برنامه سازی شی گرا

فهرست مطالب

مقدمه ای بر برنامه نویسی

زبان های برنامه نویسی

به طور دقیق، برنامه نویسی کامپیوتر شامل تمام فعالیت های در گیر در ایجاد و حفظ یک کد منبع، از قبیل طراحی، نوشتن، تست و اشکال زدایی کد منبع از هر برنامه می‌باشد. یک برنامه کامپیوتری مجموعه ای از دستور العمل ها است، به گونه ای که برای انجام یک مجموعه خاصی از وظایف، توسعه یافته و نوشته شده است. این دستور العمل ها، شامل مشخصات و عمل کردهای مورد نیازی هستند که برای ایجاد یک برنامه نوشته می شوند و در نهایت یک کد منبع برنامه را تشکیل می دهند.

یک زبان برنامه نویسی عبارت است از توضیح، اندازه گیری یا بیان فرایندها یا الگوریتم هایی که شامل داده های دیجیتال و هم چنین داده های غیر دیجیتال باشد و بتواند توسط ماشین های محاسبه اجرا شود. زبان های برنامه نویسی برای تعداد زیادی از اهداف ایجاد شده اند، اما منطق اولیه ایجاد همه آنها مشترک است. آنها برای کنترل و تنظیم رفتار و عمل کرد ماشین های محاسباتی با توجه به خروجی مورد نظر خود به کار می روند. تمام زبان های برنامه نویسی دارای دو جزء اصلی می باشند - صرف و نحو. نحو یا syntax شکل و ترتیب نمادها و کاراکترها در یک زبان خاص است. معنا شناسی یا Semantic با معنی و مفهوم مجموعه ای از کاراکترها که به طرز خاصی مرتب شده اند، همراه است.

به طور کلی، دو نوع اساسی از زبان های برنامه نویسی وجود دارند که بر اساس سطح انتزاعی معماری مجموعه دستورالعمل-های کامپیوتر می باشند. این دو نوع از زبان های برنامه نویسی، زبان سطح بالا و زبان سطح پایین می باشند. زبان های سطح بالا بسیار به سینتکس های ماشین شباهت دارند و بیشتر این زبان ها از عناصر زبان جهان واقعی استفاده می نمایند. به این ترتیب، زبان های سطح بالا کار بر پسند تر بوده و قابلیت انتقال بیشتری در سیستم عامل های مختلف ارائه می دهند. از مشخصه های این زبان ها عدم رؤیت آشکار جزئیات عملیات CPU از قبیل مدیریت دامنه و مدل دسترسی به حافظه می-باشد. به منظور درک و پردازش ورودی از کار بر توسط ماشین محاسباتی و تبدیل آن به زبان سطح بالا، یک مفسر، کامپایلر و یا مترجم به کار گرفته می شود تا آن را به یک شکل اجرایی تبدیل کند. از نمونه های متداول و رایج زبان سطح بالا می توان جاوا، C و++C را نام برد. زبان سطح پایین از نظر فرم آن بسیار نزدیک به کد ماشین داخلی کامپیوتر است و در نتیجه، به راحتی و به آسانی توسط کامپیوتر بدون نیاز به برنامه واسط مانند مفسر، کامپایلر و مترجم اجرا می

گردد. یکی از مشهورترین زبان های سطح پایین زبان اسمبلی می باشد. به نوعی می توان بیان کردزمانی که شما یک قطعه برنامه را در زبانی مانند C++ می نویسید آنگاه این برنامه تبدیل به زبان سطح پایینی همچون اسمبلی شده و آنگاه ماشین این زبان را خواهد فهمید و شروع به اجرای آن خواهد کرد. در زیر یک حلقه For به زبان پاسکال و C++ بعنوان زبان های سطح بالا نوشته شده و در ادامه همین حلقه به زبان اسمبلی (زبان سطح پایین یا زبان ماشین) هم ارائه گردیده است.

در پاسکال:

```
for i:= 1 to 9 do
        write(i);
```

در C++:

```
for( int a = ١; a < ٩; a++ )
    cout << i ;
```

در اسمبلی:

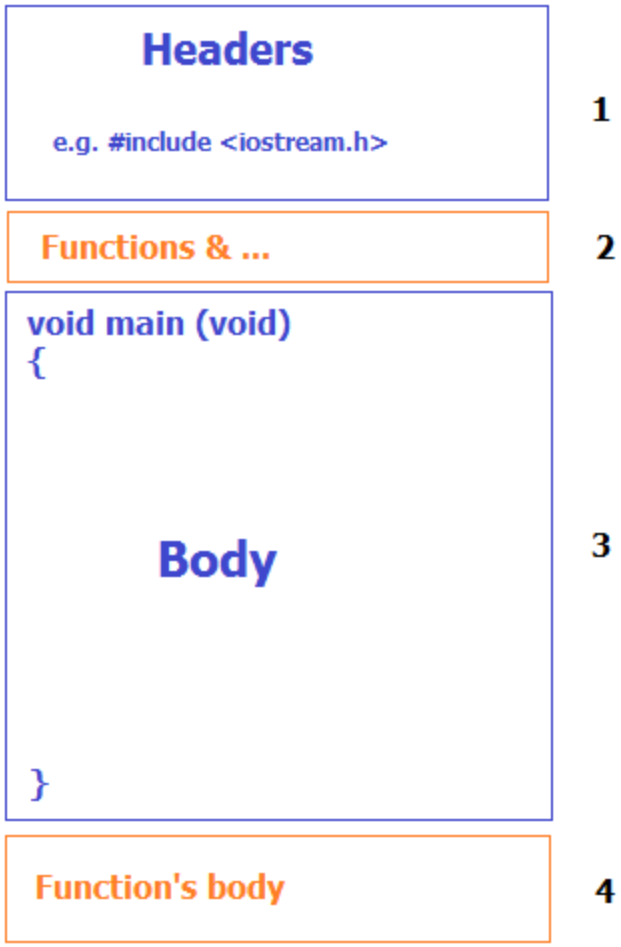
```
section
  global _start

_start:
  mov ecx,10
  mov eax, '1'

l1:
  mov [num], eax
  mov eax, 4
  mov ebx, 1
  push ecx
```

```
  mov ecx, num
  mov edx, 1
  int 0x80

  mov eax, [num]
  sub eax, '0'
  inc eax
  add eax, '0'
  pop ecx
  loop l1

  mov eax,1             ;system call number (sys_exit)
  int 0x80              ;call kernel
section         .bss
num resb 1
```

لذا میبینیم که چقدر جزئیات یک برنامه با تبدیل شدن از زبان سطح بالا به زبان ماشین بیشتر می شود.

مقدمات برنامه نویسی در C++

بطور خلاصه هر برنامه از فرمت و قالب کلی زیر پیروی می کند:



در قسمت شماره ۱ کتابخانه ها قرار می گیرند. از آنجاکه برنامه ها بزرگتر و بزرگتر می شوند برای مدیریت بهتر و سرعت پردازش بالاتر کتابخانه ها را ایجاد کردند که در واقع هر یک از کتابخانه ها خود مجموعه بسیار بزرگی از توابع از پیش تعریف شده هستند که احتمالا در حوزه ی نسبتا مشابهی کار می کنند.

در قسمت شماره ۲ توابعی که برنامه نویس خود می نویسد و در برنامه استفاده می کند تعریف می شوند که همین توابع باید بصورت مجزا در قسمت شماره ۴ بطور کامل نوشته شوند که در آینده مفصل تر به توضیح توابع خواهیم پرداخت.

همچنین در قسمت شماره ۳ در واقع بدنه اصلی برنامه قرار می گیرد و برنامه از این قسمت اجرا می شود.

دستورات ورودی خروجی

Cin

این دستور در واقع مقداری را از کاربر بعنوان ورودی دریافت می کند. این مقدار می تواند یه عدد، رشته، کاراکتر و ... باشد. این دستور معمولا به همراه یک متغیر می آید که مقدار مربوطه در آن متغییر ذخیره شود و بصورت زیر نمایش داده می شود:

```
cin>>a
Cout
```

این دستور برعکس دستور قبلی مقداری را به خروجی می فرستد. در اینجا خروجی منظور همان صفحه نمایش است.

```
cout<<a;
cout<<"Helloooo";
```

نوع داده ها

در C++ شش نوع داده وجود دارد. منظور از داده، متغیری است که در قالب متن یا عدد در طول برنامه مورد استفاده قرار می گیرد.

داده های موجود در C++ عبارتند از:

```
char, int, float, double, void, bool ,string
```

نوع char برای ذخیره داده های کاراکتری مانند 'W' , 'z' , 'a' : بکار می رود.

از نوع int برای ذخیره اعداد صحیح مانند ۱۲۸، ۵، ۴۵۰۸ استفاده می شود.

نوع float برای ذخیره اعداد اعشاری مثل ۱۲.۵، ۷۸۰۵.۱۱ بکار می رود.

نوع double برای اعداد اعشاری بزرگتر از float استفاده می شود.

نوع voi d هیچ مقداری را نمی گیرد

نوع دیگری از داده وجود دارد که برای استفاده از رشته ها مورد استفاده قرار میگیرد که stri ng گفته میشود اما در برخی از نسخه های کامپایلر زبان برنامه نویسی C++ پشتیبانی نمی شود، لذا مجبور به استفاده از آرایه ای از کاراکترها برای این منظور خواهیم بود .

نوع bool
متغیر های این نوع فقط می توانند مقدار true یا fal se را ذخیره کنند که به ترتیب معادل ۱ و ۰ می باشند.

