

# اصول برنامه سازی به زبان C++

## فصل اول – مبانی

تهیه و تنظیم : محمد نعیمی

عضو هیات علمی دانشگاه آزاد اسلامی

# مفاهیم پایه ای در برنامه نویسی

یک «برنامه» در حقیقت دستورالعمل‌های متوالی است که می‌تواند توسط یک کامپیوتر اجرا شود.

معمولاً از **زبانهای سطح بالا (high level language)** برای برنامه نویسی استفاده می‌شود.

**زبان سطح بالا:** به زبانهایی گفته می‌شود که دستورات آن برای انسان قابل فهم است یا به بیان دقیق تر شامل کلمات و عبارات معمول در زبانهایی مثل انگلیسی و ... می‌باشد.

**زبان ماشین:** کامپیوتر درکی از برنامه نوشته شده در زبان سطح بالا ندارد و تنها میتواند برنامه هایی که در فرم باینری که تنها حاوی ارقام 0 و 1 هستند (**زبان ماشین**) را درک نماید.

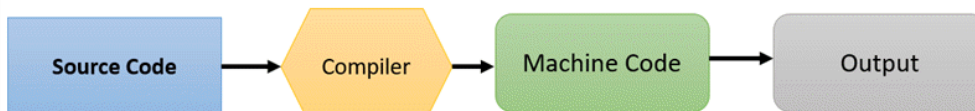
□ برنامه نوشته شده در زبان سطح بالا (**Source Code**) باید طی فرایندی به زبان ماشین تبدیل شود.

□ برای این منظور دو روش وجود دارد که بر اساس آن زبان‌های برنامه نویسی به دو نوع اصلی زبان‌های مفسری (**interpreter**) و زبان‌های کامپایلری (**compiler**) تقسیم می‌شود.

## کامپایلر (compiler) :

یک برنامه کامپیوتری است که **کل برنامه** نوشته شده به زبان سطح بالا را به **کد ماشین** تبدیل میکند. جهت کامپایل، برنامه باید با قوانین نحوی (syntax) زبان برنامه نویسی مربوطه مطابقت داشته باشد. کامپایلر تنها یک برنامه است و نمی‌تواند خطاهای موجود در برنامه شما را برطرف کند. بنابراین، اگر در برنامه خود خطای نحوی داشته باشید باید آن را اصلاح کنید در غیر این صورت کامپایل نمی‌شود.

How Compiler Works



## مفسر (interpreter) :

یک برنامه کامپیوتری است که **هر دستور برنامه** نوشته شده به زبان سطح بالا را به **کد ماشین** تبدیل می‌کند. هم کامپایلر و هم مفسر کار یکسانی را انجام می‌دهند که عبارت است از تبدیل زبان برنامه نویسی سطح بالا به کد ماشین اما، یک کامپایلر **قبل از اجرای برنامه**، برنامه را به کد ماشین تبدیل می‌کند (یک فایل اجرایی ایجاد می‌کند) اما مفسر، هنگامی که برنامه اجرا می‌شود، دستورات برنامه را به کد ماشین تبدیل می‌کند.

How Interpreter Works

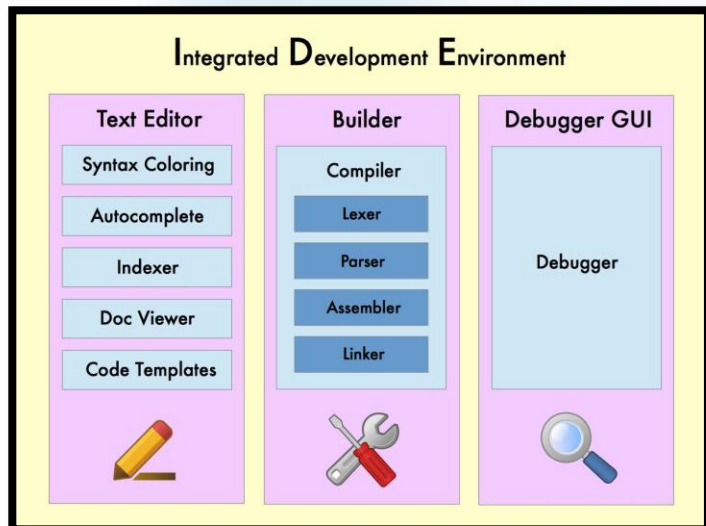


# تفاوت کامپایلر و مفسر

مفسر	کامپایلر
مفسر در هر لحظه یک دستور از برنامه سطح بالا را به کد ماشین تبدیل و اجرا می کند	کامپایلر کل برنامه را اسکن میکند و کل آن را به یکباره به کد ماشین تبدیل نموده و بعد کد ماشین اجرا می شود
مفسر زمان بسیار کمتری برای تجزیه و تحلیل کد برنامه صرف می کند. اما در زمان اجرا بسیار <b>کندتر</b> است.	کامپایلر زمان زیادی را برای تجزیه و تحلیل کد برنامه صرف می کند. اما، در زمان اجرا بسیار <b>سریعتر</b> است.
مفسر کد واسط تولید <b>نمی کند</b> لذا از نظر مصرف حافظه <b>بهتر</b> می باشد	کامپایلر کد واسط تولید <b>می کند</b> لذا از نظر مصرف حافظه <b>بدتر</b> می باشد
Python و Matlab از مفسر استفاده می کنند.	C، C++ و Java از کامپایلر استفاده می کنند.

## IDE یا محیط توسعه یکپارچه (Integrated Development Environment) :

یک محیط برنامه نویسی است که مجموعه ای از نرم افزارهای که معمولاً شامل ویرایشگر کد برنامه (Code Editor)، کامپایلر یا مفسر، اشکال زدا (Debugger) و در نهایت رابط گرافیکی یا GUI می باشند را در یکجا جمع کرده است و در اختیار برنامه نویس قرار می دهد.



