ساختمان ها و نوع شمارشی

در این قسمت نوع های داده ها پیچیده تر که توسط خود برنامه نویس در برنامه تعریف می شوند نظیر ساختمان، union و شمارشی معرفی می شوند. ساختمان نوع داده ای است که شامل متغیربا انواع مختلف است که هرکدام را یک جزء از ساختمان می نامند. یونیون مشابه ساختمان است با این تفاوت که کلیه اجزای آن از یک فضای مشترک در حافظه استفاده می کنند. یعنی در هرلحظه از یک جزء آن می توان استفاده کرد.

ساختمان
typedef
یونیون
شمارشی
فیلدهای بیتی

انواع متغیرهائی که تاکنون استفاده شدند (مانند int، float، char، ...) نوع های پیش ساخته (built-in) هستند که تنها یک مقدار تکی را در خود نگه می دارد. می توان متغیرهائی در برنامه تعریف کرد که از نوع های داده کامل تری هستند و اجازه می دهند چند مقدار مرتبط بهم تحت یک نام ذخیره شوند. چون برنامه نویس این نوع داده ها را به عنوان نوع داده جدیدی در برنامه ایجاد می کند user-defined نامیده می شوند و شامل انواع زیر هستند:

- ساختمان
- Typedef
- یونیون
- شمارشی
- فیلذهای بیتی

ساختمان

اگر در برنامه ای برای مثال اطلاعات کارمندی ذخیره می شود، نگهداری داده های مجزا مانند نام، فامیل، حقوق و غیره در متغیرهای جداگانه و تشخیص اینکه داده ها متعلق به اطلاعات کارمند است آسان نیست. ساختمان (structure) راه ساده ای برای گروه بندی چند متغیر مرتبط بهم تحت یک نام مشترک است. نوع متغیرهای درون ساختمان برخلاف آرایه می تواند متفاوت از یکدیگر و از هر نوع داده استاندارد، آرایه، اشاره گر یا حتی ساختمان باشد. هر متغیر درون ساختمان یک جزء از ساختمان نامیده می شود.

تعریف کلی ساختمان به صورت زیر است:

```
struct structurename
{
   vartype varname;
   vartype varname;
} variable ;
```

کلمه کلیدی struct شروع تعریف ساختمان است. بدنبال آن اسم نوع ساختمان می آید که مانند اسامی متغیرها هر اسم مجازی می تواند باشد. اجزای ساختمان درون آکولاد قرار می گیرند.

مثال. تعریف نوع ساختمان کارمند به نام employee با اجزای lastname، firstname و salary.

```
struct employee {
   char lastname[30];
   char firstname[30];
   float salary;
};
```

وقتی ساختمان را تعریف می کنید می توانید متغیری از نوع آن را اعلان کنید تا مانند متغیرهای دیگر فضائی از حافظه برای آن کنار گذاشته شود. فضای مورد نیاز بستگی به اندازه ساختمان دارد. اندازه ساختمان با جمع کلیه نوع های داده اجزای آن محاسبه می شود.

دو راه برای تعریف متغیر ساختمان وجود دارد: بلافاصله بعد از تعریف ساختمان یا در محل دیگری از برنامه.

مثال. اعلان متغیر ساختمان MyEmployee از نوع employee بلافاصله بعد از تعریف ساختمان.

```
struct employee {
   char lastname[30];
   char firstname[30];
   float salary;
}MyEmployee;
```

مثال. اعلان متغیر ساختمان SecondEmployee از نوع ساختمان موجود employee.

```
struct employee SecondEmployee;
```

اگر ساختمان در جاهای مختلف برنامه استفاده می شود بهتر است در یک فایل هدر قرار بگیرد سپس در فایل اصلی ضمیمه شود.

دسترسی به اجزای ساختمان

با هر جزء ساختمان می تواند مشابه متغیرهای دیگر کار کرد. اجزای ساختمان با استفاده از نام متغیر ساختمان به همراه یک نقطه (.) و سپس نام عنصر ساختمان دسترسی می شود.

مثال. مقداردهی جزء salary از متغیر ساختمان MyEmployee.

```
MyEmployee.salary=4000.44;
```

مثال.

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

struct employee {
   char lastname[30];
   char firstname[30];
   float salary;
};

int main() {
   struct employee MyEmployee;
   strcpy(MyEmployee.lastname,"Adams");
   strcpy(MyEmployee.lastname,"Amy");
   MyEmploye.Salary=1000.00;

   cout << MyEmployee.lastname << MyEmployee.lastname << MyEmploye.Salary <<endl;
   return 0;
}
```

یکی از مزایای ساختمان نسبت به متغیرهای مجزا کپی داده های یک متغیر ساختمان در متغیر دیگر از همان نوع توسط یک عبارت انتساب است.

مثال. دستور زیر کلیه اجزای ساختمان MyEmployee را در ساختمان SecondEmployee کپی می کند.