از بارزترین موارد استفاده از این نوع داده می توان به تخصیص آن برای تعیین جنسیت کاربران، فعال یا غیر فعال بودن وضعیت کاربران و غیره.

```
bool status = false;
```

ثوابت

ثابت ها یا Constants مقادیر بدون تغییر تلقی می شوند. در بعضی از برنامه ها از متغیری استفاده میکنیم که فقط یکبار لازم است آن را مقدار دهی کنیم و سپس مقدار ان متغیر در سراسر برنامه بدون تغییر باقی می ماند. مثلا در یک برنامه محاسبه ریاضی متغیری بنام P تعریف میکنیم و آن را با 3.14 مقدار دهی میکنیم و می خواهیم که مقدار این متغیر در سراسر برنامه ثابت بماند. در چنین حالاتی از ثابت ها استفاده میکنیم. یک ثابت، یک نوع متغیر است که فقط یکبار مقدار دهی می شود و سپس **تغییر دادن** مقدار آن در ادامه برنامه **ممکن نیست**.

تعریف ثابت ها مانند تعریف متغیر هاست با این تفاوت که کلمه کلیدی **const** به ابتدای تعریف اضافه می شود. پس دستور int i=3 متغیری را از نوع مقدار عددی صحیح تعریف میکند که در طول برنامه قابل تغییر خواهد بود ولی بصورت ثابت آن که در زیر نوشته شده است قابل تغییر نخواهد بود:

```
const float p=3.1415;
```

```
#include <iostream>
```

```
int main()
{
   const  float  P = 3.14 ;
   cout << P;
   return  0 ;
}
```

ثابت ها را باید موقع تعریف یکبار برای همیشه مقدار دهی کرد.

عملگرها

برای انجام عملیات بر روی داده ها از عملگرها استفاده می کنیم. عملگرها نمادهایی هستند که عملیاتی مانند جمع،ضرب، کوچکتری و از این قبیل را روی داده ها انجام می دهند که عبارتند از:

عملگرانتساب

(Assignment) (=)

از این عملگر برای نسبت دادن یک مقدار به یک داده استفاده می شود .

پنج عملگر محاسباتی موجود در C++عبارتند از:

+	جمع
-	تفریق
*	ضرب
/	تقسیم
%	باقیمانده تقسیم

عملگرهای ترکیبی

(Compund Operators) (=+ , =- , =* , =/)

عبارت	برابر است با
a += b	a=a+b
a -= b	a=a-b
a *= b+1	a=a*(b+1)
a /= b	a=a/b

عملگرهای افزایش کاهش

(-- , ++) (Increase , Decrease)

عملگرهای رابطه ای و تساوی

(Relational and equality operators) (= = , =! , > , < , =< , =>)

از این نوع عملگرها برای مقایسه دو عبارت استفاده میشود که کاربرد اونها بیشتر در عبارات شرطی است که بعدا در موردشون بحث می کنیم . فعلا اینو بدونید که این عملگرها در صورت درست بودن مقایسه، مقدار درستی و در غیر این صورت مقدار نادرستی را برمی گردانند.

شرط ها

در هر زبان برنامه نویسی یکی از ابزار لازم و ضروری استفاده از شرط ها می باشند. در واقع هر زمان که مقایسه یا اعمال شرط نیاز باشد استفاده می شود. کلمه کلیدی برای آن i f می باشد و بصورت زیر استفاده می شود:

```
if (شرط مساله){

 }
```

بعبارتی هر زمانکه شرط مساله برقرار و یا TRUE باشد کد های داخل آکولاد اجرا می شوند. برنامه زیر دو عدد از ورودی می خواند و پس از مقایسه نشان می دهد کدامیک بیشینه و دیگری کمینه می باشد:

```
#include “iostream.h”
Void main(void){
        int a,b;
        cin>> a >> b;
        if (a>b){
```

```
        cout<<"Maximum is:" << a <<" and Minimum is:" << b;
    }
    if (b>a){
        cout<<"Maximum is:" << b <<" and Minimum is:" << a;
    }
}
```

در الگوریتم بالا یک ایراد وجود دارد که اشتباه نیست اما بهتر است بهبود یابد. فرض کنید شرط اول برقرار باشد دیگر نیازی به بررسی شرط دوم نیست لذا نسخه کامل تر شرط ها به این شکل می باشد که هربار فقط یک قسمت اجرا شود و شرط دوم را در داخل else گذاشته می شود:

```
If (){

}else{

}
```

Switch

این دستور ابزاری است برای ربط دادن جواب های خاص و یا از پیش تعیین شده به حالاتی مه مد نظر ما م باشد. بعنوان مثال از کاربر می خواهیم که یک عدد فرد را وارد کند، نمونه جواب های ما مشخص می باشند. لذا می توانیم هر عدد را به حالت نوشتاری آن ربط دهیم و در نهایت بجای هر عددی که کاربر وارد کرد نوع نوشتاری آن را چاپ کنیم:

```
Switch( a ){
    Case 0: cout<< "Zero";
    Case 1: cout<<"One";
    . . .
}
```

تمرین 1

- برنامه ای بنویسید که عددی از ۱ تا ۷ را خوانده ، روزی از هفته را که معادل با آن است را در خروجی به حروف چاپ کند (switch).

- برنامه ای نوشته که یک عدد را بعنوان روز سال از کاربر گرفته و محاسبه کند این عدد چه روزی از هفته (چند شنبه) می باشد. فرض بر این است که روز شروع هفته همان شنبه در نظر گرفته شود. باقیمانده با ٪ بدست می آید).

عملگر ها

< =! => && ||

شرط هایی مانند n٪d و x>=y می توانند به صورت یک شرط مرکب با هم ترکیب می شوند. این کار با استفاده از عملگر های منطقی&& (and) و || (or) و ! (not) صورت می پذیرد. این عملگر ها به شکل زیر تعریف می شوند:

P &q درست است اگر و تنها اگر هم p و هم q درست باشند.

P || q نادرست است اگر و فقط اگر هردو نادرست باشند. بعبارتی اگر حتی فقط یکی از آنها هم درست باشد کل عبارت درست خواهد بود.

!p

حلقه ها

بطور کلی، در هر برنامه نویسی اجرای دستورات از اولین سطر شروع شده و به ترتیب تا آخرین سطر ادامه میابد. اما گاهی وقتها لازم است که یک دستور چندین بار تکرار شود و یا اینکه تحت شرایط خاصی اجرا گردد و یا از اجرای آن جلوگیری شود.

ساختارهای کنترلی (if-else) که قبلا توضیح داده شده اند به برنامه نویس این اجازه را می دهد که بر روی دستورات کنترل داشته باشد و آنها را تکرار، اجرا و متوقف سازد.

حلقه‌ی تکرار while

این نوع حلقه ساده‌ترین نوع حلقه‌ی تکرار در این زبان برنامه‌نویسی است. فرم کلی حلقه‌ی whi l e به این صورت است:

```
while(شرط اجرای حلقه){
   دستورات داخل حلقه
}
```

عبارت‌های داخل حلقه تا زمانی که شرط اجرای حلقه صحیح باشد اجرا خواهند شد.

```
int n = 1;
while(n <= 10){
  cout << n << endl;
  n++;
}
```

در این قطعه کد، ابتدا متغیر n با عدد یک مقداردهی می‌شود. سپس شرط $n \leq ۱۰$ بررسی می‌شود که صحیح است. پس اجرای قطعه کد با دستورات داخل حلقه ادامه پیدا می‌کند. در این حلقه مقدار n چاپ شده و یک واحد به آن اضافه می‌شود. سپس کنترل برنامه به ابتدای حلقه باز می‌گردد. اگر شرط حلقه همچنان صحیح باشد، عبارت‌های داخل آن مجددا اجرا خواهند شد. در نتیجه قطعه کد فوق اعداد یک تا ده را به ترتیب در سطرهای جداگانه‌ی خروجی چاپ خواهد کرد.

حلقه‌ی تکرار do-while
فرم کلی این حلقه به صورت زیر است:

```
do{
   دستورات داخل حلقه
}while(شرط اجرای حلقه);
```

تنها تفاوت این حلقه با حلقه‌ی while در این است که شرط اجرای حلقه‌ی do-while در انتهای آن بررسی می شود. به عنوان مثال:

```
int n = 1;
do{
cout << n << endl;
```

```
 n++;
}while(n <= 10);
```

در این قطعه کد نیز همانند قطعه کد قبلی اعداد یک تا ده در خروجی چاپ می‌شوند.

همانگونه که عنوان شد، در حلقه‌ی while شرط اجرای دستورات داخل حلقه در ابتدای آن بررسی می‌شود. اما در حلقه ی do-while این شرط در انتهای آن قرار دارد. در نتیجه دستورات داخل حلقه قبل از رسیدن به شرط حلقه یک بار اجرا می‌شوند. به طور خلاصه می‌توان گفت: تفاوت حلقه‌ی do-while با حلقه‌ی while در این است که دستورات داخل حلقه‌ی do-whileحداقل یک بار اجرا می‌شوند. بیشتر کاربردهای چنین حلقه‌ای هم به خاطر همین خاصیت آن است.

