اصول برنامه سازی
به زبان C++

فصل اول – مبانی

تهیه و تنظیم : محمد نعیمی

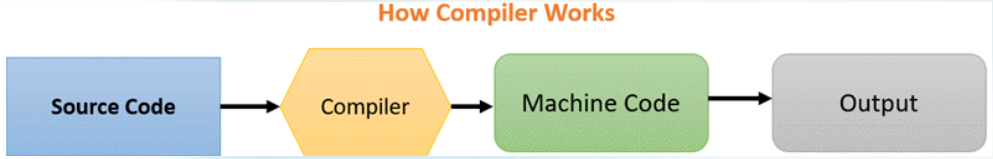
عضو هیات علمی دانشگاه آزاد اسلامی

مفاهیم پایه ای در برنامه نویسی

یک «برنامه» در حقیقت دستورالعمل‌های متوالی است که می‌تواند توسط یک کامپیوتر اجرا شود.
معمولا از **زبانهای سطح بالا**(high level language) برای برنامه نویسی استفاده می شود.
زبان سطح بالا: به زبانهایی گفته می شود که دستورات آن برای انسان قابل فهم است یا به بیان دقیق تر شامل کلمات و عبارات معمول در زبانهایی مثل انگلیسی و ... می باشد.
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 برنامه‌نویسی 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 نمی تواند 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 نمی شود.



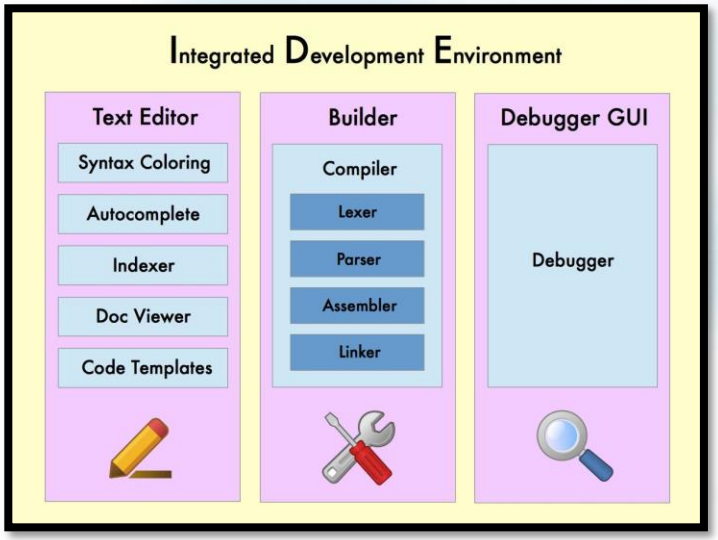
مفسر (interpreter) :
یک برنامه کامپیوتری است که **هر دستور برنامه** نوشته شده به زبان سطح بالا را به **کد ماشین** تبدیل می کند. هم کامپایلر و هم مفسر کار یکسانی را انجام می دهند که عبارت است از تبدیل زبان برنامه نویسی سطح بالا به کد ماشین اما، یک کامپایلر **قبل از اجرای برنامه**، برنامه را به کد ماشین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

تفاوت کامپایلر و مفسر

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

IDE یا محیط توسعه یکپارچه(Integrated Development Environment) :

یک محیط برنامه نویسی است که مجموعه ای از نرم افزارهای که معمولا 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 وa را یکی نمی‌داند.

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

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

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:

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

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. البتــه میتــوان در خط اعلان به متغیر نیز مقدار دهی اولیه نمود.

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

```
int x ;
```
اختصاص فضایی به طول ۲ بایت با نام x جهت ذخیره اعداد صحیح

```
float a , b = 1.52 ;
```
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

```
char ch , k='f' , first ;
```
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

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.
iostreamبرای 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 فایلهای 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 `x = 15 ;` عدد 15 را در متغیر x قرار میدهد

2- متغیر `x = y ;` محتوای متغیر y را در متغیر x قرار میدهد

3- کاراکتر `s = 'y' ;` کاراکتر y را در متغیر s قرار میدهد

4- فرمول `x = 2 * a + 5 ;` حاصل عملیات را در متغیر x قرار میدهد

برای کاراکتر ها 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. `x = y = a = 15 ;` عدد 15 را در سه متغیر x، y و a قرار میدهد

دستور ورودی C++ (در iostream قرار دارد)

; نام متغیر>>std::cin

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

دستور خروجی C++ (در iostream قرار دارد)

std::cout<<مقدار;