```
SecondEmployee = MyEmployee;
```

مقداردهی اولیه ساختمان

مشابه متغیرهای دیگر، متغیر ساختمان را می توان هنگام اعلان مقداردهی اولیه کرد. روند انجام این کار مشابه آرایه است؛ بعد از نام متغیر علامت مساوی و بدنبال آن مقادیر اجزا به ترتیب داخل آکولاد نوشته می شوند.

مثال. مقداردهی اولیه ساختمان کارمند.

```
struct employee {
   char lastname[30];
   char firstname[30];
   float salary;
}MyEmployee ={ "Adams",
               "Amy",
               1000.00
              };
```

MyEmployee

lastname	Adams
firstname	Amy
salary	1000.00

معمولا برنامه ها با چندین رکورد داده ای کار می کنند. آرایه ای از ساختمان در برنامه های مختلف مفید است.

مثال. برنامه زیر برای ذخیره اطلاعات تلفن افراد است. آرایه list از نوع ساختمان entry است.

```
#include <iostream.h>

struct entry {              // Define a structure to hold entries.
   char fname[20];
   char lname[20];
   char phone[10];
};

struct entry list[4];       // Declare an array of structures.

main(){
   for (int i = 0; i < 4; i++) {
      cout << "\nEnter first name: ";
      cin >> list[i].fname;
      cout << "Enter last name: ";
      cin >> list[i].lname;
      cout << "Enter phone in 123-4567 format: ";
      cin >> list[i].phone;
      }

   cout << "\n\n";

   for (int i = 0; i < 4; i++){
      cout << "Name:" << list[i].fname << list[i].lname;
      cout << "\t\tPhone: " << list[i].phone;
      }

   return 0;
}
```

اجزای ساختمان می توانند از هر نوع داده ای باشند مانند آرایه یا حتی ساختمان.

مثال. برنامه زیر مختصات گوشه های مستطیل را گرفته مساحت آنرا محاسبه می کند. ساختمان rectangle شامل دو جز از نوع ساختمان coord است.

```
#include <iostream.h>

struct coord {
    int x; int y;
};

struct rectangle {
    struct coord topleft;
    struct coord bottomrt;
} mybox;
```

```
main() {
    int length, width;
    long area;

    cout << "\nEnter the top left x coordinate: ";
    cin >> mybox.topleft.x;
    cout << "\nEnter the top left y coordinate: ";
    cin >> mybox.topleft.y;
    cout << "\nEnter the bottom right x coordinate: ";
    cin >> mybox.bottomrt.x;
    cout << "\nEnter the bottom right y coordinate: ";
    cin >> mybox.bottomrt.y;

    width = mybox.bottomrt.x - mybox.topleft.x;
    length = mybox.bottomrt.y - mybox.topleft.y;

    area = width * length;
    cout << "\nThe area is " << area << " units.\n";

    return 0;
}
```

نکته. تعریف نوع ساختمان را با متغیری از ساختمان اشتباه نگیرید.
نکته. دقت کنید در انتهای تعریف ساختمان علامت سمیکولن فراموش نشود.
نکته. یک ساختمان می تواند به تابع ارسال شود. ارسال ساختمان یه تابع زمانی مفید است که بخواهیم چند مقدار را به تابع ارسال کنیم.
نکته. نوع برگشتی تابع می توان ساختمان باشد. تابع همیشه یک مقدار را برمی گرداند اگر نیاز باشد چند مقدار را برگرداند یک ساختمان می تواند بکار بیاید.

مثال. برنامه ای برای محاسبه سینوس، کسینوس و تانژانت زاویه. پارامتر و نوع برگشتی تابع compute هردو از نوع ساختمان هستند.

```
#include <cmath.h>
#include <iostream.h>
struct geometry_data {
  float radius;
  double angle;
};
struct geometry_answers {
  float area;
  double sine;
  double cosine;
  double tangent;
};
geometry_answers compute(struct geometry_data mystruct);

int main () {
    geometry_data input;
    geometry_answers output;

    cout << "Enter the radius of the circle \n";
    cin >> input.radius;
    cout << "Enter the angle in rads \n";
    cin >> input.angle;

    output = compute(input);

    cout << " The area is "<< output.area << "\n";
    cout << " The sine of the angle is " << output.sine << "\n";
    cout << " The cosine of the angle is " << output.cosine << "\n";
    cout << " The tangent of the angle is " << output.tangent << "\n";
    return 0;
}
```

```
geometry_answers compute(struct geometry_data mystruct) {
     geometry_answers answer;

     answer.area = 3.14f * pow(mystruct.radius,2);
     answer.sine = sin(mystruct.angle);
     answer.cosine = cos(mystruct.angle);
     answer.tangent = tan(mystruct.angle);
     return answer;
};
```

Typedef

typedef راهی است برای تعریف برچسبهای جدید برای نوعهای داده است. هدف typedef ایجاد نوع جدیدی از انواع داده موجود است. به صورت کلی زیر استفاده می شود:

```
typedef  datatype label;
```

label برچسب جدیدی است که به نوع داده datatype داده می شود و معمولا حروف اول آن بزرگ است تا از نوع های پیش ساخته تفکیک شود.

typedef می تواند با هر نوع داده ای بکار رود حتی نوع های پایه مثل int. البته توصیه می شود با نوع های پیچیده مانند آرایه یا ساختمان استفاده شود.

مثال. تغییر نام int به Integer.