مثال: برنامه ای بنویسید که سه عدد را بعنوان نمرات سه درس از کاربر گرفته و معدل را نمایش دهد. از آنجاکه یک نمره درسی بین صفر و بیست می باشد لذا باید کاربر را طوری کنترل کرد که تا وقتی نمرات وارد شده معتبر نمی باشند مجددا درخواست ورود نمره کند!

```
#include <iostream.h>
int main(void)
{
	int a, b, c = -1;  //مقادیر پیش فرض هر سه متغیر -1 داده میشود.
	float sum =0;
	do{
		if (a == -1)
			cin>>a;
		else
			if (b == -1)
				cin>>b;
			else
				if (c == -1)
					cin>>c;
```

```
}While( (a == -1) || (b == -1) || (c == -1) )
sum = a + b + c;
cout <<"Average is:"<<sum/3;

Return 0;
}
```

این الگوریتم به این شکل کار می کند که تا وقتی مقدار a صحیح نمی باشد حلقه دور زده چک میکند که آیا کماکان مقدار آن 1- میباشد! و زمانیکه مقدار صحیحی برای a وارد شد آنگاه حلقه سراغ متغر b می رود و تا گرفتن مقدار معتبر برای متغیر b ادامه می دهد.

حلقه‌ی تکرار for

ساده‌ترین نوع تعریف حلقه‌ی for به این ترتیب است:

```
for (مقداردهی اولیه ; شرط اجرای حلقه ; رشد شمارنده){
دستورات داخل حلقه
}
```

با اجرای خط اول، متغیر n تعریف می‌شود. سپس بخش «مقداردهی اولیه» اجرا شده و مقدار n برابر عدد یک می‌شود. پس از آن «شرط اجرای حلقه» بررسی می‌شود. این شرط همانند شروط حلقه‌های قبلی عمل کرده و در صورت نادرست بودن کنترل برنامه از حلقه خارج می‌شود. اما اگر شرط صحیح باشد کنترل برنامه وارد حلقه شده و دستورات داخل آن اجرا می‌شوند. در اجرای بعدی بخش «مقداردهی اولیه» اجرا نمی‌شود. اما قبل از بررسی «شرط اجرای حلقه»، عملیات بخش «نمو» اجرا می‌شوند. همانگونه که شرح داده شد، این عملیات در اجرای اول و زمان ورود به حلقه اجرا نمی‌شوند. پس از نمو، شرط اجرای حلقه بررسی شده و به همین ترتیب اجرای برنامه ادامه پیدا می‌کند.

مثال: برنامه ای بنویسید که اعداد حاصل جمع اعداد زوج کوچکتر از 1000 که بر 7 و 3 بخش پذیر می باشند را محاسبه و نمایش دهد.

```
#include<iostream>
int main(void)
```

```
{
        Int sum=0;
        For (int i=2; i<=1000; i++)
        {
                if ( (i % 2) == 0  && (i % 7) == 0  && (i % 3) == 0 )
                        Sum += I;
        }
        cout<< sum;

        return 0;
}
```

مثال: برنامه ای نوشته که یک مقدار را از کاربر گرفته و فاکتوریل آن را محسابه و نمایش دهد.

```
#include<iostream.h>
int main(void)
{
        long int sum = 1;
        int n, cnt=1;
        cin>>n;
        while(cnt <= n)
`       {
                sum = sum * cnt;
                cnt++;
        }
        cout<<sum;
        return 0;
}
```

توابع

در این زبان برنامه نویس این قابلیت را خواهد داشت که بخش هایی از برنامه اصلی را در قالب یک برنامه پیاده سازی کند. این توابع می توانند به اختیار دارای ورودی و یا خروجی هایی باشند. از فواید نوشتن توابع اینست که به برنامه نویس کمک می کند که نیازی به تکرار یک قطعه کد نباشد و یا اینکه از یک قطعه کد نوشته شده به دفعات استفاده کند.

تابع در حقیقت زیر برنامه ایست که می تواند بر روی داده ها عمل کند و مقداری برگرداند. هر برنامه حداقل یک تابع ()main دارد. وقتی برنامه شما اجرا می شود این تابع بطور خودکار فراخوانی می شود. این تابع خود می تواند سایر توابع نوشته شده را احضار کند. توابع به دو دسته تقسیم می شوند: (1) توابع داخلی (2) توابع نوشته شده کاربر. از جمله توابع داخلی می توان به تابع جذر و مکعب اشاره کرد که در گروه توابع ریاضیاتی می باشند.

تابع جذر sqrt()

تابع مکعب cube()

فرم کلی توابع بصورت زیر می باشد:

```
(لیست پارامتر های تابع) نام تابع   مقدار خروجی تابع
{


}
```

مقدار خروجی تابع همان مقداری است که در نهایت تابع نوشته شده آن را بر می گرداند که البته اجباری نیست و ممکن است یک تابع همه عملیات پردازشی را در درون خود انجام داده و همانجا چاپ کند. لیست پارامتر های تابع متغیر هایی هستند که تعریف می شوند به این منظور که مقداری را از بیرون به درون تابع پاس بدهند. لیست پارامتر های واقعی که در حین عملیات برای تابع ارسال می شوند **آرگومان های تابع** گفته می شوند.

توابع ابتدا باید قبل از شروع تابع ()main تعریف شوند و پیاده سازی آن ها باید بعد از اتمام تابع ()main انجام شود. بصورت زیر:

```
#include <iostream>
```

```
int power(int, int);
void main(void)
{
        بدنه تابع اصلی
}
int power(int a, int b)
{
        بدنه تابع
}
```

برای مثال به تابع زیر دقت کنید:

این تابع دو مقدار را بعنوان طول و عرض یک مستطیل از کاربر دریافت کرده و آنگاه پس از محاسبه مساحت مقدار آن را برمی گرداند:

```
float Area (float length, float width)
{
        float sum;
        sum = length * width;
        return sum
}
```

از آنجا که حاصل مساحت مقداری اعشاری است لذا باید مقدار خروجی نهایی تابع هم اعشاری باشد.

حال تابعی می نویسیم که دو عدد را بعنوان پایه و توان دریافت کرده و مقدار آن را محاسبه و ارجاع دهد.

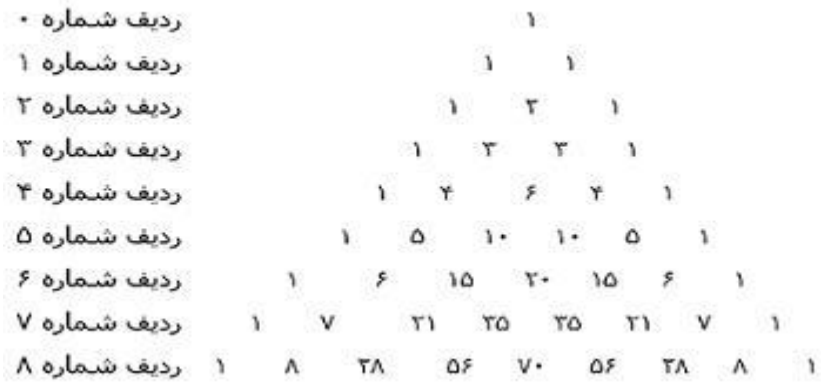
بعنوان مثال عدد 2^5 را در قالب دو عدد ۲ و ۵ دریافت کرده و سپس مقدار نهایی را بر می گرداند:

```
long int power(int a, int b)
{
        long int sum = 1;
for (int i=1; i<=b; i++)
        {
                Sum *= a;
        }
}
```

دقت کنید که متغیرsum بصورت اولیه ۱ داده می شود چون قرار است مقدار a که هر بار در خودش ضرب مشود در آن ضرب و ذخیره شود و چون عنصر بی خاصیت ضرب ۱ می باشد پس مقدار اولیه آن ۱ داده می شود. همچنین لازم به ذکر است که عنصر خنثی جمع هم صفر می باشد.

تمرین 2

تمرین ۱: مثلث خیام پاسکال را نمایش دهید.



تمرین ۲: تابع di gi t (i nt n, i nt k) پیاده سازی شود به این صورت که اگر عدد ۳۸۵۲۴ داده شود و di gi t (n, ۳) خواسته شود مقدار ۸ برگردانده شود.

تمرین ۳: تابعی نوشته که یک عدد داده شده توسط کاربر را بررسی کند که آیا مربع می باشد یا نه! مانند اعداد ۲ و ۹ و یا ۱۶.

تمرین ۴: برنامه ای نوشته که کاربر مقدار n را وارد کرده و یک تابع فراخوانده شود و حاصل سری زیر را محاسبه و نمایش دهد:

$$1+\frac{1}{2}+\frac{1}{3}+\ldots+\frac{1}{n}$$

تمرین 5: برنامه ای نوشته که همانند بالا سری مشابه را جمع و محاسبه کند با این تفاوت که مخرج هر کسر بطور مجزا به توان 2 برسد و آنگاه حاصل را محاسبه و نمایش دهد.

تمرین 6: برنامه ای بنویسید که جدول ضرب 10 * 10 را نمایش دهد.

تمرین 7: برنامه ای بنویسید که شکل زیر را چاپ کند. تعداد طبقات شکل مورد نظر توسط کاربر وارد شود که در این مثال تعداد طبقات 4 می باشد.

```
   *
  * *
 * * *
* * * *
```

آرایه ها

آرایه مجموعه ایست از عناصر هم نوع که برای ذخیره سازی مقادیر استفاده می شوند. در صورتی که به تعداد زیادی از مقادیر هم نوع نیاز داشته باشیم می توانیم از آرایه ها استفاده کنیم. تعریف آرایه به شکل زیر می باشد:

نوع آرایه نام آرایه [تعداد عناصر آرایه مورد نظر];

```
int A[5];
```

برای دسترسی به عناصر آرایه از اندیس استفاده می کنیم که اولین اندیس هر آرایه ای از عدد صفر شروع می شود .

