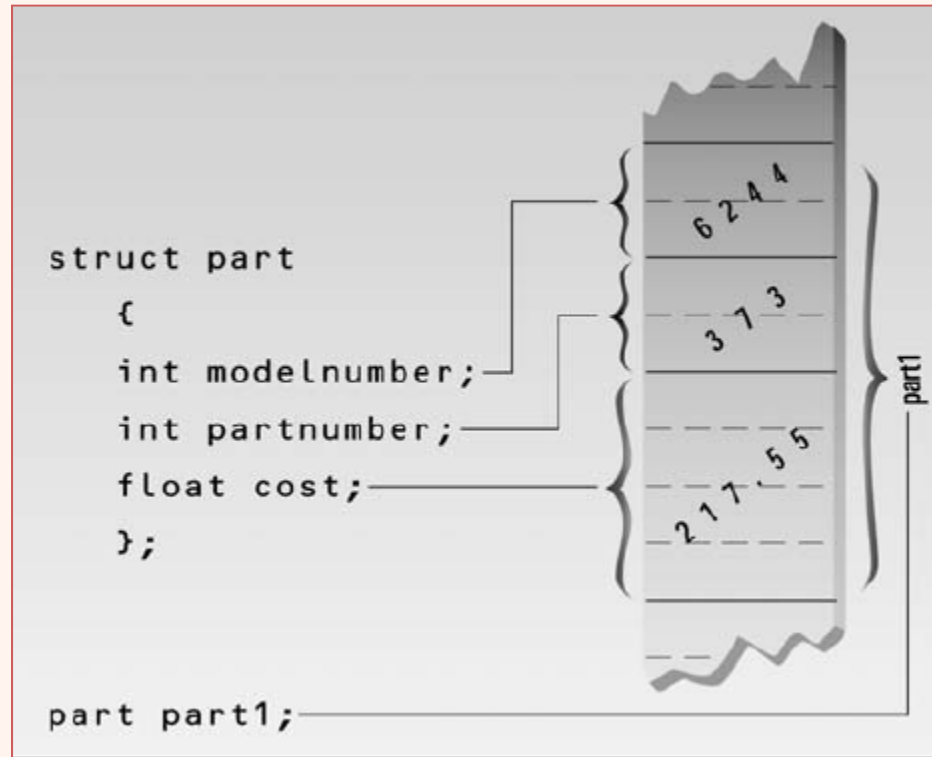
کلمه کلیدی `struct`

نام `struct`

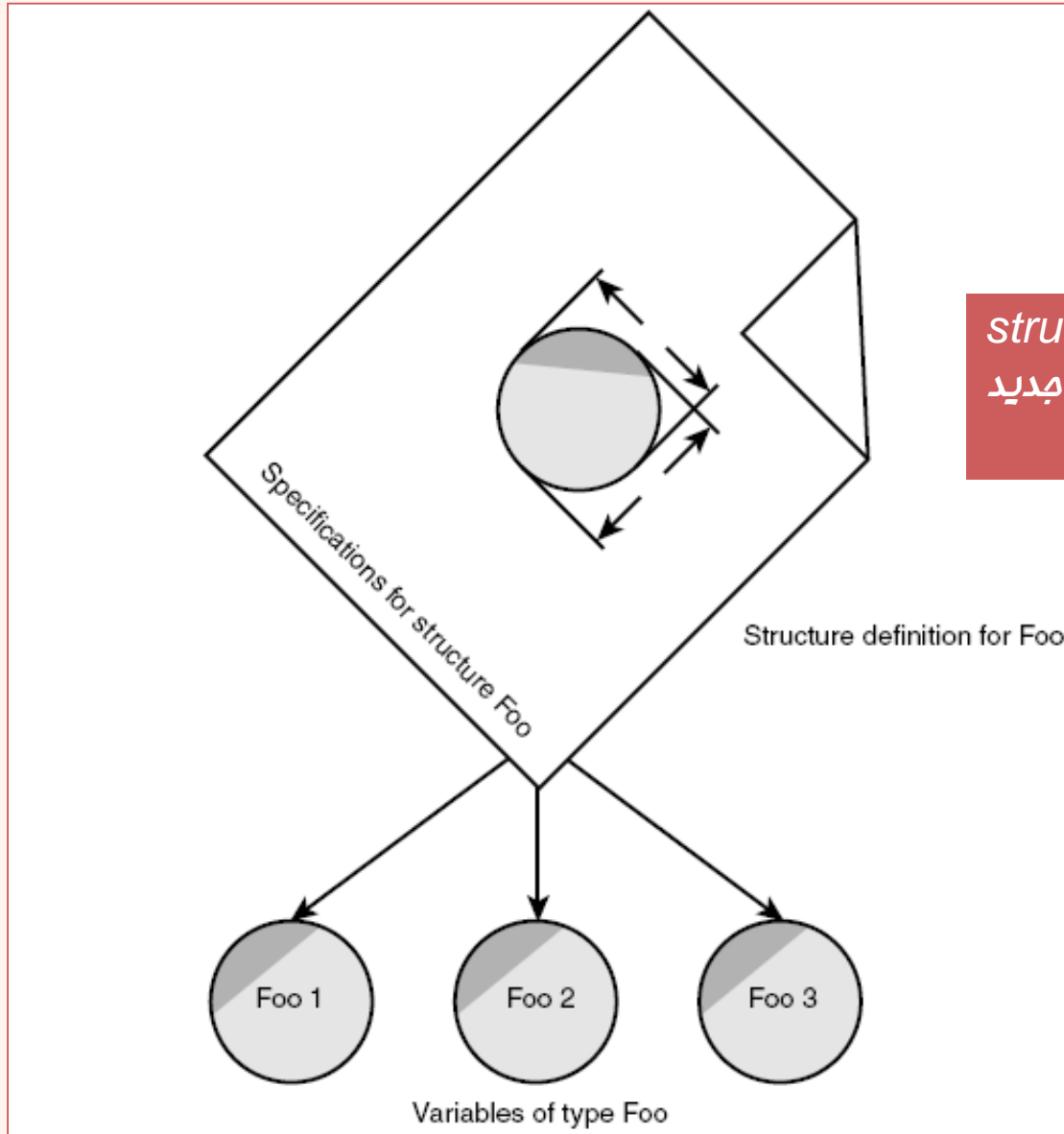
```
struct part
```

```
{  
  int modelnumber;  
  int partnumber;  
  float cost;  
};
```

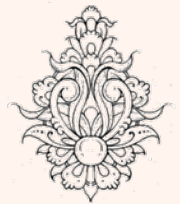
اعضا



در این سیستم نوع `int` دوبایتی در نظر گرفته شده است.