```
typedef  int Integer;
int i,j;
Integer min;
```

مثال. اعلان a و b از نوع آرایه صحیح.

```
typedef int MyArray[10];
MyArray a,b;
```

مثال. اعلان متغیرهای topleft و bottomright از نوع ساختمان. دقت کنید با استفاده از typedef کلمه struct قبل از اسم متغیر دیگر لازم نیست ذکر شود.

```
typedef struct {
    int x;
    int y;
} Coord;
Coord topleft, bottomright;
```

یونیون

یک یونیون (union) مجموعه ای از چند متغیر است که تحت یک نام گروه بندی می شوند و از یک فضای حافظه بطور مشترک استفاده می کنند. یک یونیون مشابه ساختمان تعریف و استفاده می شود فقط به جای کلمه struct کلمه کلیدی union نوشته می شود.

```
union tag
{
   union_member(s);
} instance;
```

کلمه کلیدی union برای اعلان یونیون است. tag نامی است که به یونیون داده می شود. اجزای یونیون درون آکولاد قرار می گیرند. instance یک متغیر یونیون است که می تواند درون برنامه هم با فرمت زیر اعلان شود.

```
union tag instance;
```

علت اینکه این نوع داده یونیون نام دارد اینستکه چند نوع داده را با هم متحد می کند. کلیه اجزای یونیون از یک ناحیه حافظه به صورت مشترک استفاده می کنند بنابراین درهرلحظه فقط یک جزء را می توان استفاده کرد و بطور همزمان نمی توان از این متغیرها استفاده کرد. اندازه یونیون به اندازه بزرگترین جزء آن است.

مثال.

```
union NumericType
{
   int ivalue;
   long lvalue;
   double dvalue;
}
```

مثال. تنها اولین جزء متغیر یونیون هنگام اعلان می تواند مقداردهی اولیه شود.

```
union date_tag {
    char full_date[9];
    struct part_date_tag {
        char month[2];
        char break_value1;
        char day[2];
        char break_value2;
        char year[2];
    } part_date;
}date = {"01/01/97"};
```

اجزای یونیون مشابه ساختمان توسط عملگر (.) دسترسی می شوند. دقت کنید در هر لحظه با کدام جزء دارید کار می کنید. اگر یک جزء را مقداردهید و از جزء دیگر استفاده کنید نتایج غرقابل پیش بینی خواهد بود.

شمارشی

یک داده شمارشی (enumeration) در واقع مجموعه ای از ثابت های عددی صحیح است که کلیه مقادیری که متغیرهای از این نوع می توانند داشته باشند را مشخص می کند. فرم کلی نوع شمارشی به صورت زیر است:

```
enum typename { enumeration list };
```

کامپایلر C به هر یک از عناصر نوع شمارشی عددی را نسبت می دهد که از صفر شروع می شود. از آنجائی که یک متغیر شمارشی مقادیر مجاز در محدوده اعداد صحیح را می پذیرد به اندازه ۲ بایت فضا اشغال می کند.

مثال.

```
enum colors {red, blue, green};
enum colors c;

c= red;
cout << c;
```

فیلدهای بیتی

به طور نرمال در هر زبان برنامه نویسی هر متغیری از نوع های شناخته شده است و تعداد بایت های معینی می گیرد. C++ یک نوع دیگر ارائه می دهد که در اکثر زبان های دیگر موجود نیست. فیلدهای بیتی (bit fields) اجازه دسترسی به بیت های تکی را می دهد.

یک فیلد بیتی در یک ساختمان عادی به صورت یک جزء unsigned به همراه علامت (:) و یک عدد که نشاندهنده تعداد بیت های فیلد است تعریف می شود.

مثال. متغیر linestatus یک بایت فضا اشغال می کند.

```
#include <iostream.h>

struct status {
      unsigned changeinline: 1;
      unsigned cleartosend:1;
      unsigned inactive:1;
      unsigned ringing:1;
      unsigned signalreceived:1;
};

int main() {
    status linestatus;

      if (linestatus.cleartosend)    senddata();
      if (linestatus.inactive)       dialnumber();
      if(linestatus.ringing)         answerphone();
      return 0;
}
```

نکته : اشاره گر به فیلد بیتی نمی توان داشت.

نکته. فیلدهای بیتی از نوع آرایه یا کلاس static نمی توانند تعریف شوند.

نکته. فیلدهای بیتی اجازه ارتباط مستقیم با زبان سخت افزار و خطوط ارتباطی را فراهم می آورند. این دستگاه ها داده ها را به صورت رشته ای ازبیت های منفرد می فرستند.