گفتنی است که عناصر آرایه پشت سر هم در خانه های حافظه ذخیره می شوند و هر عنصر (خانه) از آرایه به اندازه طول نوع آرایه فضا اشغال می کند. در آرایه بالا چون نوع آرایه تعریف شده int است پس هر عنصر مقدار 5 بایت و چون طول آن 5 است در نهایت 20 بایت پشت سر هم از حافظه را اشغال می کند .

آرایه ها به یک بعدی و دو بعدی و چند بعدی تعریف می شوند که بر حسب نیاز برنامه نویس کاربرد خواهند داشت. در مثال زیر یک آرایه دو بعدی را خواهید دید:

```
int A[3][8];
```

از آرایه های دو بعدی می توان برای پیاده سازی جداول مانند جدول ضرب و یا ماتریس ها در ریاضیات استفاده کرد.

برای دسترسی به عناصر آرایه کافی است اندیس آن عنصر از آرایه را در درون [] قرار داد:

```
cout << A[2];
```
این خط مقدار عنصر سوم آرایه را برای ما چاپ می کند

مثال: برنامه ای بنویسید که جمع دو ماتریس را انجام دهد به نحوی که مقادیر ماتریس ها را هم کاربر وارد کند:

```
#include <iostream.h>
int main(void)
{
        int A[3][4], B[3][4], C[3][4];
                                                گرفتن مقادیر آرایه A از کاربر
        for(int i=0; i<=2; i++)
```

```
{
        for(int j=0; j<=3; j++)
        {
                cin>>A[i][j];
        }
}
```

گرفتن مقادیر آرایه B از کاربر

```
for(int i=0; i<=2; i++)
{
        for(int j=0; j<=3; j++)
        {
                cin>>B[i][j];
        }
}
```

جمع دو آرایه و ذخیره آنها در آرایه C

```
for(int i=0; i<=2; i++)
{
        for(int j=0; j<=3; j++)
        {
                C[i][j] = A[i][j] + B[i][j];
        }
}
```

چاپ کردن آرایه C در خروجی

```
for(int i=0; i<=2; i++)
{
        for(int j=0; j<=3; j++)
        {
                cout<<C[i][j]<<"  ";
        }
```

```
            cout<<"\n";
        }
        return 0;
}
```

تمرین 3

تمرین 1: برنامه ای بنویسد که ضرب دو ماتریس را انجام دهد به نحوی که مقادیر دو ماتریس را از کاربر دریافت کند.

تمرین 2: برنامه ای بنویسید که مقادیر یک آرایه را یک خانه رو به جلو شیفت دهد و عنصر آخر را به خانه اول شیفت دهد.

برنامه نویسی شی گرا

کلاس

کلاس روشی برای بسته بندی نوع داده مجرد است . در کلاس امکان بسته بندی و محصور کردن (Encapsulation) مجموعه ای از داده ها است . روال های پردازش کننده این داده ها را به صورت یک بسته فراهم می کند. داده های داخل یک کلاس به وسیله کلاس محافظت می گردد. به گونه ای که پردازش داده های خصوصی یک کلاس از طریق روال های دخلی آن امکان پذیر است . داده های یک کلاس را متغیرهای کلاس و روال های آن را روش نامیده اند. برای مثال کلاس انسان ها یک کلاس قابل تعریف است. در این کلاس خصوصیات مشترک انسانها تعریف می گردد و هیچ انسان خاصی را نشان نمی دهد. کلاس یک نوع است. همانگونه که مثلا int یک نوع است. عملیات محاسباتی (یا غیر محاسباتی) بر روی نوع داده انجام نمی شود. بلکه این عملیات بر روی متغیرهایی که از این نوع داده تعریف می گردد انجام می شود. به طور مشابه عملیات محاسباتی (یا غیر محاسباتی) روی کلاس انجام نمی شود.

شی

دیدگاه شی گرایی (Object Oriented) از اواسط دهه ۱۹۷۰ تا اواخر ۱۹۸۰ در حال مطرح شدن بود. در این دوران تلاشهای زیادی برای ایجاد روشهای تحلیل و طراحی شی گرا صورت پذیرفت. در نتیجه این تلاشها بود که در طول ۵ سال یعنی ۱۹۸۹تا ۱۹۹۴ ، تعداد متدلوژیهای شی گرا از کمتر از ۱۰ متدلوژی به بیش از ۵۰ متدلوژی رسید. تکثیر متدلوژیها و زبانهای شی گرایی و رقابت بین اینها به حدی بود که این دوران به عنوان "جنگ متدلوژیها " لقب گرفت. از جمله متدلوژیهای پر کاربرد آن زمان می توان از Shlayer- Mellor، Coad-yourdan، Fusion، OMT، OOSE، Booch و غیره نام برد. فروانی و اشباع متدلوژیها و روشهای شی گرایی و نیز نبودن یک زبان مدلسازی استاندارد ، باعث مشکلات فروانی شده بود . از یک طرف کاربران از متدلوژیهای موجود خسته شده بودند ، زیرا مجبور بودند از میان روشهای مختلف شبیه به هم که تفاوت کمی در قدرت و قابلیت داشتند یکی را انتخاب کنند . بسیاری از این روشها ، مفاهیم مشترک شی گرایی را در قالبهای مختلف بیان می کردند که این واگرایی و نبودن توافق میان این زبانها ، کاربران تازه کار را از دنیای شی گرایی زده می کرد و آنها را از این حیطه دور می ساخت . عدم وجود یک زبان استاندارد ، برای فروشندگان محصولات نرم افزاری نیز مشکلات زیادی ایجاد کرده بود.

پیدایش شی گرایی

برنامه نویسی شی گرا در اوایل دهه ۱۹۷۰ توسط آلن کی Alan Kay طراحی شده یعنی اولین قدمهای این سبک برنامه نویسی توسط آلن کی برداشته شده است. اولین زبان شی گرا توسط آلن کی طراحی شد. اسم این زبان Small Talk است. آلن کی گفته بود که : آن چیزی که باعث شد این فکر به ذهنم برسد نحوه عملکرد سلولهای زیست محیطی بود. یعنی این سبک برنامه نویسی از روی سلولهای جاندارها الگو برداری شده است.

آن چیزی که باعث شد که آلن کی از روی سلول های جانداران الگو برداری کند نحوه زندگی سلولها بود:

هر سلول نمونه ای از اصل است و هر خصوصیاتی که دارد از اصل خود به ارث برده.(ژنتیک سلول). همچنین هر سلول رفتارهایی دارد که از اصل خود به ارث برده.

سلولها همگی مستقل از هم زندگی می کنند و براساس ارسال پیام های شیمیایی با یکدیگر ارتباط برقرار می کنند. ارسال پیام به این صورت است که پیام از پوسته یکی خارج و به پوسته دیگری وارد می شود.

سلولها می توانند از یکدیگر متمایز شوند.

با توجه به گفته های بالا :

می توان متوجه شد که همان مشخصه کلاسها رو بیان می کند یعنی هر شی از یک کلاسی تشکیل شده که ویژگی های آن کلاس رو با خودش به ارث برده است.

همانطور که می دانیم اشیا با یکدیگر ارتباط برقرار می کنند. نحوه ارتباط یا فرستادن پیام در اشیا هنگام فراخوانی رفتارها در یک رویداد است.

هر شی خودش یک شناسنامه یا Identifier دارد که ویژگی های آن شی را بیان می کند.

Small Talk مانند سلولهای جانداری عمل می کند. یعنی آلن کی در تمامی قسمتهای این زبان تعیین کرده بود که اشیا با هم ارتباط برقرار می کنند و دارای شناسنامه ای هستند و همچنین مستقل از همدیگر کار می کنند. اصول اولیه ای که آلن کی برای برنامه نویسی شی گرا تعیین کرده بود :

هر چیزی یک شی است.

هر برنامه ای شامل اشیا هست که اشیا با ارسال پیام به یکدیگر تعیین می کنند که چه کاری باید الان انجام بشود.

هر شی یک حافظه Memory برای خودش دارد که بتوان به وسیله آن اشیای دیگر را ساخت.

هر شی خودش از یک کلاس Class هست.

برنامه نویسی شی گرا

برنامه نویسی شی گرا شیوه نوینی است که در آن می توان قطعاتی را ایجاد کرد و در برنامه های مختلف مورد استفاده قرار داد.قابلیت خوانایی برنامه هایی که در این روش نوشته می شوند بالا بوده ، تست ، عیب یابی و اصلاح آن ها آسان است . شی گرایی ، بر اشیا تاکید دارد. در برنامه نویسی شی گرا اشیا به صورت انتزاع مطرح می شوند. انتزاع: به آن چیزی می گویند که شما در مورد آن فکر می کنید و در یک دید کلی مطرح می کنید. مثلاً وقتی به یک دانه شن فکر می کنید

ناخوداگاه فکرتان به سمت ساحل می رود یا وقتی به یک درخت فکر می کنید ذهنتان به سمت جنگل متمرکز می شود. به این انتزاع می گویند که در این سبک برنامه نویسی مطرح می شود که اشیا با توجه به کلاس های خودشان ساخته می شوند که خود کلاس ها ممکن است که از کلاسهای دیگری مشتق شده باشند.

فرم نوشتن کلاس ها بدین صورت خواهد بود:

```
Class نام کلاس {
    Public:

    Private:

    Protected:
}
```

هر کلاس دارای یکسری خصوصیات و رفتار هاست. خصوصیات همان متغیرها بوده و رفتارهای یک کلاس توابع عضو آن می باشند. در اینجا سه عنوان مختلف هم دیده می شود که در زیر مجموعه هر قسمت می توان متغیر یا توابعی تعریف کرد که البته از لحاظ میزان دسترسی در نوسان خواهد بود.