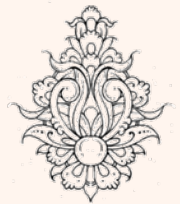
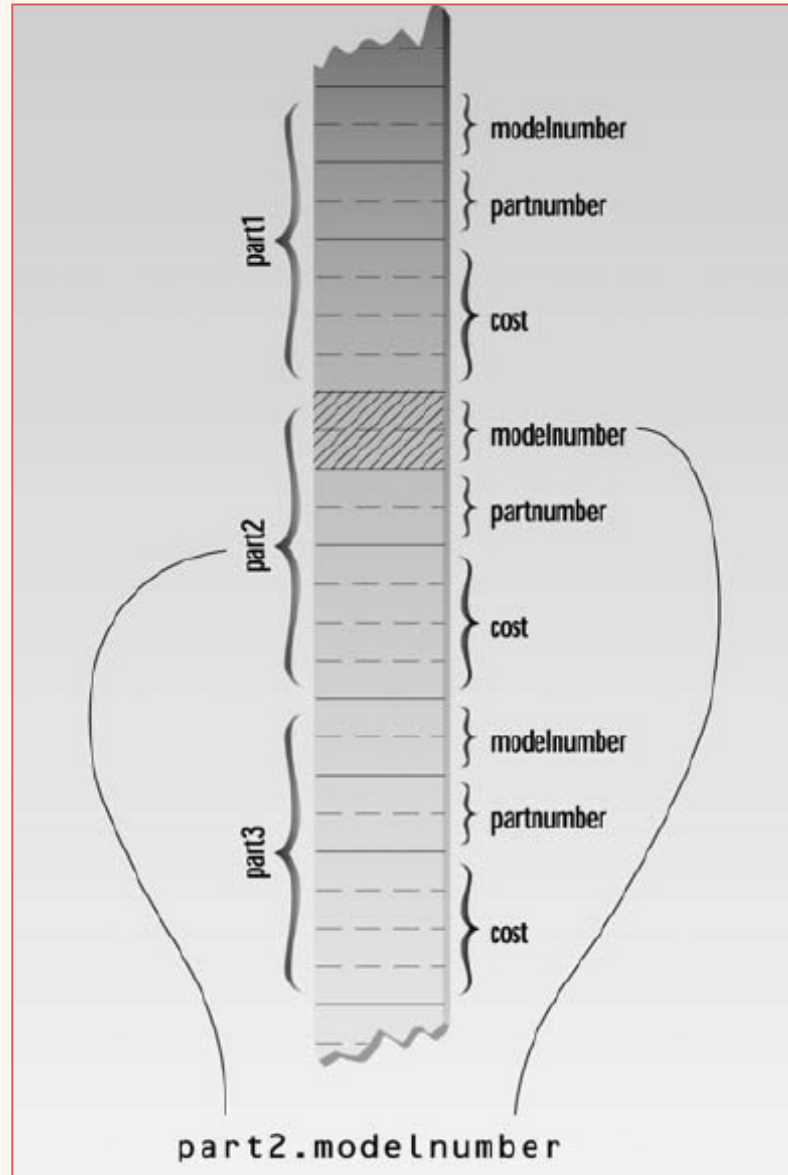
به وسیلهی تعریف *structure* می‌توان به گونه‌ای نوعی داده‌ی جدید به زبان اضافه نمود.



ساختار در حافظه

```
struct part  
{  
  int modelnumber;  
  int partnumber;  
  float cost;  
};
```

```
part part1;  
part1.modelnumber = 6244;
```



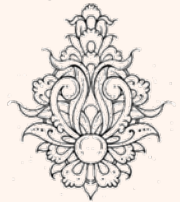
ساختار (ادامه...)

- می‌توان در انتهای تعریف و قبل از این که قرار داده شود نام شی‌های تعریف شده را قرار دهیم:

```
struct product {  
    int weight;  
    float price;  
} apple, banana, melon;
```

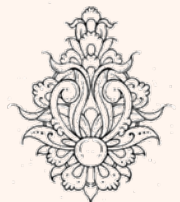
```
apple.weight  
apple.price  
banana.weight  
banana.price  
melon.weight  
melon.price
```

- دسترسی به اجزا:



```
struct part
{
    int modelnumber;
    int partnumber;
    float cost;
};
////////// Model 6244, part 373, costs $217.55
int main()
{
    part part1;
    part1.modelnumber = 6244;
    part1.partnumber = 373;
    part1.cost = 217.55;

    cout << "Model " << part1.modelnumber;
    cout << ", part " << part1.partnumber;
    cout << ", costs $" << part1.cost << endl;
    return 0;
}
```




```
struct Date
{ int day,
  month,
  year;
};
```

مقداردهی اولیه و انتساب

– می‌توان به شیوه‌ی زیر مقداردهی اولیه را صورت داد:

```
Date birthday = {23, 8, 1983};
```

– همچنین می‌توان پس از تعریف، فرآیند انتساب را انجام داد:

```
Date d1;
```

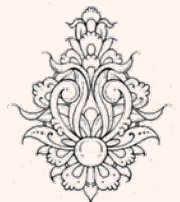
```
d1=birthday;
```

C++ اجازه می‌دهد عضوی از اعضا را مقداردهی نمایید. ولی اگر عنصر آخرین عضو نباشد تا انتها مقادیر را نمی‌توان مقداردهی نمود.

```
Date birthday={23, ,1983};
```



```
Date birthday={23,8};
```



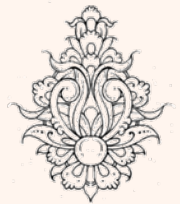
مقداردهی اولیه و انتساب (ادامه...)