1- عدد `std::cout<<15 ;` چاپ عدد 15

2- متغیر `std:: cout<<a;` محتوای متغیر a را چاپ میکند

3- فرمول `std:: cout<< 2*a+5;` حاصل عملیات را چاپ میکند

4- رشته `std:: cout<<"salam" ;` عبارت salam را چاپ میکند

دستورات

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

خروجی

```
num=30
15
```

چند برنامه ساده

- برنامه ای بنویسید که عبارت رو به رو را چاپ کند. In the name of GOD

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

روش دوم

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

- 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;
}
```

روش دوم

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

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

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

a	b	خروجی
3	15.214947	15.215

- یک عدد گرفته دو برابرآن را چاپ کند.

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

روش دوم

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

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 گوییم 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.

- **کاراکتر های 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.

تهیه و تنظیم : محمد نعیمی

کاراکتر های کنترلی– مثال

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

عبارت در زبان c++	معنای عبارت(چیزی که اگر عبارت را چاپ کنیم نمایش داده میشود)
cout<<“mohammad\naimi”; mohammad\naimi	mohammad aimi \n یعنی برو خط بعد
cout<<“\\nbc\ndf\\\\\t\”k4\’p”; “\\nbc\ndf\\\\\t\”k4\’p”	\\ 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) \nbc df\\ \”k4’p
cout<<“abc\bef”; “abc\bef”	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 اخر هم f بعد از آن چاپ میشود. abef
cout<<“ab\acd”; “ab\acd”	پس از چاپ ab با زدن یک بوق عبارت cd در ادامه نوشته می شود abcd

عملگر های ریاضی

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.

9/2 → 4

9.0/2 → 4.5

9/2.0 → 4.5

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

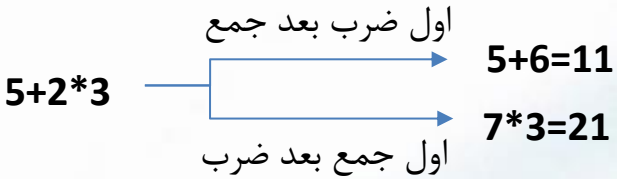
سوال هوش:

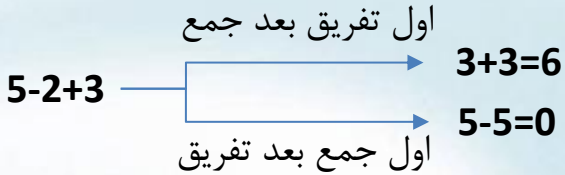
فردی ساعت 7:30 AM از چادر خود خارج شده ابتدا 10 کیلومتر به سمت جنوب حرکت میکند. سپس 20 کیلومتر به سمت شرق رفته سپس 10 کیلومتر به سمت جنوب رفته و 1 ساعت استراحت میکند. پس از صرف نهار 10 کیلومتر به سمت غرب رفته سپس 20 کیلومتر به سمت شمال رفته و به چادر خود میرسد. در چادرش یک خرس میبیند. خرس چه رنگی است؟

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.





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

علامت منفی عدد مانند	-x	بالاترین اولویت
ضرب و تقسیم	% * /	
جمع و تفریق	+ -	پایین ترین اولویت

قوانین اولویت عملگر ها

الف)بین دو عملگر با اولویت متفاوت انکه 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 کمتراز تقسیم است اما باپرانتزگذاری 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 معمولا 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 کافیست 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

میخواهیم دوریک استخرمستطیل شکل نرده بکشیم.
برنامه ای بنویسیدکه طول نرده موردنیازراچاپ کند

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

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

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;
}
```

روش دوم

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

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

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

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

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

- دستور <عبارت ۲> <عبارت ۱> #define ، هنگام کامپایل هرجا 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)

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

```
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) ;
}
```

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;
 }
```

رادیکال : 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 Δ 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.

تحلیل مسئله:

ابتدا باید ورودی ها را دریافت نمود

مکان نهایی خودرو بر اساس فرمول زیر محاسبه میشود

$$x = vt + x_0$$

چاپ نتایج

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

برنامه مبتنی بر فرمول (۴)

تعداد ثانیه را گرفته مشخص کند چند روز، ساعت،دقیقه و ثانیه است

```
main()
{
   long a , d,h,m,s ;
   cin>>a;
   s=a % 60;
   a=a /60;
   m=a % 60;
   a=a / 60;
   h=a % 24;
   d= a/24;
   cout<<d <<" "<< h <<" "<< m <<" " << s;
}
```

هر 60 ثانیه میشود 1 دقیقه لذا هرچه باقی بماند میشود ثانیه های اضافی ما

هر 60 ثانیه میشود 1 دقیقه پس کل دقایق ما میشود. از این لحظه a دقایق است

هر 60 دقیقه میشود 1 ساعت لذا هرچه باقی بماند میشود دقایق اضافی ما

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

هر 24 ساعت میشود 1 روز لذا هرچه باقی بماند میشود ساعات اضافی ما

هر 24 ساعت میشود 1 روز پس کل روزهای ما میشود.

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

روش اول: چون واحد ها همه مضرب دقیقی از هم هستند (هر ۶۰ ثانیه یک دقیق–هر ۶۰ دقیقه یک ساعت – هر ۲۴ ساعت یک روز) عملا 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 ۶۰ ثانیه به دقیقه**.

روش دوم: از بالا به پایین محاسبه میکنیم یعنی میدانیم هر روز چند ثانیه می شود (24*60*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