:Public

هر متغیر و یا تابعی در این قسمت بدین معنا خواهد بود که آنها در خارج از کلاس هم قابل دسترسی بوده و می توان آنها را به کار گرفته یا تغییر داد. معمولا متغیر های عادی در این گروه قرار می گیرند.

:Private

متغیرها و توابع عضو این قسمت بصورت خصوصی تلقی شده و فقط در همین کلاس قابل دسترسی خواهند بود. از بارزترین مثال در این گروه می توان به شماره پسورد یک کاربر یا دیگر اطلاعات خصوصی یک شی اشاره کرد.

:Protected

این گروه برای معرفی مقادیر و یا توابعی بوده که بین آن کلاس و دیگر کلاس های مشتق شده از همان کلاس در دسترس خواهد بود. کلاس های مشتق کلاس هایی هستند که از دیگر کلاس ها به ارث می برند و در قسمت های بعدی ارث بری را توضیح خواهیم داد.

شی گرایی بر چند پایه استوار است که به قرار زیرند:

Inheritance

Encapsulation

Polymorphism

Abstraction

Interface

اکنون به توضیح مختصر هر یک می پردازیم:

Inheritance ارث بری

پدر و فرزندی را در نظر بگیرید . هر پدری مشخصات فردی به خصوصی دارد . فرزند وی می تواند همه خصوصیات او را به ارث برد و خصوصیتهای دیگری نیز داشته باشد که پدرش ندارد . این یعنی ارث بری ! در برنامه نویسی شی گرا از مفهوم ارث بری استفاده های زیادی می شود . قابلیت استفاده دوباره از کد (Reusability) یکی از مزیات اصلی ارث بری است. ارث بری بدین صورت پیاده سازی می شود که فرض شود کلاس B از کلاس A به ارث برده و تولید می شود. این بدین معنا خواهد بود که کلاس B همه خصوصیات و رفتار های کلاس A را به ارث خواهد برد و همچنین می تواند گزینه های دیگری را هم در خود تعریف و به آنها اضافه کند:

Class B(کلاس مشتق شده): A(کلاس والد);

Encapsulation کپسوله سازی

همانطور که از اسمش پیداست ، به قرار دادن پیاده سازی در یک کپسول اشاره می کند ، به طوری که کاربر بیرونی از نحوه پیاده سازی مطلع نباشد و فقط بداند که این کپسول کار خاصی را انجام می دهد. هدف Encapsulation این است که ما را از پرداختن به ریز موضوعات رها کند و اشیا را به صورت یک جعبه سیاهی بدانیم که به ازای یک ورودی خاص خروجی خاصی می دهند . در C# برای کپسوله کردن از Access Modifier های Private، Protected ، Public استفاده شود.

Polymorphism چند ریختی

فرض کنید پدر شما کار خاصی را به طریق خاصی انجام می دهد . مثلا برای پختن غذا اول ظرفهای دیشب را شسته و بعد گاز را روشن می کند و بعد غذا را می پزد! شما که خصوصیات پدر و کارهای او را به ارث می برید برای مثال برای پختن غذا ابتدا گاز را روشن می کنید بعد کبریت می کشید،غذا را می پزید و بعد ظرفهای دیشب را می شویید! پختن غذا کاری است که شما از پدر خود به ارث می برید ، ولی آن را به طریق دیگری انجام می دهید . یعنی یک کار توسط فرزندان مختلف یک پدر به طرق مختلفی انجام می شود . این دقیقا همان چیزی است که به آن چند شکلی یاPolymorphism می گویند.

Abstraction تجرید

تجرید یا مجرد سازی ! به کلاسی مجرد گفته می شود که پیاده سازی متدها در آن انجام نمی شود.فرض می کنیم که شما رئیس یک شرکت بزرگ برنامه نویسی هستید و می خواهید پروژه بزرگی را انجام دهید. برای اجرای پروژه از برنامه نویسان مختلفی استفاده می کنید که ممکن است همه آنها هموطن نباشند! مثلا هندی ، ایرانی یا آلمانی باشند! اگر قرار باشد هر برنامه نویسی در نامگذاری متدها و کلاسهایش آزاد باشد ، در کد نویسی هرج ومرج به وجود می آید . شما به عنوان مدیر پروژه ، کلاسی تعریف می کنید که در آن تمام متدها با ورودی و خروجی هایشان مشخص با شند . ولی این متدها را پیاده سازی نمی کنید و کار پیاده سازی را به برنامه نویسان می دهید و از آنها می خواهید که همه کلاسهایی

که می نویسند از این کلاس شما به ارث ببرند و متدها را به طور دلخواه پیاده سازی کنند . این باعث می شود که با داشتن یک کلاس ، ورودی و خروجی های مورد نظر خود را داشته باشید و دیگر نگران برنامه نویسان نباشید. کلاسی که شما تعریف می کنید یک کلاس مجرد نامیده می شود. برای تعریف یک کلاس مجرد از کلمه کلیدی abstract استفاده می کنیم. فیلدهایی که می خواهیم در کلاس های مشتق شده از این کلاس پیاده سازی شوند حتما باید با abstract تعریف شوند . یک کلاس مجرد می تواند فیلدها و متدهای نا مجرد داشته باشد.اگر متد نا مجردی در یک کلاس مجرد تعریف کردید، حتما باید آن را پیاده سازی کنید و نمی توانید پیاده سازی آن را به کلاسهای مشتق شده بسپارید.

Interface

اینترفیس در برنامه نویسی همانند همان کلاس است تنها با این تفاوت که هیچکدام از اعضای آن پیاده سازی نمی شوند . در واقع یک اینترفیس گروهی از متدها ، خصوصیات ، رویدادها و Indexer ها هستند که در کنار هم جمع شده اند. تنها چیزی که اینترفیس دارا می باشد امضای (signature) تمامی اعضای آن می باشد. به ای معنی که ورودی و خروجی متدها ، نوع property ها و ... در آن تعریف می شوند ولی چیزی پیاده سازی نمی شود. اینترفیس ها سازنده و فیلد ندارد. یک اینترفیس نمی تواند Operator Overload داشته باشد و دلیل آن این است که در صورت و جود ویژگی ، احتمال بروز مشکلاتی از قبیل ناسازگازی با دیگر زبانهای .Net مانند VB.Net که از این قابلیت پشتیبانی نمی کند وجود داشت. نحوه تعریف اینترفیس بسیار شبیه تعریف کلاس است تنها با این تفاوت که در اینترفیس پیاده سازی وجود ندارد.

یک مثال کامل:

این مثال در واقع یک نمونه نسبتا کامل از پیاده سازی به صورت شی گرایی می باشد. در اینجا یک کلاس خودرو بصورت خلاصه و ساده پیاده سازی شده است. در این کلاس طبق تعریف از دو نمونه public و private استفاده شده است

که هر کدام متغیر ها یا متد هایی را در خود گنجانده اند (خط ۸ و خط ۱۲). بدنه برنامه اصلی از خط ۶۴ شروع می شود و یک برای نشان می دهد که کاربر می تواند با زدن کلید های A و B و یا C سه عمل مختلف توسط کلاس خودرو انجام شود. مقدار A بعنوان شتاب و حرکت در نظر گرفته می شود و مادامیکه زده می شود خودرو تابع شتاب را در خط ۴۴ را صدا می زند بدین صورت که متغیر speed به عنوان مقدار سرعت ۵ واحد به آن اضافه می شود. کلید B متد ترمز را صدا می زند که همان متغیر مقدار سرعت را با هر بار زدن ۵ واحد کم می کند و این خواندن مقدار از کاربر به همین شکل ادامه خواهد داشت تا اینکه کاربر C را زده و خارج شود. در واقع یک حلقه داریم که شرط خروج آن کلید C می باشد. همچنین می بینیم که توابع ابتدا تعریف شده و سپس پایین تر از آن پیاده سازی می شوند مانند خط ۱۷ که تابع سرعت تعریف شده و در خط ۴۴ پیاده سازی شده است.

```
1. #include <iostream>
2. #include <cstring>
3. #include <cctype>
4. using namespace std;
5.
6. class Car
7. {
8. private:
9.     int YearModel;
10.    int Speed;
11.    string Make;
12. public:
13.    Car(int, string, int);
14.    string getMake();
15.    int getModel();
16.    int getSpeed();
17.    void Accelerate();
18.    void Brake();
19.    void displayMenu();
20. };
21.
22. Car::Car(int YearofModel, string Makeby, int Spd)
23. {
24. YearModel = YearofModel;
25. Make = Makeby;
26. Speed = Spd;
27. }
28. //To get who makes the car.
29. string Car::getMake()
30. {
31.    return Make;
32. }
33. //To get the year of the car.
34. int Car::getModel()
35. {
36.    return YearModel;
37. }
38. //To holds the car actual speed.
39. int Car::getSpeed()
40. {
```

```
41.     return Speed;
42. }
43. //To increase speed by 5.
44. void Car::Accelerate()
45. {
46.     Speed = Speed + 5;
47. }
48. //To drop the speed of the car by 5.
49. void Car::Brake()
50. {
51.     Speed = Speed - 5;
52. }
53.
54. void displayMenu()
55. {
56.     cout <<"\n              Menu\n":
57.     cout << "----------------------------\n";
58.     cout << "A)Accelerate the Car\n";
59.     cout << "B)Push the Brake on the Car\n";
60.     cout << "C)Exit the program\n\n";
61.     cout << "Enter your choice: ";
62. }
63.
64. int main()
65. {
66.     int Speed = 0; //Start Cars speed at zero.
67.     char choice; //Menu selection
68.
69.     cout << "The speed of the SUV is set to: " << Speed <<endl;
70.     Car first( 2007, "GMC", Speed);
71.
72.     //Display the menu and get a valid selection
73.     do
74.     {
75.         displayMenu();
76.         cin >> choice;
77.         while(toupper(choice) < 'A' || toupper(choice) > 'C')
78.         {
79.             cout << "Please make a choice of A or B or C:";
80.             cin >> choice;
```

```
81.         }
82.
83.         //Process the user's menu selection
84.         {
85.             switch (choice)
86.             {
87.             case 'a':
88.             case 'A': cout << "You are accelerating the car. ";
89.                 cout << Accelerate(first) << endl;
90.                 break;
91.             case 'b':
92.             case 'B': cout << "You have choosen to push the brake.";
93.                 cout << Brake(first) << endl;
94.                 break;
95.             }
96.         }while (toupper(choice) != 'C');
97.
98.     return 0;
99. }
```

در خط ۹۶ گزینه toupper اگر کاراکتری کوچک باشد آن را به کاراکتر بزرگ همان تبدیل می کند. بعنوان مثال c را به C تبدیل می کند.