- با تعریف struct مشخص می‌کنیم نوع جدید چگونه است
- تعداد و نوع اعضاء مشخص می‌گردد.
- مادامی که متغیری از جنس struct تعریف نشده باشد فضایی در حافظه در نظر گرفته نمی‌شود تا بتوان مقداردهی اعضا را صورت داد.
- به وسیله‌ی استفاده از struct مشخص می‌کنیم نوع جدید چگونه است، مقداردهی پس از تعریف می‌باید صورت پذیرد.

```
struct Date
{   int day    = 23,
    month = 8,
    year  = 1983;
};
```



illegal



```

struct part
{
int modelnumber;
int partnumber;
float cost;
};
////////////////////////////////////
int main()
{ //initialize variable
part part1 = { 6244, 373, 217.55 };
part part2; //define variable

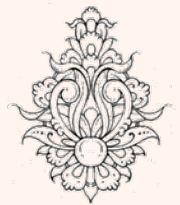
cout << "Model " << part1.modelnumber;
cout << ", part " << part1.partnumber;
cout << ", costs $" << part1.cost << endl;
part2 = part1; //assign first variable to second
//display second variable
cout << "Model " << part2.modelnumber;
cout << ", part " << part2.partnumber;
cout << ", costs $" << part2.cost << endl;
return 0;
}

```

```

Model 6244, part 373, costs $217.55
Model 6244, part 373, costs $217.55

```



```

struct PayRoll
{
    int    empNumber;
    string name;
    double hours,
           payRate,
           grossPay;
};

```

نکات

• به متغیرهای هر structure

تنها به صورت مجزا می‌توان دسترسی داشت:

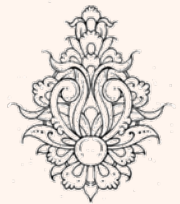
– اگر employe متغیر PayRoll باشد داریم:

```
cout << employe << endl;           // Error!
```

– اگر employe1 و employe2 متغیر PayRoll باشد داریم:

```
if (employe1 = employe2)           // Error!
```

```
if (employe1.hours == employe2.hours) // Legal
```



مثال

```
struct Circle // Declare what a Circle structure looks like
{
double radius;
double diameter;
double area;
};

const double PI = 3.14159;
int main()
{
Circle c1, c2; // Define 2 Circle structure variables

cout << "Enter the diameter of circle 1: ";
cin >> c1.diameter;
cout << "Enter the diameter of circle 2: ";
cin >> c2.diameter;

c1.radius = c1.diameter / 2;
c1.area = PI * pow(c1.radius, 2.0);
c2.radius = c2.diameter / 2;
c2.area = PI * pow(c2.radius, 2.0);

cout << "\nThe radius and area of the circles are\n";
cout << "Circle 1 -- Radius: " << c1.radius
<< " Area: " << setw(6) << c1.area << endl;
cout << "Circle 2 -- Radius: " << c2.radius
<< " Area: " << c2.area << endl;
if (c1.area == c2.area)
cout << "The two circles have the same area.\n\n";
return 0;
}
```

```
Enter the diameter of circle 1: 3
Enter the diameter of circle 2: 4

The radius and area of the circles are
Circle 1 -- Radius: 1.5 Area: 7.06858
Circle 2 -- Radius: 2 Area: 12.5664
```

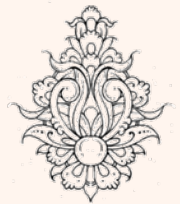


مثال

- هر نقطه دارای دو مختصات x و y می‌باشد.
- اگر بخواهیم یک نقطه را به وسیله‌ی struct نشان دهیم خواهیم داشت:

```
struct point{  
    int x;  
    int y;  
};
```

- به وسیله‌ی struct می‌توان نوعی جدید در نظر گرفت مانند `int` و `float`
- تمام تعاریفی که برای انواع مختلف صورت داده‌ایم برای struct ها نیز معتبر است.



- می‌توانیم struct ای از struct ها نیز تعریف کنیم:

```
struct rect{  
    struct point pt1;  
    struct point pt2;  
};
```

- در این حالت اگر داشته باشیم:

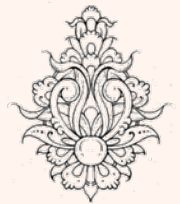
```
struct rect screen;
```

- می‌توان به تمامی اعضا دسترسی داشت:

```
screen.pt1.x; //access to x coordinate of point pt1
```

- اعضای struct در حافظه در امتداد یکدیگر واقع شده‌اند.

نکته: هنگام تعریف struct در زبان C استفاده از این لفظ لازم است.



ساختار تودرتو (ادامه...)

```
struct Costs{
    double wholesale;
    double retail;
};

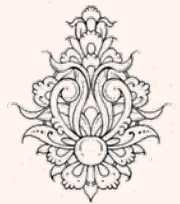
struct Item{
    char partNum[20];
    char description[20];
    Costs pricing;
};
```

Widget

partNum	<input type="text"/>				
description	<input type="text"/>				
pricing	<table><tr><td>wholesale</td><td><input type="text"/></td></tr><tr><td>retail</td><td><input type="text"/></td></tr></table>	wholesale	<input type="text"/>	retail	<input type="text"/>
wholesale	<input type="text"/>				
retail	<input type="text"/>				

Item widget;

```
strcpy(widget.partNum, "123A");
strcpy(widget.description, "iron widget");
widget.pricing.wholesale = 100.0;
widget.pricing.retail = 150.0;
```




```

struct CostInfo
{
    double food,
    medical,
    license,
    misc;
};

struct PetInfo
{
    char name[20];
    char type[20];
    int age;
    CostInfo cost; // A E
    // nested inside as c
};

```

```

int main()
{
    PetInfo pet; // Define a structure variable
    cout<<"Enter the pet info:\n";
    cout<<"pet name: ";
    cin >> pet.name;
    cout<<"\npet type: ";
    cin >> pet.type;
    cout<<"\npet age: ";
    cin >> pet.age;
    cout<<"\npet cost food: ";
    cin >> pet.cost.food;
    cout<<"\npet cost medical: ";
    cin >> pet.cost.medical;
    cout<<"\npet cost license: ";
    cin >> pet.cost.license;
    cout<<"\npet cost misc: ";
    cin >> pet.cost.misc;

    cout << "Annual costs for my " << pet.age << "-year-old "
    << pet.type << " " << pet.name << " are $"
    << (pet.cost.food + pet.cost.medical +
    pet.cost.license + pet.cost.misc) << endl;
    return 0;
}

```

```

Enter the pet info:
pet name: loosy

pet type: cat

pet age: 5

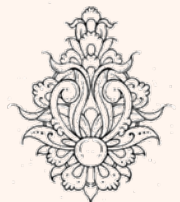
pet cost food: 250

pet cost medical: 150

pet cost license: 7

pet cost misc: 60
Annual costs for my 5-year-old cat loosy are $467

```



نوعی ارسال به توابع

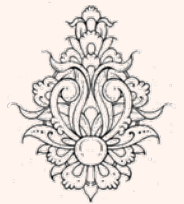
Structures as Function Arguments

- به طور کلی برای ارسال مقادیر structure به یک تابع سه رویکرد در نظر گرفته می‌شود:

– ارسال جداگانه عناصر

– ارسال structure

– ارسال اشاره‌گر آن



نوعی ارسال به توابع (ادامه...)