```
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*60*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*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)	$F=\frac{9}{5}C+32$	cin>>c; f=9/5.0*c+32; cout<<f;
تبدیل فارنهایت (F) به درجه (C)	$C=\frac{5}{9}(F-32)$	cin>>f; c=5/9.0*(f-32); cout<<c;

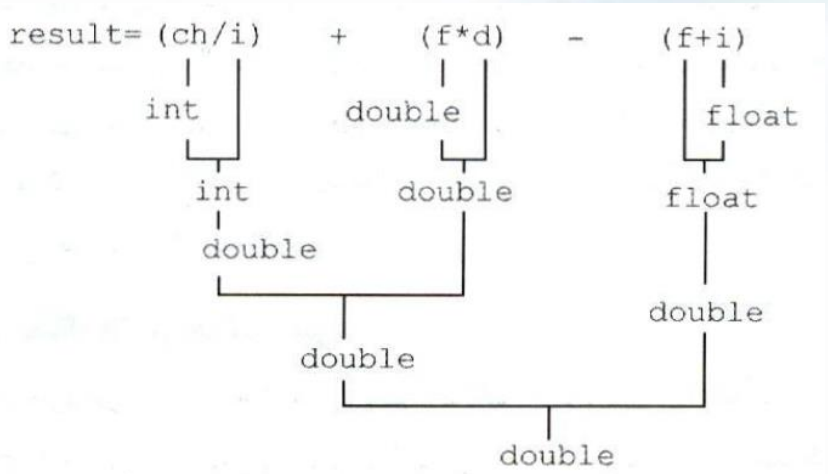
محاسبه بیمه B (۵٪) و مالیات M(۱۰٪) مربوط به حقوق H	B=0.05*H
	M=0.1*H
		cin>>h;
		b=0.05*h;
		m=0.1*h;
		cout<<b<<" "<<m;

محاسبه وتر (C) مثلث قائم الزاویه با داشتن دو ضلع A و B	$C=\sqrt{A^2+B^2}$	cin>>A>>B; C=sqrt(A*A+B*B); cout<<C;
محاسبه ضلع مقابل(A) به زاویه θ در مثلث قائم الزاویه با وتر (C) دقت کنید که در برنامه متغیری به نام θ نداریم چون طبق قواعد، نام متغیر میتواند ترکیبی از حروف انگلیسی، ارقام و _ باشد لذا در باید نام مجاز در نظر بگیرید مثلا teta یا t یا هر نام مجازی	A=C*sin(θ)	cin>>c>>teta; A=c*sin(teta); cout<<A;

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).




```
char    ch;
int     i;
float   f;
double  d;
```

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 بزرگتردر نوع کوچکتر:

ممکن است قسمتی از داده را از دست بدهیم مثلا ذخیر 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)

(float) x متغیر (نوع داده مورد نظر)

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

تبدیل نوع داده (۲)

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

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

```
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.

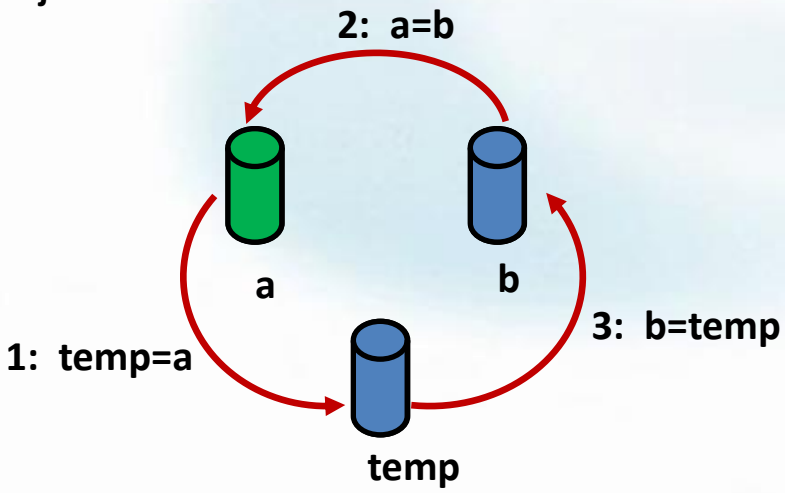
میانگین سنی سه نفر را چاپ کند.
متوسط بارندگی در سه روز را چاپ کند.
میانگین حقوق دریافتی سه استاد دانشگاه
معدل نمرات دانش آموزی که سه درس دارد.
متوسط یا میانگین سه عدد جوابی مشابه مسئله فوق دارد

متوسط متراژ سه ویلا
متوسط شیر سه گاو
متوسط وزن سه فیل
میانگین وزن سه نان

دو متغیر گرفته و محتوای آن دو ر ا 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

فرمول میانگین وزنی

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

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