تمرین 4

تمرین ۱: کلاسی کامل برای یک دانشجو کلیه عملیاتی که می تواند انجام دهد را بنویسید. از جمله عملیات دانشجو می توان به انتخاب واحد، حذف یک درس، معرفی به استاد و ... می باشند.

برنامه نویسی در C#

در سی شارپ کلاسهایی وجود دارد که با استفاده از آنها می توانیم به پایگاه دادهای مختلف وصل شد از جمله Access , SQL و غیره . وظیفه این کلاسها این است که دستورات ما را به پایگاه داده مورد نظر انتقال داده و عملیات روی آنها را پیاده سازی نماید. مثلا اگر خواستیم دستور INSERT را بر روی پایگاه داده مورد نظر پیاده سازی کنیم از این کلاس ها

استفاده می کنیم .همانطور که گفته شد کلاس هایی وجود دارد برای کار با پایگاه داده های مختلف که این کلاس ها در هدر های مخصوص به خود وجود دارند که برای استفاده از این کلاس ها باید این هدر ها را به برنامه اضافه کرد به عنوان مثال برای وصل شدن به پایگاه داده **SQL** و استفاده از آن باید هدر **System.Data.SqlClient** را به برنامه اضافه کرد مطابق شکل زیر

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Data.SqlClient;
using System.Data;
using System.Windows.Forms;
```

لازم به ذکر است که ما از هدر یا سرفصل **System.Data.SqlClient** برای آموزش استفاده می کنیم و مابقی پایگاه دادها تفاوت چندانی ندارند

کلاس SqlConnection

برای کار با هر پایگاه داده قبل از هر کاری ابتدا باید به آن پایگاه داده وصل شویم .برای وصل شدن در سرفصل **System.Data.SqlClient** از کلاس **SqlConnection** استفاده می کنیم. این کلاس دارای یک خصوصیت است به نام **ConnectionString** که نوع آن **String** است. برای وصل شدن به پایگاه داده مورد نظر باید ابتدا **ConnectionString** را مقدار دهی کنیم.مقدار **ConnectionString** شامل نام سرور ،نام پایگاه داده و تنظیمات مربوط به نحوه دسترسی در زیر یک نمونه از آن آورده شده است :

"DataSource=MAA-PC\MAA_SERVER;InitialCatalog=testDB;User ID=sa;Password=maa"

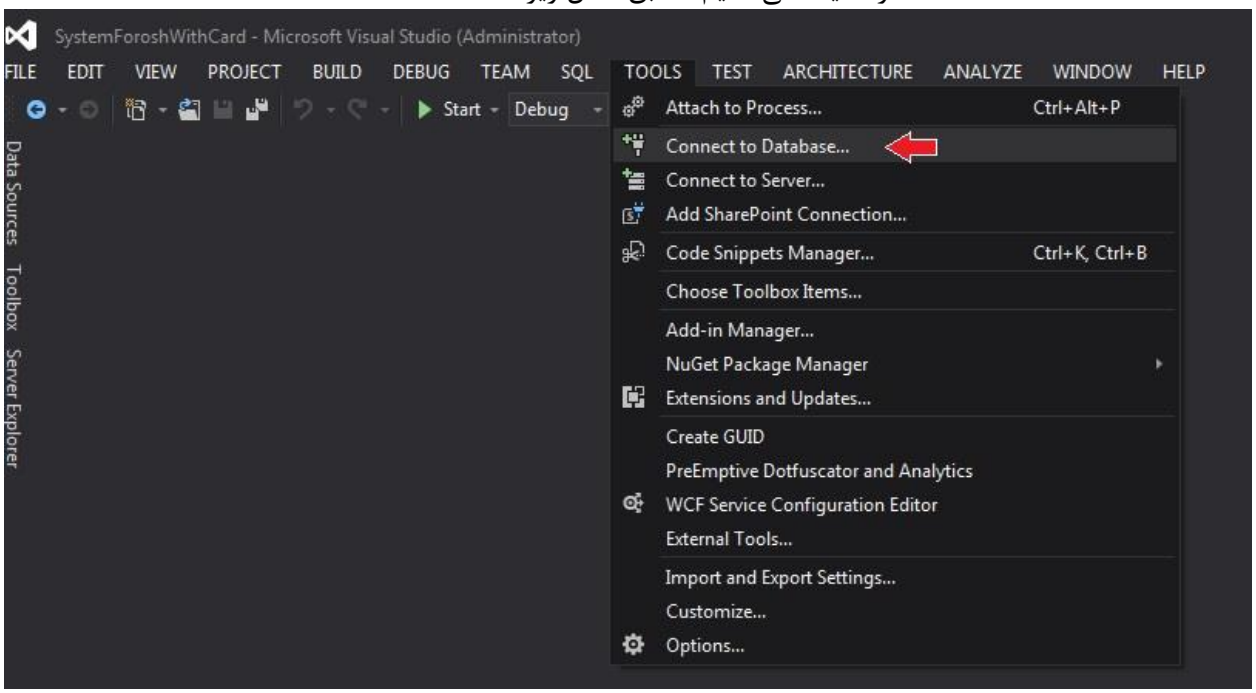
Data Source= نام سرور

Initial Catalog = نام پایگاه داده

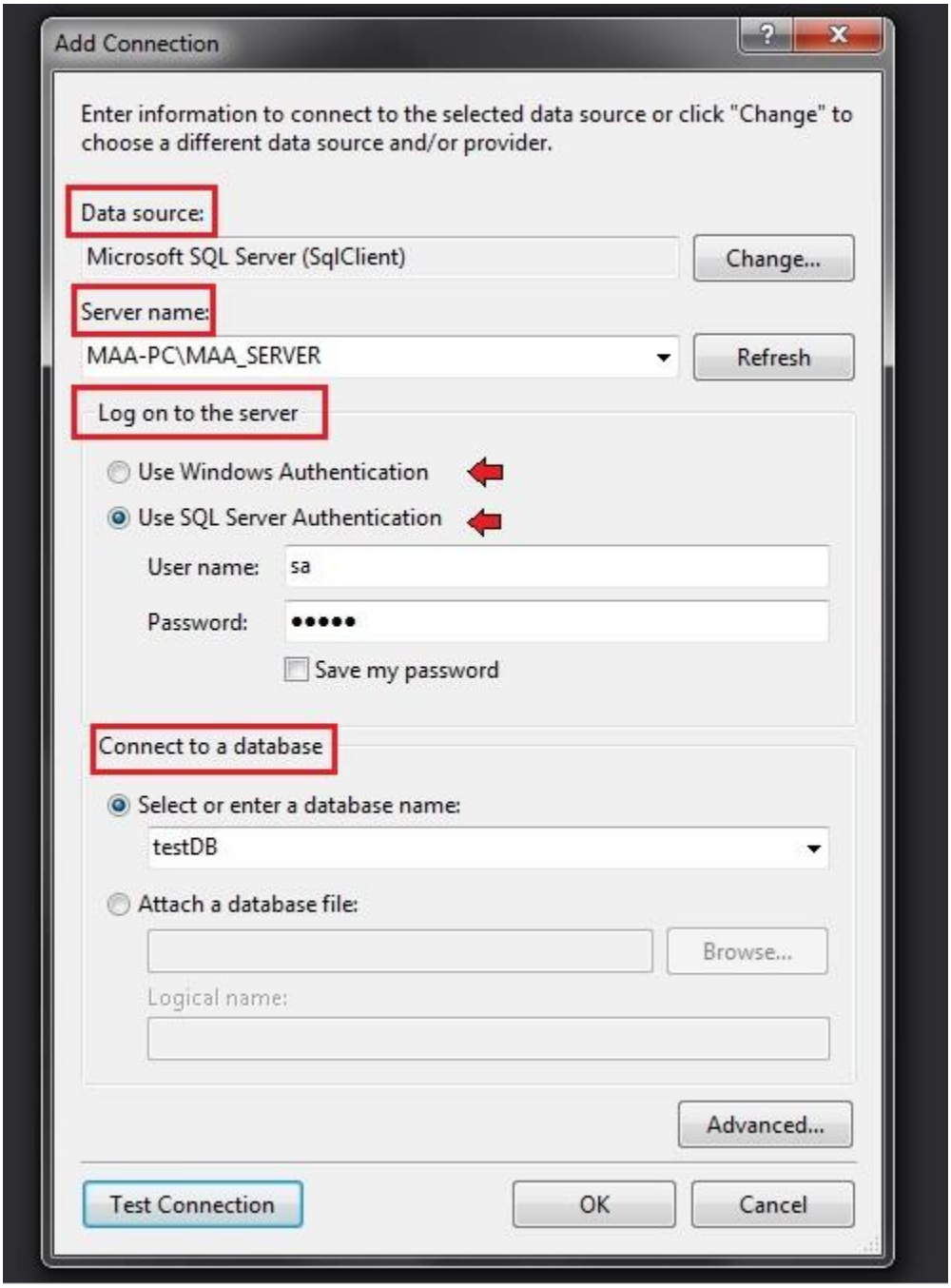
در این رشته همانطور که مشخص است نام سرور ما **MAA-PC\MAA_SERVER** و نام پایگاه داده ما **testDB** است و نحوه دسترسی را با **username** و **password** می باشد که البته می شود بدون **username** و **password** هم به پایگاه داده متصل شد که در این صورت مقدار **ConnectionString** به صورت زیر می باشد :

DataSource=MAA-PC\MAA_SERVER;InitialCatalog=testDB;Integrated Security=True

شاید این سوال پیش آید که مقدار **ConnectionString** را از کجا باید به دست آورد.باید بگویم که روشهای زیادی برای بدست آوردن این رشته وجود دارد در زیر به یکی از این روشها اشاره می کنیم .فرض کنید که **SQLSERVER** در سیستم با **Intance** به نام **MAA_SERVER** نصب شده است . در محیط **Visual Stadio** از منوی **Tools** قسمت **Connect to DataBase** زا کلیک می کنیم مطابق شکل زیر

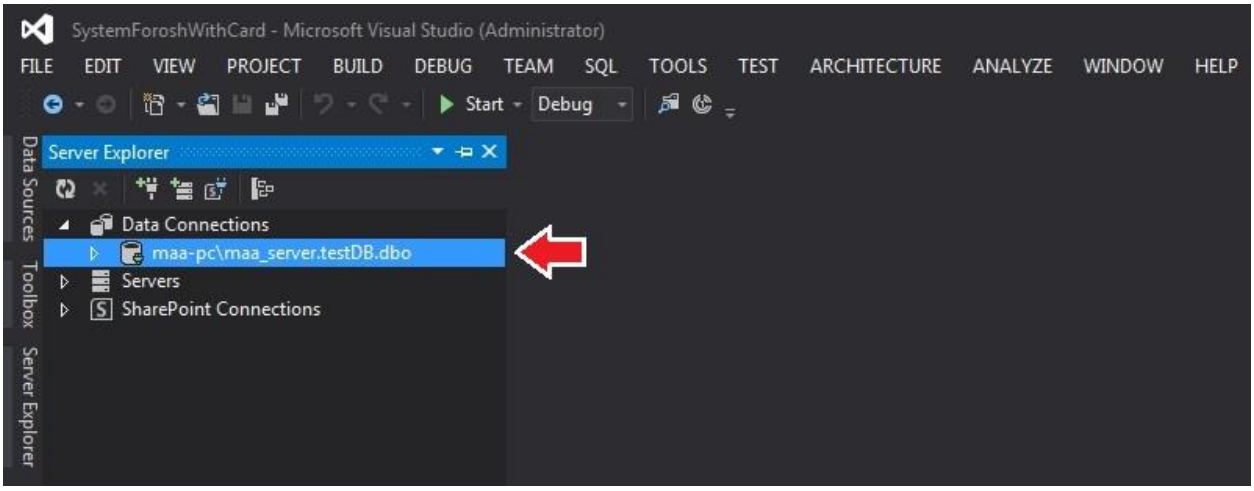


در پنجره **add Connection** مطابق شکل زیر مقادیر را مقدار دهی می کنیم

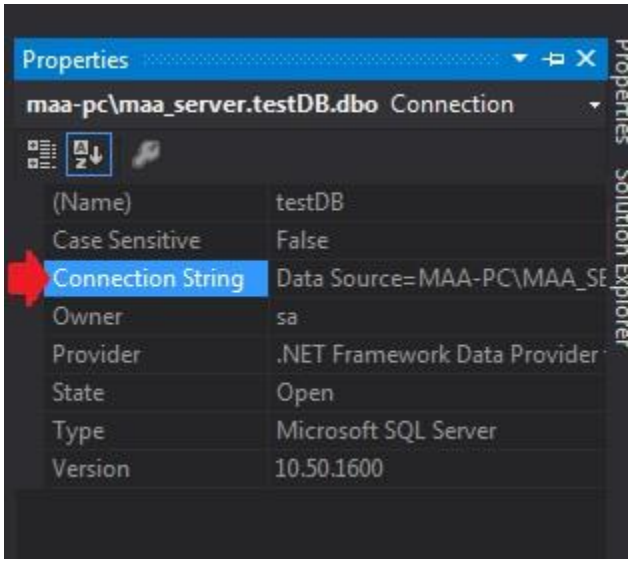


در قسمت **Data Source** نوع پایگاه داده را مشخص می کنیم بدیهی است چون که ما می خواهیم به یک بانک اطلاعاتی از نوع **SQL** وصل شویم مقدار **Microsoft SQL Server** را انتخاب کردیم در صورت نیاز اگر خواستید

نوع دیگری را انتخاب کنید با استفاده از **Change** می توانید انجام دهید.در قسمت **Server Name** نام سرور را انتخاب کرده در قسمت **log on to the server** نحوه **login** شدن به پایگاه داده را مشخص می کنیم که اگر قسمت **Use Windows Authentication**را انتخاب کنید نیاز به **User name** و **Password** نیست و در آخر در قسمت **Connect to database** نام **database** مورد نظر را انتخاب می کنید . سپس با استفاده از **Test Connection** تست می کنید که اطلاعات وارد شده درست هستند یا که خیر در صورتی که پیغام زیر را مشاهده کردید برروی **OK** کلیک کرده تا فرم **add Connection** بسته شود.هنگامی که فرم **add Connection** باز می شود به صورت اتوماتیک پنجره **Server Explorer** باز می شود اگر که مشاهده نکردید می توانید با استفاده از منوی **view->Server Explore** این پنجره را مشاهده کنید. در این پنجره قسمت **Data Connection** را باز کنید نام سرور و دیتابیس مورد نظر خود را مشاهده می کنید.



بر روی نام دیتابیس خود کلیک راست کرده سپس برروی **Properties** کلیک کنید. در سمت راست پنجره **Properties** باز می شود در این پنجره می توانید **ConnectionString** را مشاهده کنید.



حال ببینیم از این کلاس **Sql Connection** چگونه استفاده می شود . برای استفاده از این کلاس کافیست یک شی از آن مطابق زیر ایجاد شود و **connectionstring** آن برابر با رشته ای قرار بدهید که روش بدست آوردن آن را در قسمت قبل توضیح دادم.