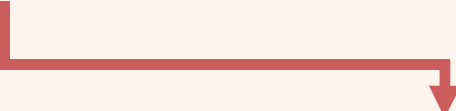
```
struct Rectangle{  
    double length,  
    width  
    area;  
};
```

```
// Define a Rectangle variable  
Rectangle box;
```

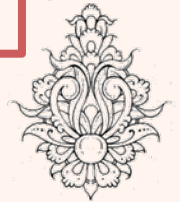
```
// Define a function that has 2 double parameters  
double multiply (double x, double y){  
    return x * y ;  
}
```

```
box.area = multiply ( box.length , box.width) ;
```

```
showRect (box) ;
```



```
void showRect (Rectangle r) {  
    cout<< r.length << endl ;  
    cout<< r.width << endl ;  
    cout<< r.area << endl ;  
}
```



```

struct InvItem
{
    int partNum;
    char description[80];
    int onHand;
    double price;
};

void getItem(InvItem&);
void showItem(InvItem);

int main()
{
    InvItem part;
    getItem (part);
    showItem(part);
    return 0;
}

```

```

void getItem(InvItem& piece)
{
    cout << "Enter the part number: ";
    cin >> piece.partNum;
    cout << "Enter the part description: ";
    cin.getline (piece.description, 80, '$');
    cout << "Enter the quantity on hand: ";
    cin >> piece.onHand;
    cout << "Enter the unit price: ";
    cin >> piece.price;
}

void showItem(InvItem piece)
{
    cout << "Part Number : " << piece.partNum << endl;
    cout << "Description : " << piece.description << endl;
    cout << "Units On Hand: " << piece.onHand << endl;
    cout << "Price : $" << piece.price << endl;
}

```

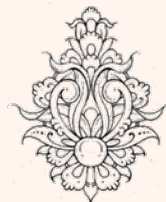
```

Enter the part number: 2
Enter the part description: The speed was too high
the driver could not control it$
Enter the quantity on hand: 234
Enter the unit price: 1.5
Part Number : 2
Description :
The speed was too high
the driver could not control it
Units On Hand: 234
Price : $1.5

```

- ارسال یک struct از طریق مقدار سبب می‌شود ابتدا یک کپی از struct مذکور ایجاد و فرآیند به روی کپی مقادیر صورت پذیرد.
- به دلیل حجم بالای داده‌ها کپی کردن معمولاً بازده سیستم را کاهش می‌دهد.
- ارسال از طریق ارجاع می‌تواند مشکل را مرتفع سازد.
- ارسال از طریق ارجاع ثابت سبب می‌شود از احتمال تغییر مقادیر توسط تابع جلوگیری به عمل آید.

```
void showItem (const InvItem&); // function prototype  
void showItem (const InvItem &piece) // function header
```

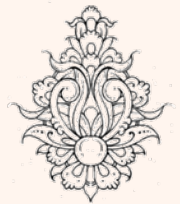


ساختار به عنوان مقدار بازگشتی

- یک تابع می‌تواند یک struct را به عنوان مقدار برگشتی بازگرداند.

```
struct Circle
{
    double radius;
    double diameter;
    double area;
};
```

```
Circle getData()
{
    Circle temp;
    temp.radius = 10.0;
    temp.diameter = 20.0;
    temp.area = 314.159;
    return temp;
}
```





```
struct Circle
{
double radius;
double diameter;
double area;
};
```

```
Circle getInfo();
const double PI = 3.14159;
```

```
int main()
{
Circle c1, c2;

c1 = getInfo();
c2 = getInfo();
```

```
c1.area = PI * pow(c1.radius, 2.0);
c2.area = PI * pow(c2.radius, 2.0);
```

```
cout << "\nThe radius and area of the circles are\n";
cout << "Circle 1 -- Radius: " << c1.radius
<< " Area: " << c1.area << endl;
cout << "Circle 2 -- Radius: " << c2.radius
<< " Area: " << c2.area << endl;
if (c1.area == c2.area)
cout << "The two circles have the same area.\n\n";
```

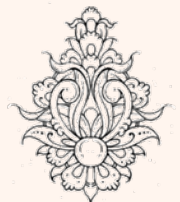
```
return 0;
}
```

```
Circle getInfo() // Function return type is a Circle structure
{
Circle round; // Local structure variable to hold input data

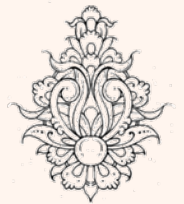
cout << "Enter the diameter of a circle: ";
cin >> round.diameter;
round.radius = round.diameter / 2;
return round;
}
```

```
Enter the diameter of a circle: 4
Enter the diameter of a circle: 8
```

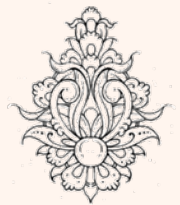
```
The radius and area of the circles are
Circle 1 -- Radius: 2 Area: 12.5664
Circle 2 -- Radius: 4 Area: 50.2654
```



- C++ تنها اجازه می‌دهد یک مقدار بازگشتی داشته باشیم.
- به وسیله‌ی استفاده از struct یک مقدار برگشت داده می‌شود، هر چند متشکل از تعدادی عضو باشد.
- با استفاده از struct تعدادی عنصر را کنار هم داشته، می‌توان همه را در قالب یک object از تابع بازگشت داد.