- ❑ برای زبان C++ محیط های IDE های مختلفی وجود دارد که استفاده از آنها متناسب با سلیقه افراد می باشد.
- ❑ codeblocks و vs code دو نمونه از نمونه های مختلف IDE جهت زبان C++ می باشند.
- ❑ برخی از IDE ها توسط چندین زبان مورد استفاده قرار میگیرند نظیر vs code

# متغیر – variable

## متغیر (variable):

فضایی از حافظه اصلی است که ۱- یک نام دارد ۲- دارای نوع است از متغیر ها برای ذخیره و بازیابی مقادیر استفاده میکنیم

## شناسه (identifier):

نامی است برای مشخص نمودن متغیرها، توابع و ... مورد استفاده قرار میگیرد. شناسه های مجاز با **حروف** شروع شده و در ادامه میتواند **حروف**، **رقم** یا **\_** قرار گیرد. جای خالی بین حروف شناسه مجاز نیست و

C++ یک زبان Case Sensitive است یعنی A و a را یکی نمی داند.

چهار نام مقابل برای C++ متفاوت در نظر گرفته میشود AB Ab aB ab بلند ترین طول مجاز برای نام یک متغیر ۳۱ کاراکتر است.

انتخاب نامهایی که با کلمات کلیدی C++ یکی باشد مجاز نیست. برخی از این کلمات (if for while end)

نامهای زیر برای نام گذاری متغیر ها در C++ معتبر نمی باشند

(کاراکتر غیر مجاز) a4\*e (کاراکتر غیر مجاز) we.4 (نام کلیدی) for (شروع با رقم) 1x

نمونه ای از چند نام معتبر

X12sx2 t2 a\_3cc3 a2323 for1 if12

قواعدی با عنوان clean code مطرح میشود که میتوان از آنها برای نام گذاری بهتر متغیر ها استفاده نمود.

برخی توصیه ها در نام گذاری متغیر ها:

□ نام متغیر ها با حروف کوچک شروع شوند

□ در نامهایی که ترکیب چند کلمه هستند حروف ابتدایی کلمات دوم به بعد حروف بزرگ باشند studentName

□ از کلمات اختصاری نا مفهوم استفاده نکنید

# متغیر - نوع و اعلان

متغیر ها بر اساس نوع مقداری که ذخیره میکنند داری انواعی هستند. چند نوع پر کاربرد متغیر ها:

حافظه لازم	مقادیر	نوع داده	
۲ بایت	32767 تا -32768	int	عدد صحیح
۲ بایت	0 تا 65535	unsigned int	
۴ بایت	-2147483648 تا 2147483647	long int	
۴ بایت	0 تا 4294967295	unsigned long int	
۱ بایت	یک کاراکتر ( مانند: @ a )	char	کاراکتر
۴ بایت	7 رقم دقت اعشار	float	اعشار دار
۸ بایت	15 رقم دقت اعشار	double	
۱۰ بایت	19 رقم دقت اعشار	long double	

**اعلان متغیر:** قبل از آنکه به متغیرها مقداری تخصیص داده شود و از آنها استفاده گردد بایستی آنها را در برنامه اعلان نمود.

به عبارت دیگر باید قبل از استفاده، فضایی از حافظه با نام متغیر اختصاص داده شود.

بدیهی است که این اختصاص فقط به معنی دریافت حافظه است و مقدار دهی به آن باید در برنامه انجام پذیرد. البته میتوان در خط اعلان به متغیر نیز مقدار دهی اولیه نمود.

**انتهای دستورات در زبان C++ باید علامت ; قرار داده شود. ; نام متغیر نوع متغیر**

اختصاص فضایی به طول ۲ بایت با نام x جهت ذخیره اعداد صحیح  
float a , b = 1.52 ; b جهت ذخیره اعداد دارای اعشار و قرار دادن عدد 1.52 در متغیر b  
اختصاص دو فضا به طول ۴ بایت با نامهای a و b جهت ذخیره اعداد دارای اعشار و قرار دادن عدد 1.52 در متغیر b  
اختصاص سه فضا به طول ۱ بایت با نام ch و k و first جهت ذخیره کاراکتر و قرار دادن کاراکتر f در متغیر k  
char ch , k='f' , first ;

# ساختار یک برنامه ساده به زبان C++

```
#include<iostream>
#include<header فایل>

main()
{
    تعریف متغیر ها

    دستورات برنامه
}
```

**iostream** کتابخانه ای است که توابع پایه ای زبان C++ در آن قرار دارد و در برنامه های C++ این کتابخانه را لازم است اضافه کنیم.

**iostream** برای توابع ورودی و خروجی مورد استفاده است

با توجه به توابع مورد استفاده در برنامه لازم است فایل های **header** مربوطه را اضافه نماییم.