```
string Connectionstr = "Data Source=MAA-PC\\MAA_SERVER;Initial Catalog= testDB;User ID=sa;Password=maa";
SqlConnection connection = new SqlConnection();
connection.ConnectionString = Connectionstr;
```

این کلاس همچنین دارای دو متد که یکی برای باز کردن دیتابیس و دیگری برای بستن دیتابیس می باشد. هر زمان که خواستیم دستوری به دیتابیس بفرستیم ابتدا دیتابیس را باز می کنیم و پس از ارسال دستور و اجرای آن دیتابیس را می بندیم تا برای استفاده های دیگر آن را آزاد کنیم.

```
connection.Open();
// code ....
connection.Close();
```

کلاس SqlCommand

با توجه به مطالب قبل با نحوه اتصال با پایگاه داده آشنا شدید در این بخش به موضوع ارسال دستورات و اجرای آنها می پردازیم. برای ارسال واجرای دستورات باید از کلاس **sql Command** استفاده کنید. نحوه استفاده از این کلاس به این صورت است که یک شی از آن ایجاد می کنیم در هنگام ایجاد شی جدید از این کلاس دو آرگومان نیاز دارید که با آن ارسال کنید یکی **Connecti on** و دیگری دستوری که می خواهید در دیتابیس اجرا شود مطابق شکل زیر که در آن نحوه نوشتن دستور **i nsert** را توسط این کلاس نشان می دهد

```
string Connectionstr = "Data Source=MAA-PC\\MAA_SERVER;Initial Catalog= testDB;User ID=sa;Password=maa";
SqlConnection connection = new SqlConnection();
connection.ConnectionString = Connectionstr;
string commnd = "insert into tabletest values('ali','25')";
SqlCommand com = new SqlCommand(commnd, connection);
connection.Open();
com.ExecuteNonQuery();
connection.Close();
```

نکته ای که باید با آن اشاره کرد این است که باید حتما از متد **ExecuteNonQuery** استفاده کنیم تا تغییرات ما ذخیره شود.

کلاس sqldataadapter

از این کلاس بیشتر برای پر کردن جدول **datatable** استفاده می شود در واقع با این کلاس بیشتر اطلاعات را از دیتابیس استخراج کرده در فرمهای برنامه نمایش می دهیم. برای استفاده از این کلاس هم نیز کافیست یک شی از آن ایجاد کنید و متد **Fill** آن را فراخوانی کنید مانند کد زیر