- تابعی به نام makepoint بنویسید که طول و عرض را گرفته، یک نوع از جنس نقطه بازگرداند.



```

#include <iostream>
using namespace std;
#define MAXX 8
#define MAXY 8

struct point{
    int x;
    int y;
};
struct rect{
    struct point pt1;
    struct point pt2;
};
struct point makepoint(int , int );
int main()
{
    rect screen;
    point middle;
    screen.pt1=makepoint(0,0);
    screen.pt2=makepoint(MAXX,MAXY);
    middle=makepoint((screen.pt1.x+screen.pt2.x)/2,(screen.pt1.y+screen.pt2.y)/2);
    cout << "The X of middle point is:" << middle.x <<endl;
    cout << "The Y of middle point is:" <<middle.y <<endl;
}
struct point makepoint(int x, int y)
{
    struct point temp;
    temp.x=x;
    temp.y=y;
    return temp;
}

```

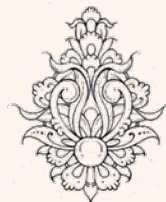
← C++ style declaration

← C style declaration

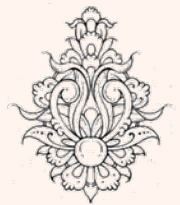
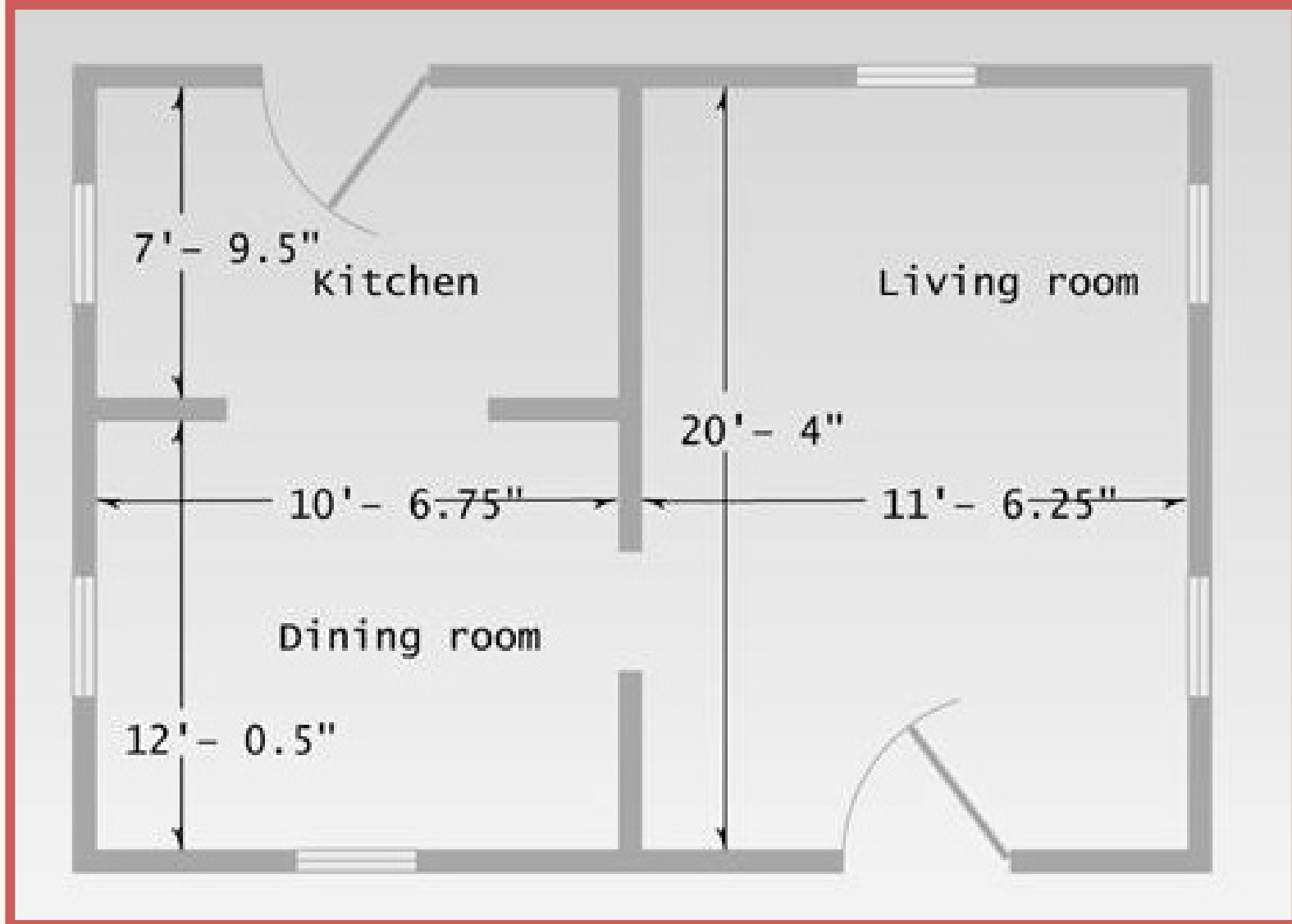
```

The X of middle point is:4
The Y of middle point is:4

```



A Measurement Example



```

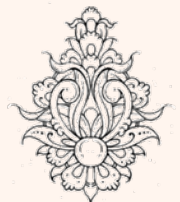
struct Distance //English distance
{
    int feet;
    float inches;
};
/////////////////////////////////////////////////////////////////
struct Room //rectangular area
{
    Distance length; //length of rectangle
    Distance width; //width of rectangle
};
/////////////////////////////////////////////////////////////////
int main()
{
    Room dining; //define a room

    dining.length.feet = 13; //assign values to room
    dining.length.inches = 6.5;
    dining.width.feet = 10;
    dining.width.inches = 0.0;|

    //convert length & width
    float l = dining.length.feet + dining.length.inches/12;
    float w = dining.width.feet + dining.width.inches/12;
    //find area and display it
    cout << "Dining room area is " << l * w
         << " square feet\n" ;
    return 0;
}

```

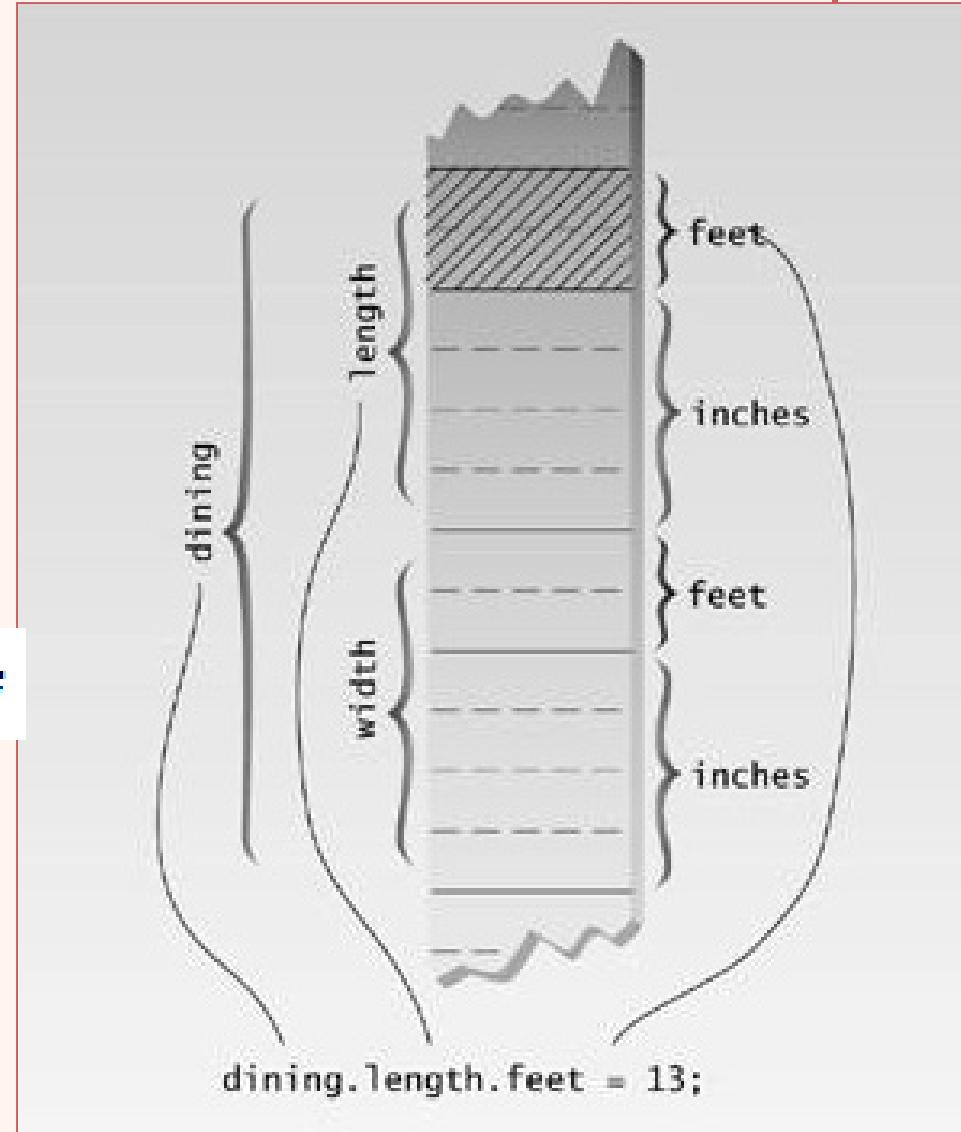
Dining room area is 135.417 square feet



دستیابی به عناصر

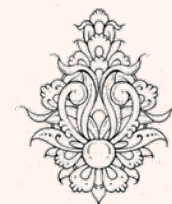
dining.length.feet = 13;

```
Room dining = { {13, 6.5}, {10, 0.0} };
```



- به لحاظ تئوری عمق می تواند به هر اندازه که ممکن است ادامه یابد.
- می توانیم داشته باشیم:

apartment1.laundry_room.washing_machine.width.feet



تمرین (آرایه‌ای از struct)

- برنامه‌ای بنویسید که مشخصات ۵ دانشجو شامل نام و نام خانوادگی و معدل را از ورودی بخواند و آنها را بر اساس معدل مرتب نموده و سپس چاپ نماید.



```
#define siz 5
```

```
struct stdinfo{  
    char name[30];  
    char Lname[30];  
    float avg;  
};
```

```
int main()  
{  
    int i ,j;  
    stdinfo s[siz],temp;  
    for(i=0;i<siz;i++){  
        cout << i+1 <<"th student information:";  
        cin>> s[i].name >> s[i].Lname >>s[i].avg;  
        cout << endl;  
    }  
    for(i=1;i<siz;i++)  
        for(j=0;j<siz-i;j++)  
            if(s[j].avg>s[j+1].avg)  
                {  
                    temp=s[j];  
                    s[j]=s[j+1];  
                    s[j+1]=temp;  
                }  
    for(i=0;i<siz;i++)  
        cout<< i+1<< "th is:"<<s[i].name <<"\t"<< s[i].Lname <<"\t"<< s[i].avg <<endl;  
}
```

```
1th student information:ali alvai 15  
2th student information:reza rezaie 13  
3th student information:amir amiri 17  
4th student information:mohammad mohammadi 14  
5th student information:hasan hasani 11
```

```
1th is:hasan      hasani  11  
2th is:reza      rezaie  13  
3th is:mohammad mohammadi  14  
4th is:ali       alvai   15  
5th is:amir      amiri   17
```


Union

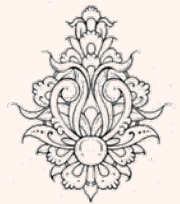
- Union محلی از حافظه است که توسط چندین متغیر به طور اشتراکی مورد استفاده قرار می‌گیرد.

```
union mytypes_t {  
    char c;  
    int i;  
    float f;  
} mytypes;
```

- در این حالت سه عضو تعریف می‌گردد:

```
mytypes.c  
mytypes.i  
mytypes.f
```

- از آنجا که تمامی عناصر تعریف شده به مکانی یکسان در حافظه اشاره می‌نمایند تغییر هر عنصر به صورت مستقل از دیگری امکان‌پذیر نیست.



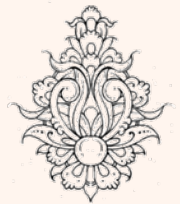
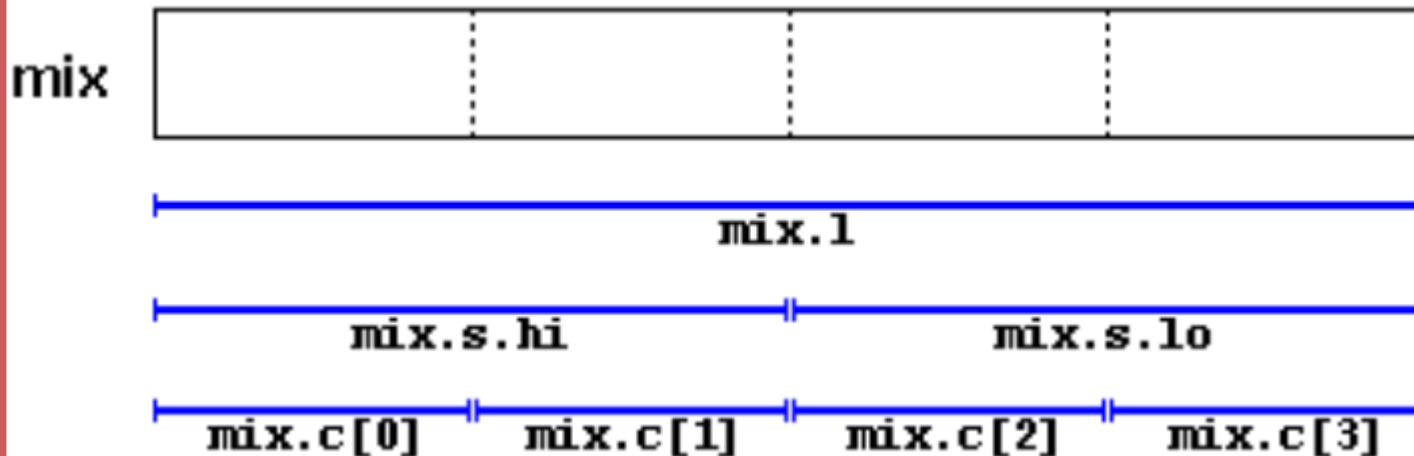
Union (ادامه...)

- از مسئولیت‌های برنامه‌نویس است که تخیرات را دنبال کند و سبب بروز خطا در برنامه نگردد.

```
union mix_t {  
    long l;  
    struct {  
        short hi;  
        short lo;  
    } s;  
    char c[4];  
} mix;
```

- به عنوان مثال اگر داشته باشیم:

- می‌توان در نظر گرفت:



```
union PaySource // Declare a union.
{
short hours; // These two variables share
float sales; // the same memory space.
};
```

```
int main()
{
PaySource employee1; // employee1 is a PaySource union.
// This employee can have hours or
// sales, but not both at once.
char hourlyType; // 'y' if hourly, 'n' if on commission
float payRate, grossPay;

cout << "This program calculates either hourly wages or "
<< "sales commission.\n";
cout << "Is this an hourly employee (y or n)? ";
cin >> hourlyType;
if (hourlyType == 'y') // This is an hourly employee.
{
cout << "What is the hourly pay rate? ";
cin >> payRate;
cout << "How many hours were worked? ";
cin >> employee1.hours;
grossPay = employee1.hours * payRate;
cout << "Gross pay: $" << grossPay << endl;
}
else // Employee works on commission.
{
cout << "What are the total sales for this employee? ";
cin >> employee1.sales;
grossPay = employee1.sales * 0.10;
cout << "Gross pay: $" << grossPay << endl;
}
```

```
This program calculates either hourly wages or sales commission.
Is this an hourly employee (y or n)? y
What is the hourly pay rate? 23
How many hours were worked? 65
Gross pay: $1495
```



```
#include <iostream>
using namespace std;

union myUnion{

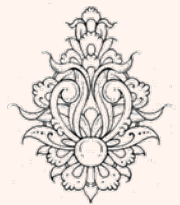
    int intNum;
    float fNum;
};

int main() {
    myUnion smp1;
    cout<<"please enter a number:\n";
    cin>>smp1.fNum;

    cout<<hex<<smp1.intNum<<endl;

    return 0;
}
```

```
please enter a number:
1.75
3fe00000
```



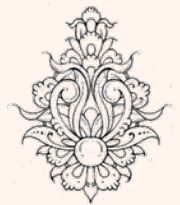
Anonymous unions

- در C++ نوعی union با نام anonymous می‌توان تعریف نمود.
- anonymous union بدون نام تعریف می‌گردد.
- در این صورت است که به صورت مستقیم می‌توان به عناصر مربوط دسترسی پیدا نمود.

structure with regular union	structure with anonymous union
<pre>struct { char title[50]; char author[50]; union { float dollars; int yens; } price; } book;</pre>	<pre>struct { char title[50]; char author[50]; union { float dollars; int yens; }; } book;</pre>

book.price.dollars
book.price.yens

book.dollars
book.yens



```

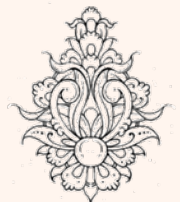
int main()
{
    union // Declare an anonymous union.
    {
        short hours;
        float sales;
    };
    char hourlyType; // 'y' if hourly, 'n' if on commission
    float payRate, grossPay;
    cout << "This program calculates either hourly wages or "
    << "sales commission.\n";
    cout << "Is this an hourly employee (y or n)? ";
    cin >> hourlyType;
    if (hourlyType == 'y') // This is an hourly employee.
    {
        cout << "What is the hourly pay rate? ";
        cin >> payRate;
        cout << "How many hours were worked? ";
        cin >> hours; // Anonymous union member
        grossPay = hours * payRate;
        cout << "Gross pay: $" << grossPay << endl;
    }
    else // Employee works on commission.
    {
        cout << "What are the total sales for this employee? ";
        cin >> sales; // Anonymous union member
        grossPay = sales * 0.10;
        cout << "Gross pay: $" << grossPay << endl;
    }
    return 0;
}

```

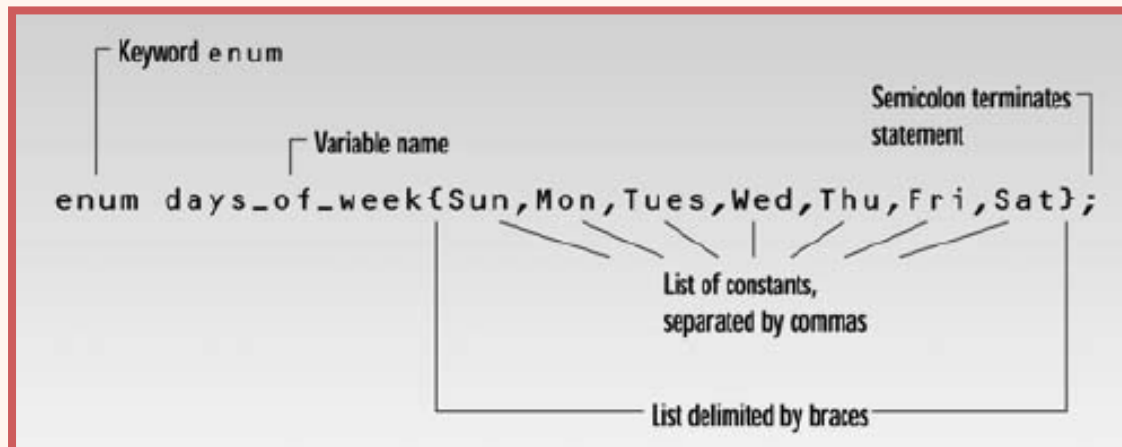
```

This program calculates either hourly wages or sales commission.
Is this an hourly employee (y or n)? n
What are the total sales for this employee? 56
Gross pay: $5.6

```



- به وسیلهی struct می‌توان نوع جدیدی را ایجاد نمود.
 - راه دیگر استفاده از **enumeration** می‌باشد.
 - هنگامی که لیست کوتاهی در دست باشد از enumeration استفاده می‌شود.
 - بدین ترتیب لیستی از نام‌هاست که مقداری مجاز را به خود اختصاص می‌دهد.
 - به این مقادیر مجاز **enumerator** می‌گوییم.
- ```
enum days_of_week { Sun, Mon, Tue, Wed, Thr, Fri, Sat };
days_of_week day1, day2;
```



## ثابت شمارشی (ادامه...)

```
// dayenum.cpp
// demonstrates enum types
#include <iostream>
using namespace std;

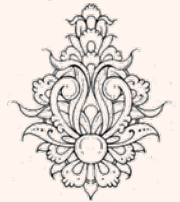
//specify enum type
enum days_of_week { Sun, Mon, Tue, Wed, Thu, Fri, Sat };

int main()
{
 days_of_week day1, day2; //define variables
 //of type days_of_week
 day1 = Mon; //give values to
 day2 = Thu; //variables

 int diff = day2 - day1; //can do integer arithmetic
 cout << "Days between = " << diff << endl;

 if(day1 < day2) //can do comparisons
 cout << "day1 comes before day2\n";
 return 0;
}
```

Days between = 3  
day1 comes before day2





# زکات

• اگر خواسته باشیم مقدار از یک شروع شود:

```
enum days_of_week { Sun=1, Mon, Tue, Wed, Thu, Fri, Sat };
```

• می‌توان مقادیر را به صورت صریح بیان نمود.

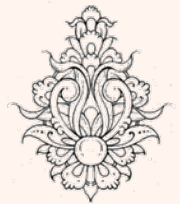
```
enum FooSize { SMALL = 10, MEDIUM = 100, LARGE = 1000 };
```

• چند مثال:

```
enum coins { penny, nickel, dime, quarter, half-dollar,
dollar };
```

```
enum switch { off, on };
```

```
enum months { Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep,
Oct, Nov, Dec };
```

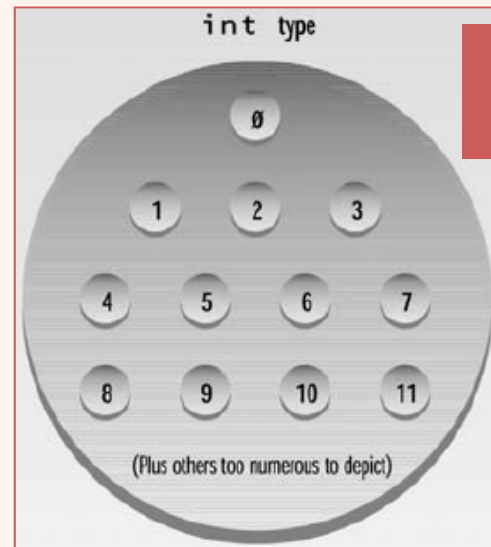


# نکات

- اگر داشته باشیم:

```
enum MyEnumType { ALPHA, BETA, GAMMA };
enum MyEnumType x; // legal in both C and C++
MyEnumType y; // legal only in C++
int i = BETA; // give i a value of 1
int j = 3 + GAMMA; // give j a value of 5
MyEnumType x = 2; // NOT be allowed by compiler
MyEnumType y = 123; // NOT be allowed by compiler
```

تعداد کوچکی از مقادیر به  
وسیله‌ی اسامی خاصی  
نامگذاری و به آن‌ها مراجعه  
می‌شود



مجموعه‌ی بزرگی از مقادیر  
بدون نامگذاری



# ثابت شمارشی (ادامه...)

```
// define a new enum named Animal
enum Animal
{
 ANIMAL_CAT = -3,
 ANIMAL_DOG, // assigned -2
 ANIMAL_PIG, // assigned -1
 ANIMAL_HORSE = 5,
 ANIMAL_GIRAFFE = 5,
 ANIMAL_CHICKEN // assigned 6
};
```



## ثابت شمارشی (ادامه...)

```
int ParseFile()
{
 if (!OpenFile())
 return -1;
 if (!ReadFile())
 return -2;
 if (!Parsefile())
 return -3;

 return 0; // success
}
```

استفاده از ثابت‌های شمارشی به افزایش خوانایی برنامه کمک می‌کند

```
enum ParseResult
{
 SUCCESS = 0,
 ERROR_OPENING_FILE = -1,
 ERROR_READING_FILE = -2,
 ERROR_PARSING_FILE = -3,
};
```

```
ParseResult ParseFile()
{
 if (!OpenFile())
 return ERROR_OPENING_FILE;
 if (!ReadFile())
 return ERROR_READING_FILE;
 if (!Parsefile())
 return ERROR_PARSING_FILE;

 return SUCCESS;
}
```

توسعه  
بهره‌دهی

```

#include <iostream>
using namespace std;
int main(){
 enum Days{Sunday=2,Monday, Tuesday, Wednesday, Thursday, Friday, Saturday=1};
 Days TheDay;
 int j;
 cout<<"Please enter the day number (from 1 to 5)\n";
 cin>>j;
 TheDay=(Days)j;
 if(TheDay==Saturday || TheDay==Wednesday)
 cout<<"We have ITP! :)\n";
 else if(TheDay==6 || TheDay==7)
 cout<<"it is holiday. :D\n";
 else
 cout<<"We DO NOT have ITP! :(\n";
}

```

```

Please enter the day number<from 1 to 5>
5
We have ITP! :)

```



```

#include <stdio.h>
int main(){
 enum Days{Sunday=2,Monday, Tuesday, Wednesday, Thursday, Friday, Saturday=1};
 enum Days TheDay;
 printf("Please enter the day number(from 1 to 5)\n");
 scanf("%d",&TheDay);
 if(TheDay==Saturday || TheDay==Wednesday)
 printf("We have ITP! :)\n");
 else if(TheDay==6 || TheDay==7)
 printf("it is holiday. :D\n");
 else
 printf("We DO NOT have ITP! :(\n");
}

```

```

Please enter the day number(from 1 to 5)
7
it is holiday. :D

```