## برخی توابع مهم و فایل های header آنها

### <iostream>

rand()	تولید عدد صحیح تصادفی
--------	-----------------------

### <conio.h>

getch()	با فشردن هر کلید از صفحه کلید، کد اسکی کلید دریافت می شود
---------	---

### <time.h> <stdlib.h>

srand(time(NULL))
با اجرای این دستور اعداد تصادفی تابع rand در اجراهای مختلف متفاوت خواهد شد

### <math.h>

sqrt(a)	$\sqrt{a}$
pow(a,b)	$a^b$
floor(a)	گرد به سمت عدد کوچکتر
round(a)	گرد به سمت عدد نزدیک
exp(a)	$e^a$
abs(a)	$ a $

### <iomanip>

setw(k);	در دستور cout استفاده میشود و قبل از هر قسمت چاپ تعیین میکند که این قسمت از محل مکان نما در محدوده k کارامتر از راست نوشته شود
setprecision(k)	در دستور cout استفاده میشود و تعداد رقم اعشار (با احتساب خود . ممیز) تعیین می کند



## عملگر انتساب =

برای مقدار دهی به یک متغیر از عملگر = استفاده میکنیم. مقدار = نام متغیر

1- عدد	<code>x = 15 ;</code>	عدد 15 را در متغیر x قرار میدهد
2- متغیر	<code>x = y ;</code>	محتوای متغیر y را در متغیر x قرار میدهد
3- کاراکتر	<code>s = 'y' ;</code>	کاراکتر y را در متغیر s قرار میدهد
4- فرمول	<code>x = 2 * a + 5 ;</code>	حاصل عملیات را در متغیر x قرار میدهد

برای کاراکترها لازم است آن را درون '' قرار دهیم وگرنه برنامه آن را به صورت نام متغیر در نظر میگیرد به تفاوت مفهوم `x=y` و `s='y'` دقت نمایید.

میتوان از چند عملگر انتساب در یک دستور استفاده نمود. عدد 15 را در سه متغیر x, y و a قرار میدهد `x = y = a = 15 ;`

نام متغیر `std::cin>>` ;

`std::cin>>a;`

`std::cin>>x>>y>>z;`

دستور ورودی C++ (در iostream قرار دارد)

دستور خروجی C++ (در iostream قرار دارد)

مقدار `std::cout<<` ;

1- عدد	<code>std::cout&lt;&lt;15 ;</code>	چاپ عدد 15
2- متغیر	<code>std:: cout&lt;&lt;a;</code>	محتوای متغیر a را چاپ میکند
3- فرمول	<code>std:: cout&lt;&lt; 2*a+5;</code>	حاصل عملیات را چاپ میکند
4- رشته	<code>std:: cout&lt;&lt;"salam" ;</code>	عبارت salam را چاپ میکند

دستورات  
`x=15;`  
`std::cout<<"num="<<x*2<<"\n"<<x;`

خروجی  
`num=30`  
`15`

In the name of GOD

□ برنامه ای بنویسید که عبارت رو به رو را چاپ کند.

```
#include<iostream>
main()
{
    std::cout<<"In the name of GOD";
}
```

```
#include<iostream>
using namespace std;
main()
{
    cout<<"In the name of GOD";
}
```

روش دوم

□ برنامه ای بنویسید که در متغیر a از نوع float عدد 3.141565 را قرار داده و آن را با 2 رقم اعشار در محدوده 8 خانه چاپ کند. (مطالعه آزاد)

```
#include<iostream>
#include<iomanip>
using namespace std;
main()
{
    float a=3.141565;
    cout<<setw(8)<<setprecision(3)<<a;
}
```

<iomanip>	
setw(k);	در دستور cout استفاده میشود و قبل از هر قسمت چاپ تعیین میکند که این قسمت از محل مکان نما در محدوده k کارامتر از راست نوشته شود
setprecision(k)	در دستور cout استفاده میشود و تعداد رقم اعشار (با احتساب خود . ممیز) تعیین می کند



## چند برنامه ساده (۲)

□ یک عدد گرفته آن را چاپ کند.

```
main()
{
    int a;
    cin>>a;
    cout<<a;
}
```

```
main()
{
    int a;
    cout<<"a:";
    cin>>a;
    cout<<"your number is "<< a;
}
```

روش دوم

```
main()
{
    int a;
    float b;
    cin>>a>>b;
    cout<<setprecision(a+1)<<b;
}
```

□ یک صحیح و یک اعشاری گرفته عدد اعشار را با تعداد رقم اعشاری مشخص شده توسط عدد صحیح چاپ کند. (مطالعه آزاد)

خروجی	b	a
15.215	15.214947	3

```
main()
{
    int a,b;
    cin>>a;
    b=a*2;
    cout<<b;
}
```

□ یک عدد گرفته دو برابر آن را چاپ کند.

```
main()
{
    int a;
    cin>>a;
    cout<<2*a;
}
```

روش دوم

تهیه و تنظیم : محمد نعیمی

# مقادیر کاراکتری

مقادیر کاراکتری در زبان C++ درون ' ' قرار میگیرند. برای ذخیره داده های کاراکتری در کامپیوتر از کد ASCII استفاده میشود. در حقیقت برای هر کاراکتر کد منحصر به فردی در نظر گرفته می شود.

مثلا کداسکی کاراکتر 'A' عدد 65 یا کداسکی کاراکتر '0' عدد 48 می باشد. ارقام هم میتوانند ماهیت کاراکتری داشته باشند هم ماهیت عددی مثلا عدد 15 هم میتواند یک عدد باشد با ارزش 15 و هم میتواند از دو کاراکتر '1' و '5' تشکیل شده باشد. کد اسکی در مقادیر دارای ترتیب به صورت متوالی میشود یعنی 'B' دارای کد 66 و رقم '1' دارای کد اسکی 49

□ به تعدادی کاراکتر در کنار هم رشته می گوئیم و آن را درون " " قرار میدهیم. مثل

"alireza" یا "152" یا "I love c++"

در زبان C++ متغیر رشته ای نداریم و در فصل آرایه ها نحوه مدیریت رشته ها را توضیح خواهیم داد.

□ **کاراکترهای کنترلی:** برخی کاراکتر ها به صورت ترکیب یک کاراکتر با \ می باشد که معنی انجام یک عملیات یا یک کاراکتر را میدهد. این کاراکتر ها در هنگام دستور چاپ یا درون برنامه داری معنی خاصی هستند. مثلا برای اینکه کاراکتر \ را بخواهیم نشان دهیم لازم است \ را بنویسیم.

\n یا endl	برو به خط جدید
\t	فضای خالی به اندازه tab
\b	یک خانه به عقب برو
\a	صدای بوق سیستم
\"	کاراکتر "
\'	کاراکتر '
\0	کاراکتر NULL (کاربرد در رشته ها)
\?	کاراکتر ?
\\	کاراکتر \

در ابتدای endl نباید \ گذاشته شود. این عبارت همان کاربرد \n را دارد و برخلاف سایر کاراکتر های کنترلی برای استفاده درون " " قرار نمیگیرد. عملا کاراکتر کنترلی نیست اما برای رفتن به خط بعد در دستور چاپ کاربرد دارد.

# کاراکترهای کنترلی – مثال

کافیست برای تشخیص مفهوم رشته دارای کاراکتر کنترلی هر \ را به کاراکتر بعدی یکی در نظر گرفت در مثالهای زیر هر \ با کاراکتر بعدی با یک رنگ مشخص شده جهت درک بهتر.

معنای عبارت (چیزی که اگر عبارت را چاپ کنیم نمایش داده میشود)	عبارت در زبان C++
mohammad aimi یعنی برو خط بعد \n	cout<<"mohammad\naimi"; mohammad\naimi
\\ ابتدا معادل \ بوده و ابتدا \nbc نوشته میشود با رسیدن به \n مکان نما به خط بعد میرود سپس df نوشته میشود سپس دو \ چاپ میشود سپس با رسیدن به \t به تعداد tab جای خالی گذاشته و ادامه عبارت نوشته میشود. (رنگی نمودن عبارات برای فهم بهتر مطلب است)	cout<<"\\nbc\\ndf\\\\\\t\\'k4\\'p"; "\\nbc\\ndf\\\\\\t\\'k4\\'p"
ابتدا abc چاپ و با رسیدن به \b مکان نما یک خانه به عقب میرود یعنی روی c قرار میگیرد و سپس حرف e جای آن نوشته میشود و در آخر هم f بعد از آن چاپ میشود.	cout<<"abc\bef"; "abc\bef"
پس از چاپ ab با زدن یک بوق عبارت cd در ادامه نوشته می شود	cout<<"ab\acd"; "ab\acd"

# عملگر های ریاضی و انتساب

عملگر	مفهوم	عبارت ریاضی	عبارت C++
+	جمع دو عدد	$x + y$	$x+y$
-	عدد اول منهای عدد دوم	$x - y$	$x-y$
*	ضرب دو عدد	$xy$	$x*y$
/	عدد اول تقسیم بر عدد دوم	$\frac{x}{y}$ $x \div y$ $x/y$	$x/y$
%	باقی مانده تقسیم عدد اول بر دوم	$x \bmod y$	$x\%y$

**نکته مهم:** اگر دو عملوند مربوط به عملگر / ، عدد صحیح باشند حاصل این عملیات خارج قسمت صحیح (DIV) می باشد و اگر حداقل یکی از آنها عدد اعشار دار باشد تقسیم دقیق انجام میشود.

$$\underbrace{9/2}_{4} \quad \underbrace{9.0/2}_{4.5} \quad \underbrace{9/2.0}_{4.5} \quad \underbrace{9.0/2.0}_{4.5}$$

## عملگر های توسعه یافته

عملگر	عبارت نمونه	عبارت معادل	توضیح
+=	$x+=a$	$x=x+a$	مقدار a را به مقدار x اضافه کن و در x ذخیره کن
-=	$x-=a$	$x=x-a$	مقدار a را از مقدار x کم کن و در x ذخیره کن
*=	$x*=a$	$x=x*a$	مقدار a را در مقدار x ضرب کن و در x ذخیره کن
/=	$x/=a$	$x=x/a$	مقدار x را به مقدار a تقسیم کن و در x ذخیره کن

# عملگر های افزایشی – کاهشی درون عبارتی

نام عملگر	عملگر	عبارت نمونه	توضیح
پیش افزایش	++	++a	یک واحد به a اضافه کن سپس از این مقدار جدید a در عبارتی که a در آن بکار رفته استفاده کن
پس افزایش	++	a++	از مقدار فعلی a در عبارتی که a در آن بکار رفته استفاده کن و سپس یک واحد به a اضافه کن
پیش کاهش	--	--a	یک واحد از a کم کن سپس از این مقدار جدید a در عبارتی که a در آن بکار رفته استفاده کن
پس کاهش	--	a--	از مقدار فعلی a در عبارتی که a در آن بکار رفته استفاده کن و سپس یک واحد از a کم کن

	مثال ۱	مثال ۲	مثال ۳	مثال ۴
	x=4; y=7; c=x++*y--;	x=4; y=7; c=++x*y--;	x=4; y=7; c=x++*--y;	x=4; y=7; c=++x*--y;
x	5	5	5	5
y	6	6	6	6
c	28	35	24	30

اگر شما واقعا یک مسئله را به خوبی درک کنید  
پاسخ مسئله از درون آن استخراج میشود  
زیرا پاسخ جدای از مسئله نیست

## Krishnamurti

### سوال هوش:

فردی ساعت 7:30 AM از چادر خود خارج شده ابتدا 10 کیلومتر به سمت جنوب حرکت میکند. سپس 20 کیلومتر به سمت شرق رفته سپس 10 کیلومتر به سمت جنوب رفته و 1 ساعت استراحت میکند. پس از صرف نهار 10 کیلومتر به سمت غرب رفته سپس 20 کیلومتر به سمت شمال رفته و به چادر خود میرسد. در چادرش یک خرس میبیند. خرس چه رنگی است؟

راهنمایی: پاسخ سوال را با دقت به مسیر طی شده پیدا خواهید کرد.



# اولویت عملگر ها

گاهی ترتیب اعمال عملگرها ممکن است باعث پاسخهای متفاوتی گردد.

$$5+2*3 \begin{cases} \xrightarrow{\text{اول ضرب بعد جمع}} 5+6=11 \\ \xrightarrow{\text{اول جمع بعد ضرب}} 7*3=21 \end{cases}$$

$$5-2+3 \begin{cases} \xrightarrow{\text{اول تفریق بعد جمع}} 3+3=6 \\ \xrightarrow{\text{اول جمع بعد تفریق}} 5-5=0 \end{cases}$$

جدول زیر برای اولویت عملگر ها مشخص گردید است (عملگر های در یک سطح، از نظر اولویت با هم برابرند)

-x	علامت منفی عدد مانند
%, *	ضرب و تقسیم
+	جمع و تفریق

**بالاترین اولویت**

**پایین ترین اولویت**

**قوانین اولویت عملگر ها**

الف) بین دو عملگر با اولویت متفاوت آنکه اولویتش بالاتر است ابتدا اعمال میشود.

ب) بین دو عملگر هم اولویت آنکه در سمت چپ عبارت قرار دارد ابتدا اعمال میشود.

اگر بخواهید بر خلاف اولویت جدول عملگر ها عملگری زودتر اعمال شود از پرانتز گذاری استفاده میکنیم.

**مثال:** در عملیات  $(a+b)/2$  اولویت جمع کمتر از تقسیم است اما با پرانتز گذاری جمع زودتر اعمال میشود

**نکته:** برای نوشتن عبارات ریاضی کافی است کل صورت را در یک پرانتز تقسیم بر کل مخرج در یک پرانتز کرد (مخرج) / (صورت) و در سایر موارد همان شکل ریاضیاتی فرمول معمولاً بدون پرانتز گذاری استفاده میشود.

مثال:

$$\frac{5y - 3x}{2a}$$

$$(5*y-3*x) / (2*a)$$

با توجه به اولویت ها نوشتن بسیاری از فرمول ها مثل چند جمله ای ها نیازی به پرانتز گذاری ندارد مگر آنکه در فرمول ریاضی آن پرانتز باشد

برای محاسبه دستی و تشخیص چگونگی عملکرد یک فرمول کفایت ابتدا از چپ به راست تمام توانها را اعمال کنیم در مرحله بعد از چپ به راست تمام ضرب و تقسیم ها و در مرحله سوم از چپ به راست تمام جمع و تفریق ها را اعمال کنیم و نتیجه نهایی همان نتیجه برنامه خواهد بود

# برنامه مبتنی بر فرمول

طول و عرض مستطیل را گرفته و مساحت آن را چاپ کند.

```
main()
{
    int tol,arz,s;
    cin>>tol>>arz;
    s=tol*arz;
    cout<<s;
}

main()
{
    int tol,arz;
    cin>>tol>>arz;
    cout<<tol*arz;
}
```

**روش دوم**

میخواهیم دوریک استخر مستطیل شکل نرده بکشیم.  
برنامه ای بنویسید که طول نرده مورد نیاز را چاپ کند

```
main()
{
    int tol,arz;
    cin>>tol>>arz;
    cout<<2*(tol+arz);
}
```

راهنمایی:  
- مسئله محاسبه محیط  
مستطیل است  
- بحث اولویت عملگر ها  
در فرمول محیط

**#define pi 3.14**

```
main()
{
    int r;
    cin>>r;
    cout<<r*r*pi;
}
```

```
main()
{
    const float pi=3.14;
    int r;
    cin>>r;
    cout<<r*r*pi;
}
```

```
main()
{
    int r;
    cin>>r;
    cout<<r*r*3.14;
}
```

شعاع دایره را گرفته  
مساحت آن را چاپ نماید.  
(به ۳ روش)

- دستور **<عبارت ۲>** **<عبارت ۱>** **#define**، هنگام کامپایل هر جا **عبارت ۱** در برنامه بود را با **عبارت ۲** جایگزین میکند و سپس برنامه را کامپایل میکند. لذا در این حالت اصلا بحث تعریف متغیر pi نیست فقط جایگزینی با عدد 3.14 انجام میشود
- عبارت **const** قبل از خط تعریف متغیر تنها باعث میشود امکان تغییر متغیر را نداشته باشیم و به صورت مقدار ثابت می شود.

## برنامه مبتنی بر فرمول (۲)

```
main()
{
    int gh , t ;
    float mt;
    cin>>gh;
    cin>>t;
    mt=gh*(t/100.0);
    cout<< gh-mt;
}
```

```
main()
{
    int gh , t;
    cin>>gh;
    cin>>t;
    cout<< gh*((100-t)/100.0) ;
}
```

مبلغ کالا و درصد تخفیف را گرفته  
مبلغ پرداختی را چاپ کند.

gh    **قیمت**  
t      **تخفیف**  
mt    **مبلغ تخفیف**

ضرایب معادله درجه دو را گرفته و ریشه های آن را بدست آورد (فرض کنید دلتا بزرگتر از صفر می باشد)

**#include<math.h>**

main()

```
{
    int a,b,c,delta;
    double x1,x2;
    cin>>a>>b>>c;
    delta=b*b-4*a*c;
    x1=(-b+sqrt(delta))/(2*a);
    x2=(-b-sqrt(delta))/(2*a);
    cout<<"x1="<<x1<<"\n";
    cout<<"x2="<<x2;
}
```

**رادیگال : sqrt(x)**  
**برای استفاده از این تابع کتابخانه**  
**math.h نیز باید include شود**

**تحلیل مسئله:**

**ابتدا باید ضرایب معادله (a,b,c) را گرفت**

**دلتا را بر اساس فرمول مقابل به دست می آوریم  $\Delta = b^2 - 4ac$**   
**با فرض مثبت بودن  $\Delta$  دو ریشه داریم طبق فرمولهای زیر**

$$x_1 = \frac{-b + \sqrt{\Delta}}{2a}$$

$$x_2 = \frac{-b - \sqrt{\Delta}}{2a}$$

**چاپ نتایج**

## برنامه مبتنی بر فرمول (۳)

تعداد روز را گرفته مشخص کند چند ماه، هفته و روز است

```
main()
{
    int a ,m,h,r ;
    cin>>a;
    m=a / 30;
    a=a % 30;
    h=a / 7;
    r=a % 7;
    cout<<m <<" "<< h <<" "<< r;
}
```

از انجایی که یک ماه تعداد دقیقی از یک هفته نیست (۴ هفته و دو روز میشود یک ماه) لذا روش اول صفحه قبل قابل استفاده نیست. اما همه مضربی از روز هستند لذا به روش دوم یعنی از بالا به سمت پایین حساب میکنیم یعنی اول ماه بعد هفته و هر چه ماند می شود روز

a	m	h	r
259	8		
19		2	
			5

مکان فعلی و سرعت خودرو را گرفته و با گرفتن زمان، مکان نهایی خودرو را مشخص کند.

```
main()
{
    int v,t,x0,x;
    cin>>v>>t>>x0;
    x=v*t+x0;
    cout<<x;
}
```

تحلیل مسئله:

ابتدا باید ورودی ها را دریافت نمود

مکان نهایی خودرو بر اساس فرمول زیر محاسبه میشود

$$x = vt + x_0$$

چاپ نتایج

## برنامه مبتنی بر فرمول (۴)

main()

{

long a , d,h,m,s ;

cin>>a;

s=a % 60; هر 60 ثانیه میشود 1 دقیقه لذا هر چه باقی بماند میشود ثانیه های اضافی ما

a=a / 60; هر 60 ثانیه میشود 1 دقیقه پس کل دقیق ما میشود. از این لحظه a دقیق است

m=a % 60; هر 60 دقیقه میشود 1 ساعت لذا هر چه باقی بماند میشود دقیق اضافی ما

a=a / 60; هر 60 دقیقه میشود 1 ساعت پس کل ساعات ما میشود. از این لحظه a ساعات است

h=a % 24; هر 24 ساعت میشود 1 روز لذا هر چه باقی بماند میشود ساعات اضافی ما

d= a/24; هر 24 ساعت میشود 1 روز پس کل روزهای ما میشود.

cout<<d <<" "<< h <<" "<< m <<" "<< s;

}

**روش اول:** چون واحد ها همه مضرب دقیقی از هم هستند (هر 60 ثانیه یک دقیق-هر 60 دقیقه یک ساعت - هر 24 ساعت یک روز) عملاً همه مضربی از واحد های کوچکتر هستند، لذا باقی مانده تقسیم بر 60 میشود ثانیه هایی که نمیتوانند به واحد بزرگتر از خود یعنی دقیقه تبدیل شوند و خارج قسمت تقسیم میشود تبدیل هر 60 ثانیه به دقیقه.

**روش دوم:** از بالا به پایین محاسبه میکنیم یعنی میدانیم هر روز چند ثانیه می شود ( $24 \times 60 \times 60 = 86400$ ) لذا ابتدا خارج قسمت تقسیم بر این عدد میشود تعداد روز و باقی مانده تقسیم میشود ثانیه هایی که باید تبدیل به ساعت، دقیقه و ثانیه شوند. حال همین کار را برای تبدیل ثانیه های باقی مانده به واحد های ساعت و دقیقه و ثانیه میکنیم

a	s	m	h	d
523614	54			
8726		26		
145			1	
				6

main()

{

long a , d,h,m,s ;

cin>>a;

d=a / (24\*60\*60);

a=a %(24\*60\*60);

h=a / (60\*60);

a=a % (60\*60);

m=a / (60);

s= a%(60);

cout<<d <<" "<< h <<" "<< m <<" "<< s;

}

هر روز  $24 \times 60 \times 60$  ثانیه است. هر تعداد از این ثانیه میشود روز

مقداری که باقی بماند میشود ثانیه هایی که کمتر از روز (a)

هر ساعت  $60 \times 60$  ثانیه است. هر تعداد از این ثانیه میشود ساعت

مقداری که باقی بماند میشود ثانیه هایی که کمتر از ساعت (a)

هر دقیقه 60 ثانیه است. هر تعداد از این ثانیه میشود دقیقه

مقداری که باقی بماند میشود ثانیه هایی که کمتر از دقیقه است

a	s	m	h	d
523614				6
5214			1	
1614		26		
	54			

## برنامه مبتنی بر فرمول (۵)

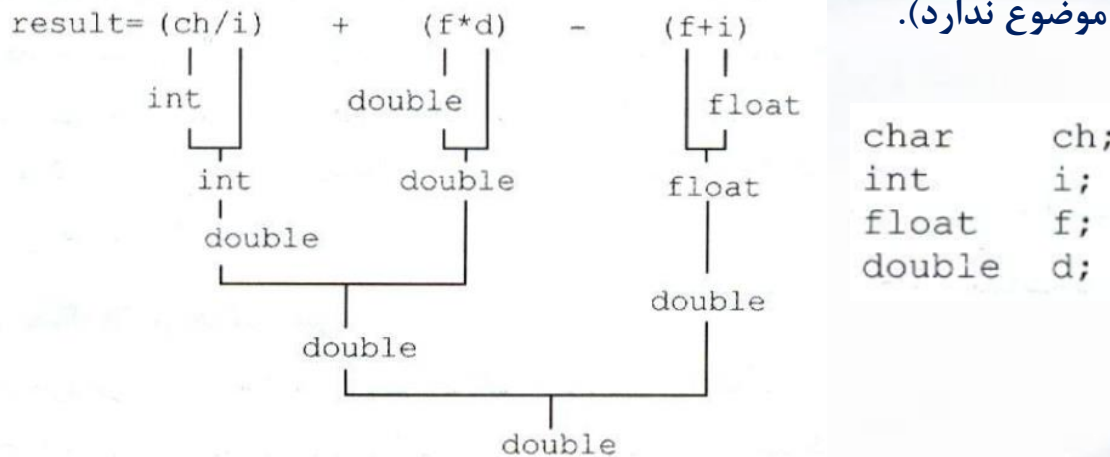
مسئله	فرمول مسئله	کد قسمت اصلی برنامه
تبدیل درجه (C) به فارنهایت (F)	$F = \frac{9}{5}C + 32$	<pre>cin&gt;&gt;c; f=9/5.0*c+32; cout&lt;&lt;f;</pre>
تبدیل فارنهایت (F) به درجه (C)	$C = \frac{5}{9}(F - 32)$	<pre>cin&gt;&gt;f; c=5/9.0*(f-32); cout&lt;&lt;c;</pre>
محاسبه بیمه B (۵٪) و مالیات M (۱۰٪) مربوط به حقوق H	$B = 0.05 * H$ $M = 0.1 * H$	<pre>cin&gt;&gt;h; b=0.05*h; m=0.1*h; cout&lt;&lt;b&lt;&lt;" "&lt;&lt;m;</pre>
محاسبه وتر (C) مثلث قائم الزاویه با داشتن دو ضلع A و B	$C = \sqrt{A^2 + B^2}$	<pre>cin&gt;&gt;A&gt;&gt;B; C=sqrt(A*A+B*B); cout&lt;&lt;C;</pre>
محاسبه ضلع مقابل (A) به زاویه $\theta$ در مثلث قائم الزاویه با وتر (C) دقت کنید که در برنامه متغیری به نام $\theta$ نداریم چون طبق قواعد، نام متغیر میتواند ترکیبی از حروف انگلیسی، ارقام و _ باشد لذا در باید نام مجاز در نظر بگیرید مثلا $t$ یا $teta$ یا هر نام مجازی	$A = C * \sin(\theta)$	<pre>cin&gt;&gt;c&gt;&gt;teta; A=c*sin(teta); cout&lt;&lt;A;</pre>



# تبدیل نوع داده

## تبدیل نوع داده در عبارات محاسباتی:

اگر در یک عملیات چند نوع متغیر داشته باشیم هنگام محاسبه همه به بزرگترین نوع تبدیل می شوند. به عنوان مثال در تقسیم دو عدد از انواع `int` و `float` هر دو به شکل `float` در محاسبه شرکت داده میشوند. (دقت کنید نوع متغیری که نتیجه عملیات در آن ذخیره میشود دخالتی در این موضوع ندارد).



## تبدیل انواع داده هنگام انتساب:

اگر نوع متغیری که مقدار متغیر دیگر یا نتیجه یک عملیات میخواهد در آن ذخیره شود متفاوت باشد دو حالت داریم.

الف) ذخیره نوع کوچکتر در نوع بزرگتر:

مشکلی بوجود نخواهد آمد و داده ای از دست نخواهد رفت مانند ذخیره `char` در `int` یا ذخیره `int` در `float`

ب) ذخیره نوع بزرگتر در نوع کوچکتر:

ممکن است قسمتی از داده را از دست بدهیم مثلاً ذخیره مقدار `float` در `int` حداقل قسمت اعشار را از دست خواهیم داد

میتوان با قالب زیر نوع داده یک متغیر را به صورت دستی در عملیات تغییر داد (نوع داده متغیر فقط در عملیات تغییر میکند و نوع داده متغیر در برنامه همان نوع تعریف شده اول می باشد)

متغیر (نوع داده مورد نظر) `(float) x`

متغیر `x` در عملیات به صورت `float` در نظر گرفته می شود

## تبدیل نوع داده (۲)

نوع و مقدار متغیر ها برای چهار دستور زیر در نظر گرفته شود

```
int x=2, y=9, m;  
float k;
```

```
k=y/x;
```

**x** و **y** هر دو از نوع **int** هستند و تقسیم (**div**) خواهد بود و اینکه **k** از نوع **float** است تاثیری در عملیات ندارد.

ذخیره نتیجه عملیات که **int** است در متغیر **float** ایجاد مشکلی نخواهد کرد و **4.0** در **k** ذخیره میشود

```
m=y/x;
```

**x** و **y** هر دو از نوع **int** هستند و تقسیم (**div**) خواهد بود. (4)

ذخیره نتیجه عملیات که **int** است در متغیر **int** یعنی **4** در **m** ذخیره میشود

```
k=(float)y/x;
```

**y** به فرم **float** در عملیات شرکت میکند پس عملیات تقسیم با اعشار خواهد بود. (4.5)

نتیجه عملیات **float** است و **4.5** در **k** ذخیره میشود

```
m=y/(float)x;
```

**x** به فرم **float** در عملیات شرکت میکند پس عملیات تقسیم با اعشار خواهد بود (4.5)

ذخیره نتیجه عملیات که **float** است در متغیر **int** باعث حذف رقم اعشار می شود و **4** در **m** ذخیره میشود

یک کاراکتر گرفته کاراکتر بعدی آن را چاپ کند.(سه روش)

```
main()  
{  
    char ch;  
    cin>>ch;  
    cout<<(char)(ch+1);  
}
```

```
main()  
{  
    char ch;  
    cin>>ch;  
    ch++;  
    cout<<ch;  
}
```

```
main()  
{  
    char ch;  
    cin>>ch;  
    cout<<++ch;  
}
```

یک کاراکتر گرفته  
کد اسکی آن را چاپ کند

```
main()  
{  
    char ch;  
    cin>>ch;  
    cout<<(int)ch;  
}
```

سه عدد گرفته میانگین اعداد را چاپ نماید.

$$\text{میانگین} = \frac{a+b+c}{3}$$

```
main()
{
    int a,b,c,sum;
    float avg;
    cin>>a>>b>>c;
    sum=a+b+c;
    avg=sum/3.0;
    cout<<avg;
}

main()
{
    int a,b,c;
    float avg;
    cin>>a>>b>>c;
    avg=(a+b+c)/3.0;
    cout<<avg;
}

main()
{
    int a,b,c;
    cin>>a>>b>>c;
    cout<<(a+b+c)/3.0;
}

main()
{
    int a,b,c,sum;
    cin>>a>>b>>c;
    sum=a+b+c;
    cout<<sum/3.0;
}
```

اگر بجای تقسیم بر 3.0 بر 3 تقسیم کنیم با دقت اعشار محاسبه نمیشود. چون اعداد و sum از نوع int هستند. (بحث نوع داده ها در عملیات)

اگر صورت را در پرانتز نگذاریم ابتدا عدد سوم تقسیم بر 3.0 می شود بعد با دو عدد اول جمع (بحث اولویت عملگرها) در حالی که قرار است ابتدا سه عدد جمع شوند بعد حاصل بر 3.0 تقسیم شود.

برنامه هایی که جواب آنها پاسخ فوق است (سه عدد گرفته میانگین اعداد را چاپ نماید). دقت کنید که در برخی حالات لازم است نوع سه متغیر a,b,c از نوع float باشد.

میانگین سنی سه نفر را چاپ کند.

متوسط بارندگی در سه روز را چاپ کند.

میانگین حقوق دریافتی سه استاد دانشگاه

معدل نمرات دانش آموزی که سه درس دارد.

متوسط یا میانگین سه عدد جوابی مشابه مسئله فوق دارد

متوسط متراژ سه ویلا

متوسط شیر سه گاو

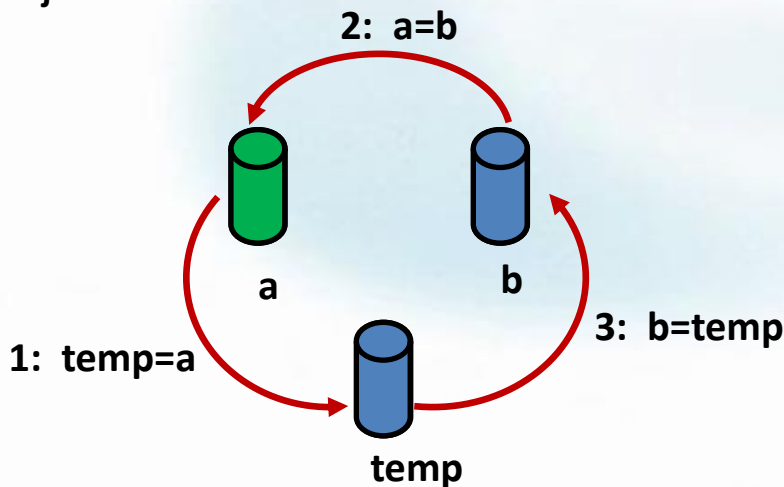
متوسط وزن سه فیل

میانگین وزن سه نان

دو متغیر گرفته و محتوای آن دو را تغییر دهید.

تحلیل: فرض کنید محتوی یک لیوان **a** و لیوان **b** را قرار است جا بجا کنیم. برای این کار نیاز به لیوان سوم داریم نام آن را **temp** میگذاریم و به شکل زیر کار را انجام میدهیم

```
main()
{
    int a,b,temp;
    cin>>a>>b;
    temp=a;
    a=b;
    b=temp;
    cout<<a<<" "<<b;
}
```



نمره و تعداد واحد سه درس یک دانشجو را گرفته معدل او را حساب کند

$$avg = \frac{\sum_{i=1}^n (w_i * grade_i)}{\sum_{i=1}^n w_i}$$

فرمول میانگین وزنی

معدل ۳ درس (واحدی)

$$avg = \frac{vahed_1 * num_1 + vahed_2 * num_2 + vahed_3 * num_3}{vahed_1 + vahed_2 + vahed_3}$$

```
main()
{
    float num1,num2,num3;
    int v1,v2,v3;
    cout<<"num1 and v1:";
    cin>>num1>>v1;
    cout<<"num2 and v2:";
    cin>>num2>>v2;
    cout<<"num3 and v3:";
    cin>>num3>>v3;
    cout<<((num1*v1+num2*v2+num3*v3)/(v1+v2+v3));
}
```

چون نمرات اعشاری (float) هستند، عملیات صورت کسر به فرم float خواهد بود و لازم نیست صورت یا مخرج به فرم float تبدیل شوند و تقسیم با دقت اعشار محاسبه می شود.

```
float sorat=num1*v1+num2*v2+num3*v3;
int makhraj=v1+v2+v3;
cout<<sorat/makhraj;
```

معادل دستور بالا