```
string Connectionstr = "Data Source=MAA-PC\\MAA_SERVER;Initial Catalog= testDB;User ID=sa;Password=maa";
SqlConnection connection = new SqlConnection();
connection.ConnectionString = Connectionstr;
string commnd = "insert into tabletest values('ali','25')";
SqlCommand com = new SqlCommand(commnd, connection);
connection.Open();
com.ExecuteNonQuery();
connection.Close();
```

شاید سوال پیش بیاید که **DataTable** چیست و چه کاربردی دارد. برای اینکه بتوانیم یک کپی از جدول دیتابیس در سی شارپ داشته باشیم از این کلاس استفاده می کنیم این کلاس به صورت آرایه دوبعدی عمل می کند و ما می توانیم به راحتی و با سرعت زیاد به عناصر آن دستیابی داشته باشیم. از کاربردهای دیگر این کلاس این است که ما می توانیم جدول **dataGridView** را نیز با آن پر کنیم برای این کار کافی است یک **datagridview** را در فرم قرار دهیم و خاصیت **datasource** آن را مقدار دهیم

Datagridview1.datasource=dt;

هرچند کلاس های بسیار زیادی هستند که با دیتابیس عملیات های مختلفی را انجام می دهند اما این ها کلاس هایی بودند که عملیات لازم برای انجام یک ارتباط ساده با پایگاه داده را انجام می دهند.

مباحث روز دنیای برنامه نویسی

قابل توجه همه دانشجویان گرامی که این مبحث جز امتحان پایان ترم نمی باشد. هرچند مطالعه آن اختیاری می باشد ولی پیشنهاد می شود که به منظور کسب دید بهتر نسبت به دنیای برنامه نویسی مروری بر آن داشته باشید دنیای امروز ، دنیای کسب و کار و به اصطلاح بیزینس است . به ویژه کسانی که به طور حرفه ای در حوزه کامپیوتر مشغول به فعالیت میباشند ، در صورت هوشمندی و مجهز بودن به دانش روز ، میتوانند از این سفره گسترده ، سهمی برای خود بردارند . در همین زمینه برنامه نویسی کامپیوتر از جمله مهارت ها و مشاغلی است که تخصص در آن میتواند آینده شغلی شما را تضمین کند . البته در حال حاضر محیط های برنامه نویسی متعددی در عرصه کامپیوتر وجود دارد . ما در اینجا به معرفی ۱۰ زبان برنامه نویسی برتر پرداخته ایم که تسلط به یکی از آنها در حد یک متخصص و خبره ، میتوند باعث موفقیت شما در یافتن شغلی در حوزه I T شود . اما خب اگر این امکان وجود داشته باشد ، یعنی اگر همت کافی و همچنین علاقه و شورو شوق آن را داشته باشید که حداقل به دو زبان برنامه نویسی مسلط باشید ، آنگاه به قول معروف ، نور علی نور است . ضمن اینکه قطعا برنامه نویسان حرفه ای به خوبی میدانند که اصول و اسکلت اصلی بسیاری از زبان ها و کامپایلر های برنامه نویسی ، بسیار به هم شبیه بوده و اگر شما به یک زبان تسلط پیدا کنید ، یادگیری فرامین و دستورالعمل ها و جزئیات یک زبان دیگر ، کار کمابیش ساده ای است .

البته ذکر این نکته لازم است که ما در اینجا به شما فقط سرنخ میدهیم و اگر شما پس از خواندن این مطلب واقعا تصمیم گرفتید یک برنامه نویس شوید و یا مهارت های ذوقی و آماتوری خود را تکمیل کنید ، بهتر است اطلاعات تکمیلی و بیشتری را از منابع مختلف به دست آورید . اطلاعات و دانسته هایی که به ترغیب شما جهت انتخاب زبا اصلی شما به عنوان یک خبره در آن زبان کمک میکند و همانطور که اشاره شد ، ممکن است شما را تشویق کند که به عنوان یک زبان فرعی و دوم ، زبان ثانویه دیگری را هم آموزش ببینید .

همچنین فراموش نشود که اکنون ده ها و ده ها زبان برنامه نویسی در جهان کامپیوتر وجود دارد که هر یک متولیان و طرفداران خود را دارد . اما مهم این است که ۱۰ زبان برتر و اول دنیای برنامه نویسی کدام یک میباشند

.

بنابراین اگر تصمیم دارید در دنیای برنامه نویسی کامپیوتر ، برای خود جایی باز کنید و یا بالاتر از آن ، به طور جدی به این فکر هستید که از این راه امرار معاش کنید ، با ما همراه باشید :

JAVA

برای سرمایه گذاری در پروژه های اقتصادی ، بهترین زبان ، زبان برنامه نویسی جاوا و .Net مایکروسافت است که از بین این دو ، باز هم جاوا حرف اول را می زند و طبق آخرین اطلاعات ، بیش از ۹۹ میلیون نفر در سراسر دنیا به این زبان برنامه نویسی می کنند که این خود نشان دهنده فضای مناسب برای این زبان و تعداد زیادی از برنامه های کاربردی و ... میباشد که به این زبان نوشته شده اند و همگی آنها نیاز به پشتیبانی ، نگهداری و بروز رسانی دارند.

علاوه بر همه اینها ، زبان برنامه نویسی سیستم عامل Android (برای موبایل) ، جاوا میباشد . سیستم عامل Android که خود گستردگی خوبی دارد ، برای تکامل و توسعه ، روز به روز نیاز به برنامه ها و ابزار های جدیدی دارد که همگی باید به زبان جاوا باشند . طبق آخرین آمار انجمن برنامه نویسان TIOBE ، زبان برنامه نویسی جاوا به لحاظ کارایی و استفاده ، دارای بالاترین رنکینگ در سرتا سر دنیاست ، یعنی رتبه اول را از آن خود نموده است .

C#

C# یک زبان برنامه نویسی جند مدلی است که شامل دستوری ، تابعی ، عمومی ، شیء گرا و جزء گرا است . این زبان توسط شرکت مایکروسافت و از دل زبان .NET مایکروسافت خلق و پرورش یافت و بعد ها توانست استاندارد های ISO و Ecma را دریافت کند و بعنوان یک زبان استاندارد تایید شود .

در عین حال زبان C# ، زبان برگزیده مایکروسافت برای ایجاد سیستم عامل Windows Phone ۷ (سیستم عامل موبایل هایی همچون HTC) میباشد . این زبان نیز مانند زبان جاوا برای پروژه های اقتصادی و بزرگ ، انتخاب بسیار خوبی است . گر چه تعداد برنامه نویسانی که به این زبان برنامه نویسی میکنند ، به پای برنامه نویسان جاوا نمی رسد ، ولی به لطف پشتیبانی مایکروسافت ، این زبان یکی از مدعیان قوی در بین زبانهای برنامه نویسی است .

C/C++

میدانیم که این دو زبان با هم فرق دارند و در واقع زبان C++ بر اساس زبان C ساخته شده و به نوعی به آن اضافه شده است ، ولی به هر جهت با هم در نظر گرفته میشوند . اما آنچه که در بیشتر موارد دیده میشود ، این است که زبان C++ بیشتر در نرم افزار های مهندسی و صنعتی مورد استفاده قرار گرفته است و این خود بر خاص بودن این زبان را میرساند.

چند نمونه از موارد ، عبارتند از نرم افزار های مربوط به برنامه های کاربردی ، درایور های دستگاه های صنعتی مختلف ، نرم افزار های مربوط به سیستم های الحاقی با اصطلاحا Embedded Systems ، همچنین برنامه های مربوط به Server - Client ها با ضریب اجرایی فوق العاده بالا و همچنین نرم افزار های سرگرمی مانند بازی های ویدیویی.

نا گفته نماند C++ در مقایسه با تمام زبان های برنامه نویسی ، حتی جاوا ، از قدرت تاثیر گذاری فوق العاده زیادی برخوردار است . به هر حال زبانهای C و C++ در فهرست بندی TIOBE ، به ترتیب رده های دوم و سوم قرار دارند .

PHP

php یک زبان عمومی است که برای برنامه های کابردی و بخصوص برای طراحی صفحات وب بسیار محبوب است . در واقع اگر میخواهید مستقل و آزاد برنامه نویسی کنید ، PHP یکی از زبان های ایده آل و خوب است . این زبان ، یک زبان اسکریپتی است که اساس و بنیان ایجاد آن ، برای طراحی صفحات پویای وب بوده است. در ایران نیز روند رشد بازار آن بسیار بالا بوده و در آینده بازار بسیار پر رونق تری خواهد داشت.

PYTHON

زبان های برنامه نویسی پویا ، بخصوص Pyt hon ، برای ایجاد برنامه های کاربردیپ تحت وب و برنامه های کاربردی ابری در قالب هایی مثل Dj ango استفاده میشوند . قابل ذکر است موتور برنامه های گوگل ، با زبان Pyt hon نوشته شده است و فقط با این زبان پشتیبانی و بروز میشود .

تمرین تشویقی

این تمرین به منظور تشویق دانشجویان برای فعالیت بیشتر داده شده که اختیاری می باشد ولی انجام آن تا حد اکثر یک نمره را برای دانشجو بصورت تشویقی ثبت خواهد کرد. لازم به ذکر است که دانشجو باید بتواند بصورت عملی هم از آن دفاع کند. مدت زمان لازم برای حل آن ۴۸ ساعت می باشد. به جواب های ناکامل در صورت انجام در مسیر درست مساله، هم امتیاز مثبتی در نظر گرفته خواهد شد.

مثلث خیام پاسکال را به نحوی چاپ کنید که هر سطر **فرد** از اعداد در این مثلث بصورت یکی در میان عدد ۱ باشد. مانند شکل زیر: