به نام خدا

دانشگاه صنعتی بیرجند

گروه مهندسی کامپیوتر و فناوری اطلاعات

جزوه درس مبانی کامپیوتر و برنامه نویسی

sajaddehganzadeh78@gmail.com

❖ جایگاه این درس در رشته مهندسی کامپیوتر و مهندسی IT

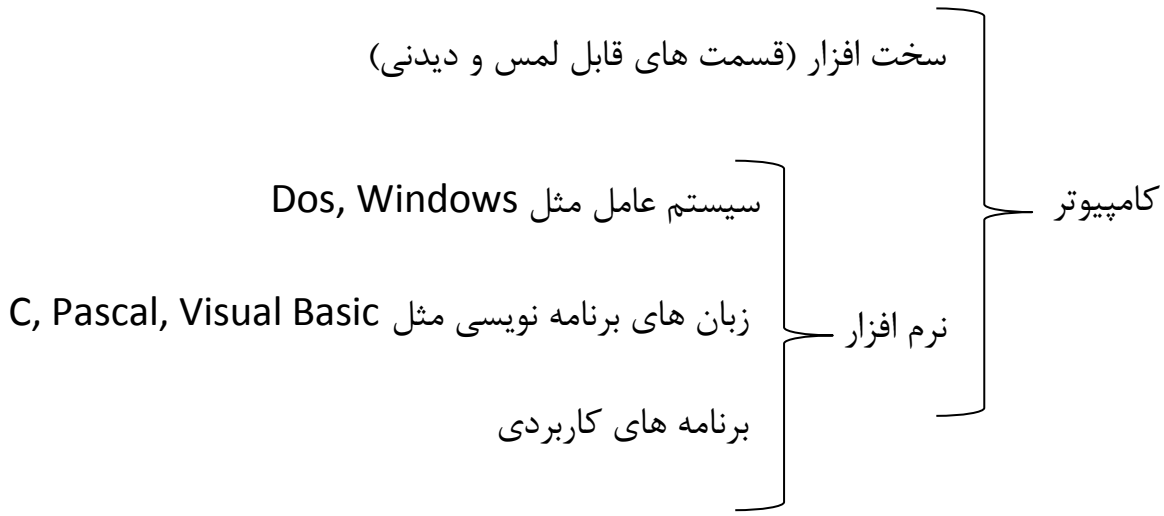
این درس اولین درس دانشگاهی رشته می باشد و نقطه شروعی برای ورود به دنیای جالب برنامه نویسی و علم و فن کامپیوتر هست . بنابراین یاد گیری اصول اولیه برنامه نویسی در این درس از جایگاه ویژه ای برخوردار است.

این درس پایه و اساس برنامه نویسی است که جزء اصول این رشته می باشد و این اصول را به فرگیران یاد می دهد . بنابراین یادگیری دقیق این درس به همراه ارائه پروژه های عملی که لازمه این درس می باشد جزء اهم مسائل می باشد .

❖ اهداف درس

- الگوریتمی برای حل مسئله ارائه دهد.
- اصول و مبانی اولیه نرم افزار و سخت افزار را بشناسد.
- اهداف و مفاهیم زبانهای برنامه نویسی را بداند.
- مفاهیم اولیه برنامه نویسی ساخت یافته را بداند و اصول لازم را در مرحله اجراء بکار ببرد .
- دستورات زبان C++ را در برنامه ها بکار ببرد.
- از توابع و روال های استاندارد زبان C++ در صورت لزوم استفاده نماید.
- از توابع ، روال ها برای جدا کردن قطعات برنامه استفاده کند.

جلسه اول: مفاهیم پایه کامپیوتر



بیت: کوچکترین واحد اطلاعاتی در کامپیوتر که می تواند مقادیر ۰ و ۱ را داشته باشد.

بایت: به مجموع ۸ بیت یک بایت گوییم. مثلاً: ۰۱۰۰۱۰۱۱

۱ کیلو بایت:۱۰۲۴ بایت

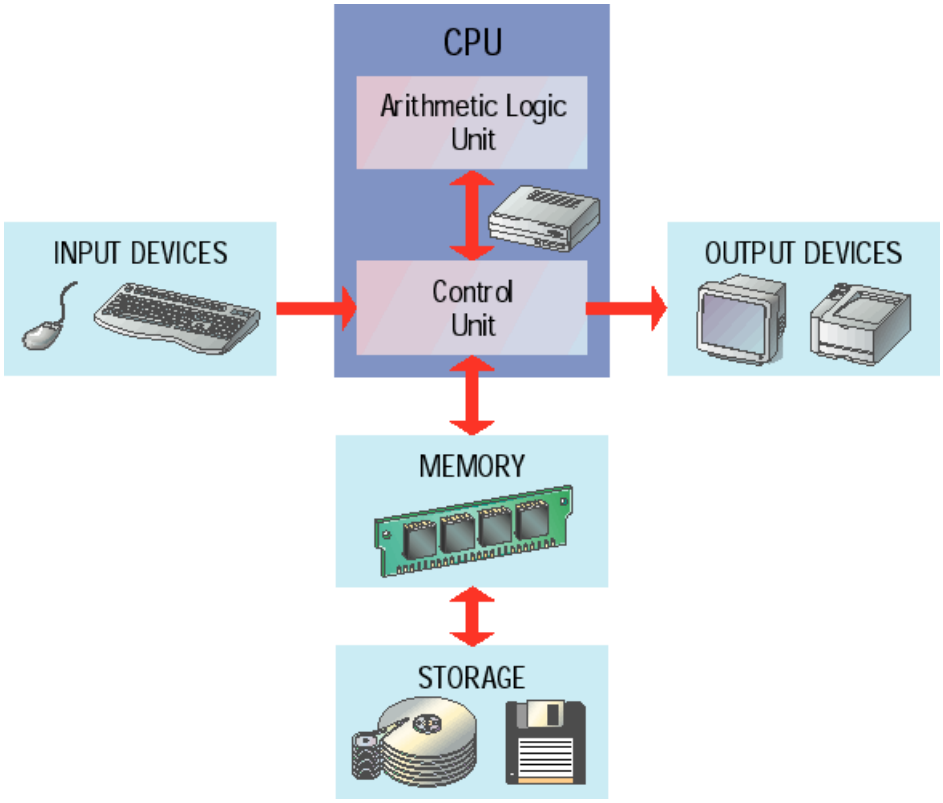
۱ مگا بایت: ۱۰۲۴ کیلو بایت

۱ گیگا بایت: ۱۰۲۴ مگا بایت

اجزای اصلی یک کامپیوتر

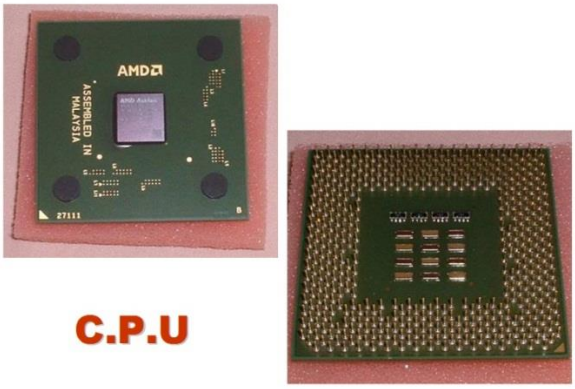
هر سیستم کامپیوتری از چهار قسمت اصلی تشکیل می شود:

۱) واحد پردازش مرکزی ۲) واحد ورودی ۳) واحد خروجی ۴) حافظه



واحد پردازش مرکزی

ریز پردازنده، تراشه ی الکترونیکی است که انجام عملیات پردازشی، منطقی، ریاضی و کنترلی را بر عهده دارد. این قسمت اصلی ترین بخش و در واقع مغز کامپیوتر محسوب می شود. نام دیگر آن Central Processing Unit می باشد.

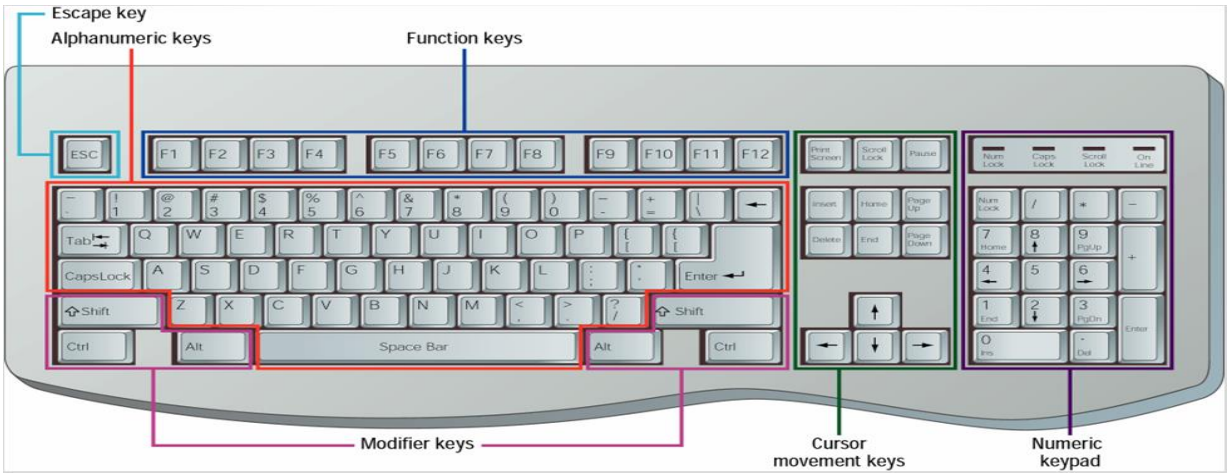


واحد ورودی

واحدی که داده ها را از دستگاه های ورودی دریافت کرده و جهت تبدیل آن ها به داده های قابل فهم کامپیوتر، آن را به واحد کنترل تحویل می دهد. قابل ذکر است که واحد کنترل بخشی از CPU می باشد. دستگاه های ورودی شامل موس، کیبورد، اسکنر، وسایل بازی و .... هستند.


Keyboard
Microsoft
Internet Keyboard


CREATIVE
MOUSE
Optical 3000
Exceptional Precision and Style at an Attractive Price
Mouse

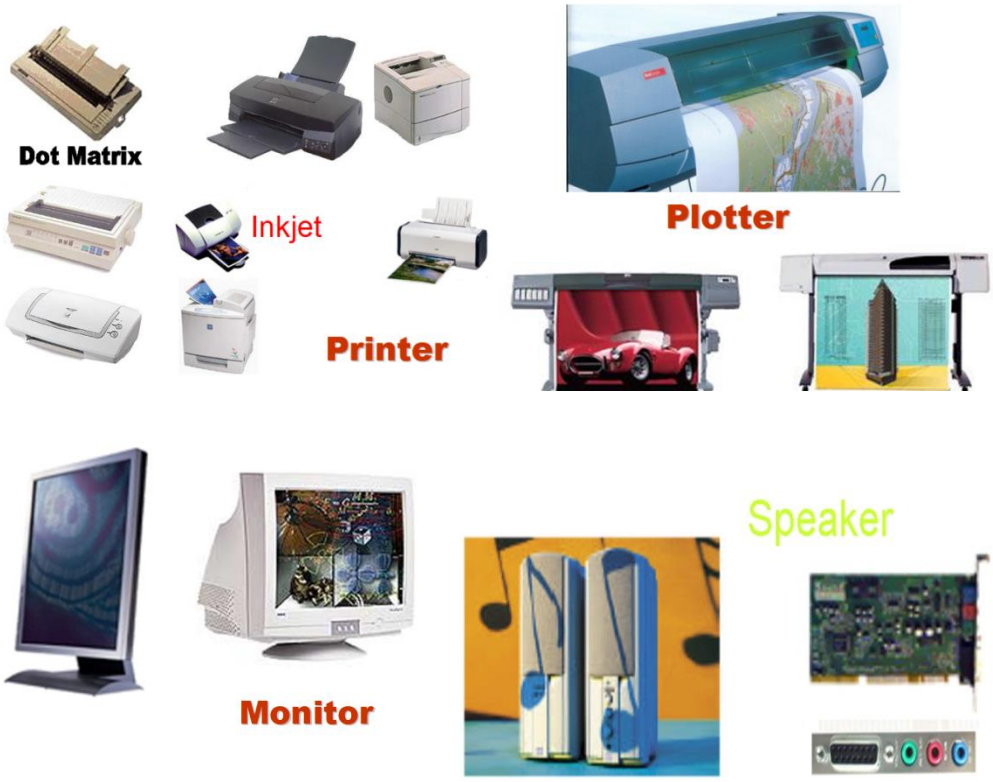

Escape key
Alphanumeric keys
Function keys
ESC
F1 F2 F3 F4 F5 F6 F7 F8 F9 F10 F11 F12
Tab
CapsLock
Enter
Shift
Ctrl
Alt
Space Bar
Modifier keys
Cursor movement keys
Numeric keypad


Scanner


Joystick

واحد خروجی

واحدی است که اطلاعات تولید شده توسط کامپیوتر را از حافظه اصلی دریافت کرده و به دستگاه های خروجی منتقل می نماید. دستگاه های خروجی عبارتند از: چاپگر، پلاتر یا نقشه کش، مونیتور یا صفحه نمایش، اسپیکر و ...



حافظه ها

حافظه مکانی است که اطلاعات موقتاً یا دائماً در آن قرار می گیرد. حافظه های درون سیستم به دو دسته ی حافظه های اصلی (Main Memory) و حافظه های جانبی یا ثانویه (Secondary Memory) تقسیم می شوند. هر برنامه برای اجرا ابتدا بایستی در حافظه ی اصلی قرار بگیرد و سپس توسط CPU اجرا شود. پس از اجرای برنامه، برای ذخیره اطلاعات برنامه، آن ها را روی حافظه ی جانبی ذخیره می کنند.

وب سایت

به مجموعه ای از صفحات وب که توسط یک سرور وب، میزبانی می شوند یک وب سایت گوییم.

مرورگر وب

یک نرم افزار کاربردی است که اسناد فرامتنی وب را پیدا می کند و آن ها را در کامپیوتر کاربر باز می نماید. اولین مرورگر وب Mosaic بود که در سال ۱۹۹۳ شروع به کار کرد.

از انواع دیگر مرورگر های وب می توان به عنوان مثال اینترنت اکسپلورر، موزیلا فایرفاکس و ... را نام برد.

ایمیل

چیزی شبیه سیستم پستی قدیمی می باشد. ما می توانیم به وسیله ی ایمیل پیام خود را کمتر از یک ثانیه به تمام دنیا بفرستیم. صدا، تصویر و فایل های مختلفی می توانند به نامه ی ما پیوست شوند. بر خلاف سیستم تلفن، این سیستم یک سیستم افلاین می باشد. یعنی در زمان ارسال نامه لازم نیست طرف مقابل پشت سیستم خود باشد.

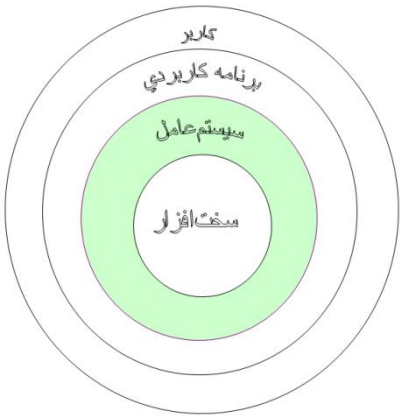
موتور جستجو

از این وسیله برای جستجوی یک تاپیک یا موضوع خاص در اینترنت استفاده می کنیم. این موتورهای جستجو از طریق مرورگرهای وب در دسترس خواهند بود. هر کلمه کلیدی یا عبارتی که وارد کنید با مجموعه ای از نتایج روبرو خواهید شد. در ادامه مثال هایی از چند موتور جستجو اورده شده است:

International search engine examples

Alta Vista http://www.altavista.com
Google: http://www.Google.com
HotBot http://www.hotbot.com
Infoseek/GO http://www.go.com/
Lycos http://www.lycos.com
MetaCrawler http://www.metacrawler.com
MSN Internet Search http://search.msn.com
Web Crawler http://www.webcrawler.com
Yahoo http://www.yahoo.com

سیستم عامل



سیستم عامل یک نرم افزار سیستمی است که در واقع ارتباط بین سخت افزار و نرم افزار کاربردی را کنترل می نماید. به عبارت دیگر به عنوان واسط سخت افزار و برنامه های کاربردی انجام وظیفه می کند.

زبان های برنامه‌نویسی

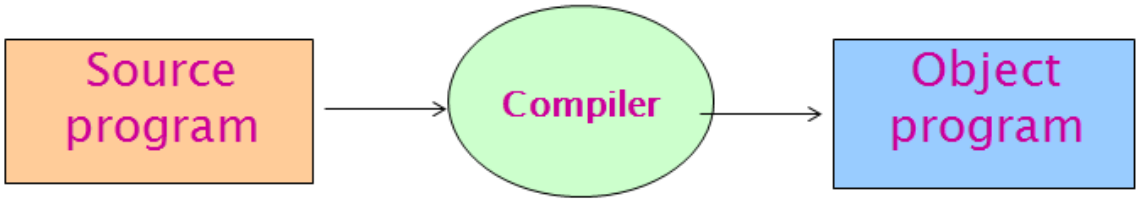
نرم‌افزارها توسط زبان های برنامه‌نویسی نوشته می‌شوند. زبان های برنامه‌نویسی، یک سیستم ارتباطی هستند که توسط آن ها می‌توان دستورات لازم را به ماشین انتقال داد.

هر زبان برنامه‌نویسی به مجموعه‌أی از علایم، قواعد و دستورالعمل‌ها گفته می‌شود که امکان ارتباط با کامپیوتر را جهت بیان کاری یا حل مسئله‌ای فراهم می‌کند.

در حالت کلی زبان های برنامه‌نویسی را به سه دسته زیر تقسیم‌بندی می‌کنند:

- زبان های سطح بالا
- زبان های سطح پایین
- زبان های سطح میانی

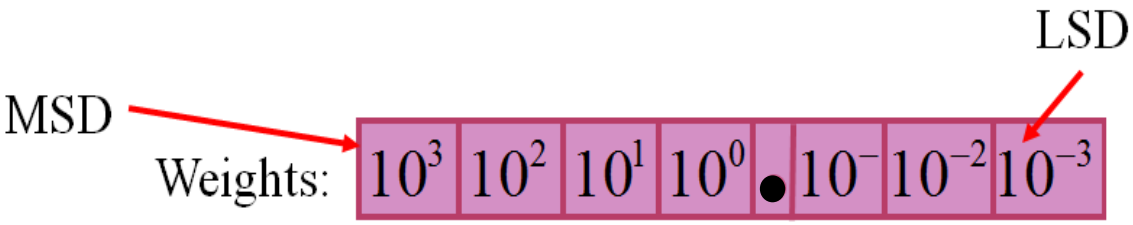
کامپایلر برنامه نوشته در یک زبان سطح بالا را به برنامه مقصد تبدیل می‌کند.



جلسه دوم: آشنایی با سیستم اعداد

مرور سیستم دهدهی

- پایه ۱۰ است و ارقام ۰، ۱، ... ۹ می باشند.
- برای اعداد بزرگتر از ۹، یک رقم با اهمیت تر به سمت چپ اضافه کنید. مثلا: ۱۹>۹
- هر محل دارای یک وزن است:

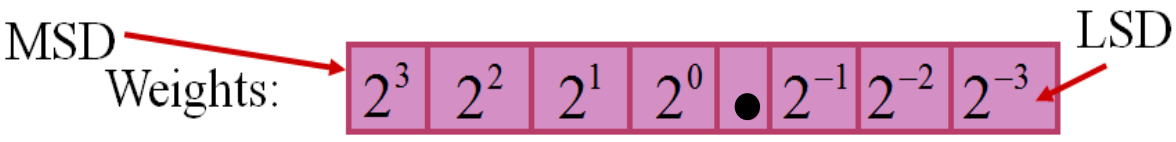


- به عنوان مثال عدد ۱۹۳۶,۲۵ را می توان به صورت زیر نمایش داد:

$$1\times10^3+9\times10^2+3\times10^1+6\times10^0+2\times10^{-1}+5\times10^{-2}$$

سیستم عدد نویسی دودویی

- پایه ۲ است و ارقام ۰، ۱هستند.
- برای اعداد بزرگتر از ۱، یک رقم با اهمیت تر به سمت چپ اضافه کنید. مثلا: ۱۰>۱
- هر محل دارای یک وزن است:



- به عنوان مثال عدد ۱۰۱۱۱,۰۱ را می توان به صورت زیر محاسبه کرد:

$$1\times2^4+0\times2^3+1\times2^2+1\times2^1+1\times2^0+0\times2^{-1}+1\times2^{-2}=$$

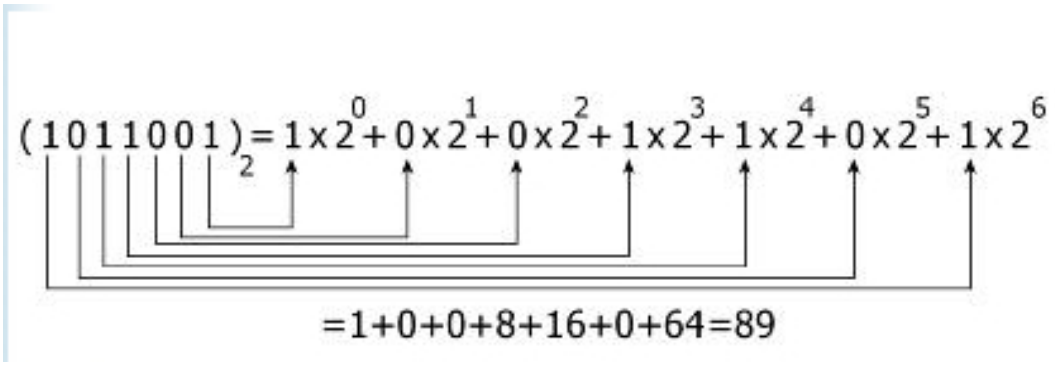
$$=1\times16+0\times8+1\times4+1\times2+1\times1+0\times0.5+1\times0.25=23.25$$

مبنای دو (باینری)

ارزش مکانی در مبنای 10 توانهای 10 است (یکان، دهگان، صدگان،...) و ارزش مکانی رقم ها (بیتها) در مبنای 2 توانهای 2 است.

$2^0 = 1$ $2^1 = 2$ $2^2 = 4$ $2^3 = 8$ $2^4 = 16$

$2^5 = 32$ $2^6 = 64$ $2^8 = 256$ $2^7 = 128$ $2^9 = 512$ $2^{10} = 1024$



سیستم عدد نویسی دودویی

$(110000.0111)_2 = (\ ?\ )_{10}$

جواب: ۴۸,۴۳۷۵

در دنیای کامپیوتر:

- $۲^{۱۰}$=۱۰۲۴ با K (کیلو) نشان داده می شود.
- $۲^{۲۰}$=۱۰۴۸۵۷۶ با M (مگا) نشان داده می شود.
- $G = ۲^{۳۰}$ (گیگا)
- $T = ۲^{۴۰}$ (ترا)

چه تعداد بیت در یک حافظه 16GByte وجود دارد؟

مبناهای ۸ و ۱۶

- **مبنای ۸**
  - پایه ۸ است و رقم ها 0, 1, 2, 3, 4, 5, 6, 7 هستند
  - $(236.4)_8 = (158.5)_{10}$
  - $2\times8^2+3\times8^1+6\times8^0+4\times8^{-1}=158.5$
- **مبنای ۱۶**
  - پایه ۱۶ است و رقمهای 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 از سیستم دهدیی قرض گرفته شده اند و از A, B, C, D, E, F به ترتیب برای نمایش رقمهای ۱۰، ۱۱، ۱۲، ۱۳، ۱۴، ۱۵ استفاده می گردد.
  - $(D63FA)_{16} = (877562)_{10}$

$$13\times16^4+6\times16^3+3\times16^2+15\times16^1+10\times16^0=877562$$

تبدیل از دهدهی به دودویی

تبدیل اعداد اعشاری:

معادل دودویی $(0.8542)_{10}$ را تا شش رقم دقت پیدا کنید.

$0.8542 \times 2 = 1 + 0.7084 \quad a_{-1} = 1$

$0.7084 \times 2 = 1 + 0.4168 \quad a_{-2} = 1$

$0.4168 \times 2 = 0 + 0.8336 \quad a_{-3} = 0$

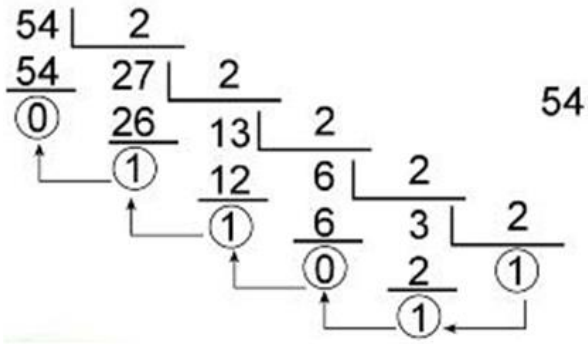
$0.8336 \times 2 = 1 + 0.6672 \quad a_{-4} = 1$

$0.6672 \times 2 = 1 + 0.3344 \quad a_{-5} = 1$

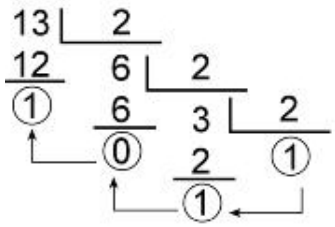
$0.3344 \times 2 = 0 + 0.6688 \quad a_{-6} = 0$

$(53.8542)_{10} = (\ ?\ )_2$

تبدیل از مبنای ده به مبنای دو به صورت زیر است:



$54=(110110)_2$



$13=(1101)_2$

جمع دودویی

$\begin{array}{r} 1 \\ 1 \\ \hline 10 \end{array}$ +	$\begin{array}{r} 1 \\ 0 \\ \hline 1 \end{array}$ +	$\begin{array}{r} 0 \\ 1 \\ \hline 1 \end{array}$ +	$\begin{array}{r} 0 \\ 0 \\ \hline 0 \end{array}$ +

در جمع آخر، 1، رقمی نقلی است که با بیتهای بعدی جمع می شود.

مثال : جمع زیر را در مبنای 2 انجام دهید.

$29=(00011101)_2$

$17=(00010001)_2$

$\begin{array}{r} 29 \\ 17 \\ \hline ? \end{array}$ +

$\begin{array}{r} \overset{1}{0}\overset{1}{0}011101 \\ 00010001 \\ \hline (00101110)_2 \end{array}$ +

امتحان نتیجه $(00101110)_2 = 2+4+8+32 = 46$

جلسه سوم: آشنایی با الگوریتم

هدف های کلی

- شناخت حل مسئله و ارائه الگوریتم
- شناخت اجزاء لازم برای حل مسئله
- بررسی صحت الگوریتم

هدف های رفتاری

دانشجو پس از مطالعه این فصل باید بتواند:

- الگوریتمی را برای حل مسئله ارائه دهد.
- الگوریتم های مختلف برای یک مسئله را مقایسه کند.
- شرط ها و حلقه ها را در الگوریتم بکار ببرد .

در زندگی روزمره، انسان با مسائل مختلفی روبروست و برای هر کدام از این مسائل (حل مشکلات) راه حلی و روشی را بر می‌گزیند. مسائلی از قبیل راه رفتن، غذا خوردن، خوابیدن و غیره که بشر تقریباً هر روز آنها را پیش روی خود دارد.

همه این مسائل نیاز به روشی برای حل کردن دارند مثلا راه رفتن باید با ترتیب خاصی و مراحل معینی انجام شود. تا مسئله راه رفتن برای بشر حل شود. اصطلاحاً روش انجام کار یا حل مسئله را الگوریتم آن مسئله می‌نامند.

تعریف الگوریتم

هر دستورالعملی که مراحل انجام کاری را با زبانی دقیق و با جزئیات کافی بیان نماید بطوریکه ترتیب مراحل و شرط خاتمه عملیات در آن کاملا" مشخص شده باشد را الگوریتم گویند. به عبارتی دیگر: الگوریتم مجموعه‌ای از دستورالعمل ها، برای حل مسئله می‌باشد که شرایط زیر را باید دارا باشد:

- دقیق باشد
- جزئیات کامل حل مسئله را داشته باشد.
- **پایان‌پذیر باشد.**

مراحل الگوریتم

برای حل یک مسئله باید الگوریتم آن مسئله را مشخص کنیم (یا بیابیم). که اصطلاحاً طراحی الگوریتم برای آن مسئله نامیده می‌شود. در طراحی الگوریتم معمولاً سه مرحله زیر را از هم جدا می‌کنند:

- **خواندن داده‌ها**
- **انجام محاسبات**
- **خروجی ها**

مثال : الگوریتمی بنویسید که دو عدد از ورودی دریافت کرده مجموع دو عدد را محاسبه و چاپ نماید.

خروجی‌ها	انجام محاسبات	ورودی ها
مجموع دو عدد	جمع دو عدد	a , b

0ـ شروع
۱ـ a, b را بخوان.
۲ـ مجموع a , b را محاسبه و در sum قرار بده.
۳ـ sum را در خروجی چاپ کن
۴ـ پایان

مثال: الگوریتمی بنویسید که سه عدد از ورودی دریافت کرده مجموع و میانگین **سه** عدد **را** محاسبه **و** چاپ کند.

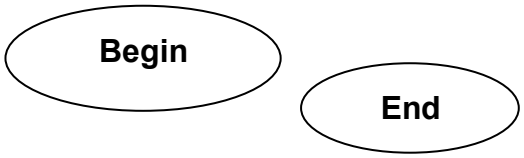
خروجی‌ها	انجام محاسبات	ورودی ها
چاپ مجموع	محاسبه مجموع	a , b , c
چاپ میانگین	محاسبه میانگین	

0ـ شروع
۱ـ سه عدد از ورودی بخوان
۲ـ مجموع سه عدد را محاسبه و در sum قرار بده.
۳ـ sum را بر سه تقسیم کرده،در ave قرار بده.
۴ـ ave , sum را در خروجی چاپ کن.
۵ـ پایان.

فلوچارت

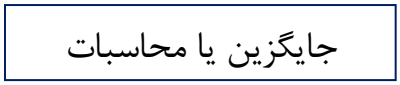
معمولاً درک یک الگوریتم با شکل راحت تر از نوشتن آن بصورت متن می‌باشد. لذا الگوریتم را با فلوچارت (flowchart) نمایش می‌دهند. فلوچارت از شکل‌های زیر تشکیل می‌شود.

۱. علامت‌های شروع و پایان: که معمولا از یک بیضی استفاده می‌کنند:



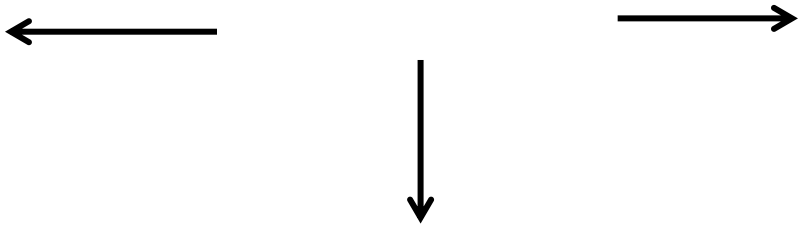
۲. علامت های ورودی و خروجی: که معمولا از متوازی‌الاضلاع استفاده می‌شود:

۳. علامت های محاسباتی و جایگزینی: برای نمایش دستورات جایگزینی و محاسباتی از مستطیل استفاده می‌کنند:

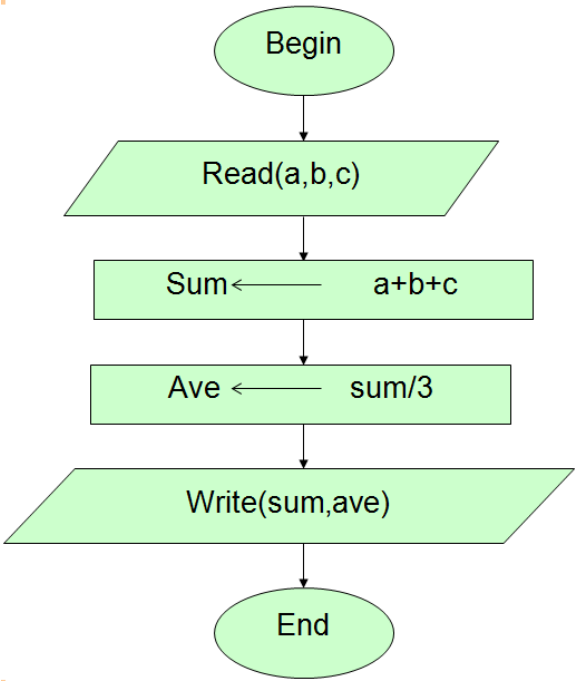


۴. علامت شرط: برای نمایش شرط از لوزی استفاده می‌شود.

۵. علامت اتصال: برای اتصال شکل‌های مختلف بهم از فلش‌های جهت‌دار استفاده می‌کنند.

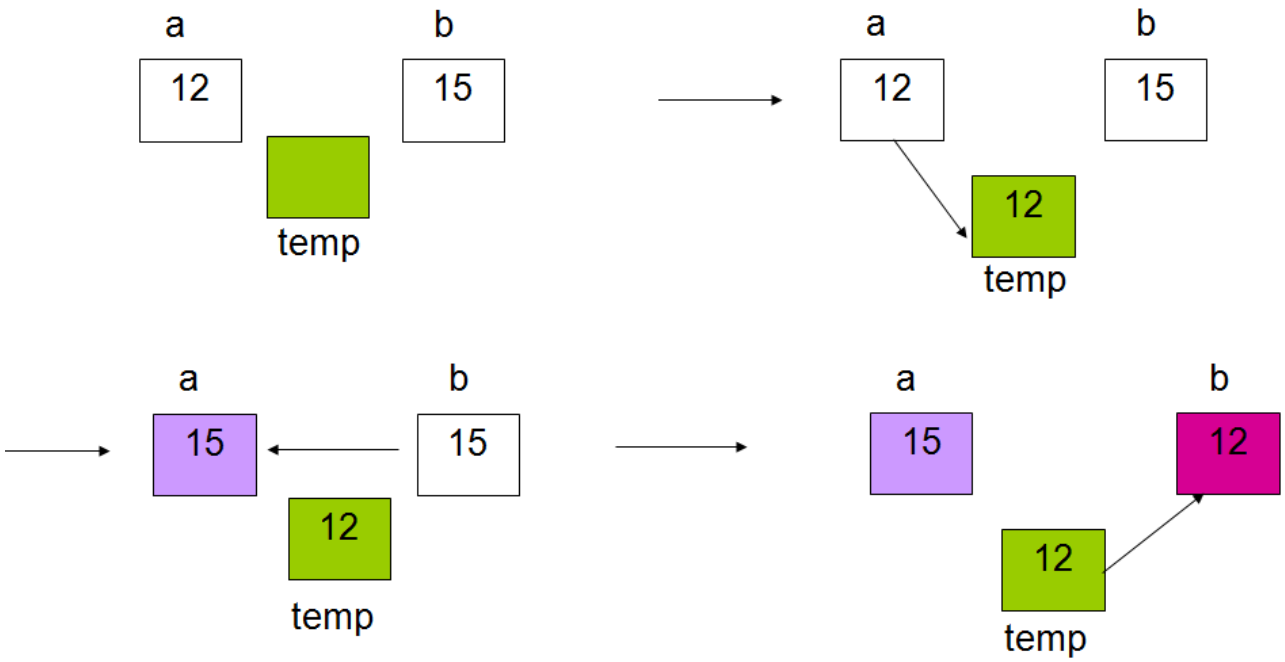


مثال» فلوچارت مجموع سه عدد

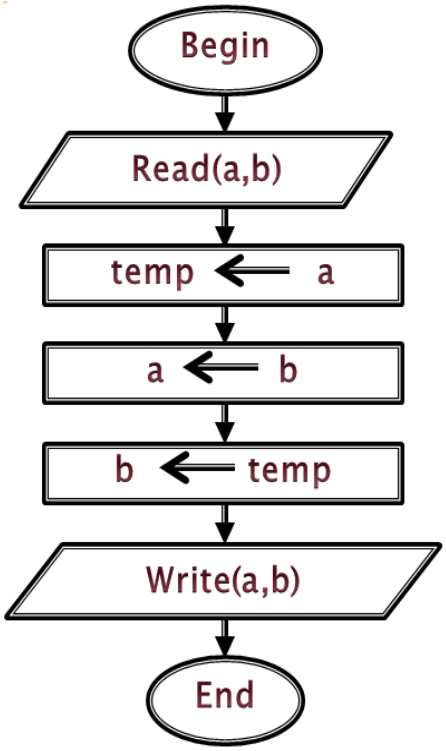


مثال: فلوچارتی رسم نمائید که دو عدد از ورودی دریافت کرده سپس محتویات دو عدد را با هم جابجا نماید.

برای حل این مسئله a , b را دو متغیر که در آن ها دو عدد خوانده شده، قرار می‌گیرد در نظر می‌گیریم. سپس با استفاده از یک متغیر کمکی محتویات این دو عدد را جابجا می‌کنیم :



فلوچارت مسئله بالا بصورت زیر خواهد بود:



تمرین

۱ـ فلوچارتی رسم نمائید که طول و عرض مستطیل را از ورودی دریافت کرده محیط و مساحت آنرا محاسبه و چاپ کند.

۲ـ فلوچارتی رسم نمائید که شعاع دایره‌ای را از ورودی دریافت کرده، محیط و مساحت آنرا محاسبه و چاپ نماید.

۳ـ فلوچارتی رسم کنید که سه عدد Third , second, first را از ورودی دریافت کرده، محتویات آن ها را جابجا نموده، حاصل را در خروجی چاپ کند.

۴ـ فلوچارتی رسم نمائید که دو عدد از ورودی دریافت کرده، سپس محتویات دو عدد را بدون استفاده از متغیر کمکی جابجا کند.

۵ـ فلوچارتی رسم نمائید که عددی (درجه حرارت برحسب سانتیگراد) را از ورودی دریافت کرده سپس آن را به درجه فارنهایت تبدیل کند.

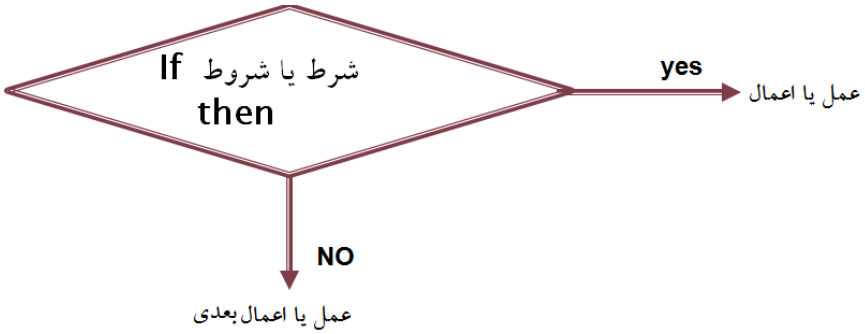
$$f = \frac{9}{5}\, c + 32$$

دستورالعمل‌های شرطی

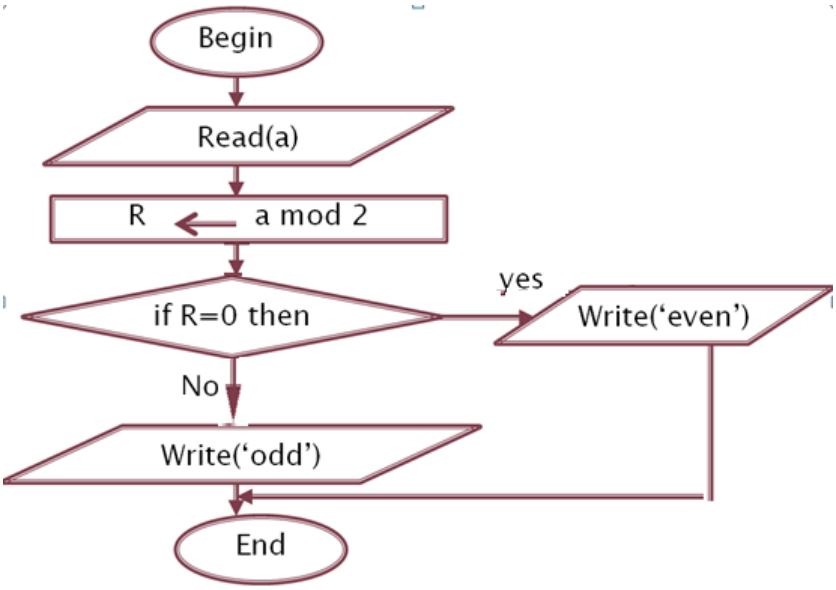
در حل بسیاری از مسائل یا تقریباً تمام مسائل نیاز به استفاده از شروط جزء، نیازهای اساسی محسوب می‌شود. همانطور که ما خودمان در زندگی روزمره با این شرط‌ها سرکار داریم. بطور مثال اگر هوا ابری باشد ممکن است چنین سخن بگوییم:

اگر هوا بارانی باشد سپس چتری برمی‌دارم. در غیر اینصورت چتر برنمی‌دارم.

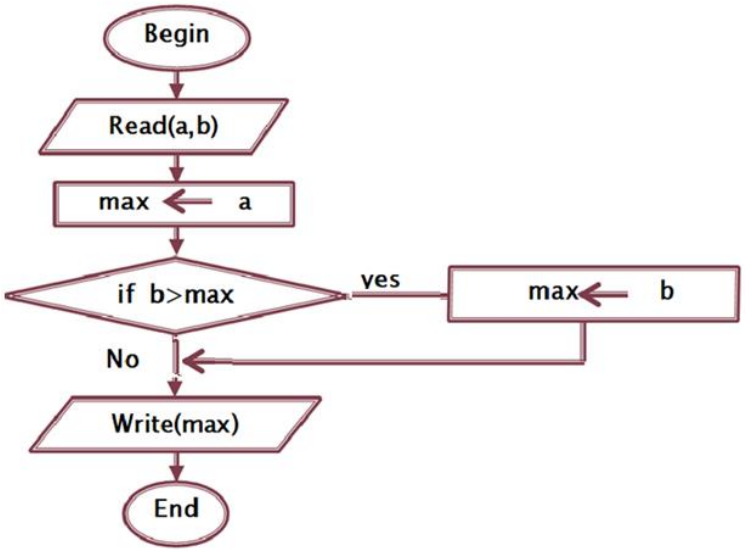
در حالت کلی شرط را بصورت زیر نمایش می‌دهند:



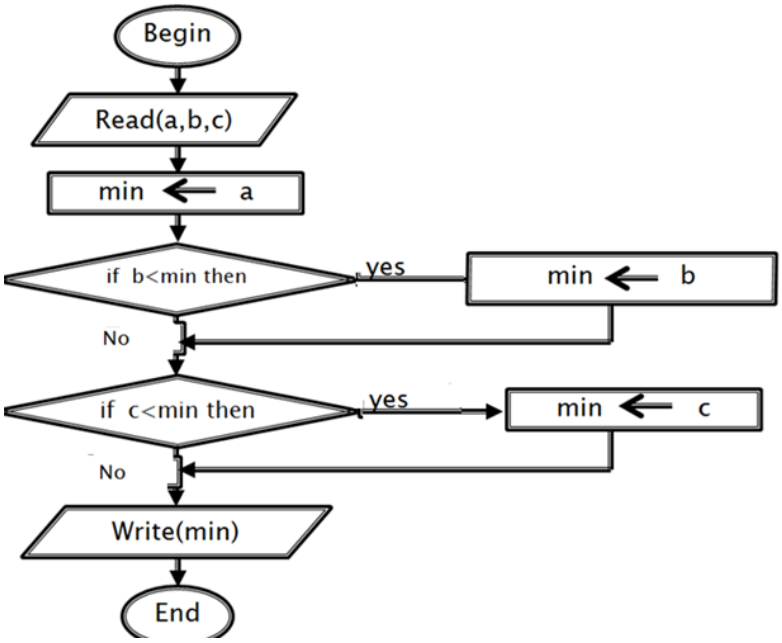
مثال : فلوچارتی رسم نمائید که عددی را از ورودی دریافت کرده، فرد یا زوج بودن آن را تشخیص دهد.



مثال : فلوچارتی رسم کنید که دو عدد از ورودی دریافت کرده بزرگترین عدد را پیدا کرده در خروجی چاپ نماید.



مثال : فلوچارتی رسم نمائید که سه عدد از ورودی دریافت کرده، کوچکترین عدد را یافته در خروجی چاپ نماید:



نمونه اجرای فلوچارت بالا **بصورت** زیر می‌باشد:

	a	b	c	Min	خروج
1	12	11	17		11
2				12	
3				11	
4				11	
5				11	

تمرین

۱- فلوچارتی رسم کنید که عددی را از ورودی دریافت کرده، قدر مطلق عدد را در خروجی چاپ کند.

۲- فلوچارتی رسم نمائید که عددی از ورودی دریافت کرده مثبت، منفی یا صفر بودن عدد را تشخیص داده، در خروجی با پیغام مناسب چاپ کند.

۳- فلوچارتی رسم نمائید که عددی را از ورودی دریافت کرده، بخش پذیری آن بر ۳ و ۵ را بررسی نماید.

۴ـ فلوچارتی رسم نمائید که ضرایب یک معادله درجه دوم را از ورودی دریافت کرده، ریشه‌های آن را محاسبه در خروجی چاپ کند.

حلقه‌ها

در حل بسیاری از مسائل با عملیاتی روبرو می‌شویم ، که نیاز به تکرار دارند و عمل تکرار آنها به تعداد مشخصی انجام می‌گیرد. فرض کنید، بخواهیم میانگین ۱۰۰ عدد را محاسبه کنیم، در اینصورت منطقی بنظر نمی‌رسد که ۱۰۰ متغیر مختلف را از ورودی دریافت کنیم سپس آنها را جمع کنیم.

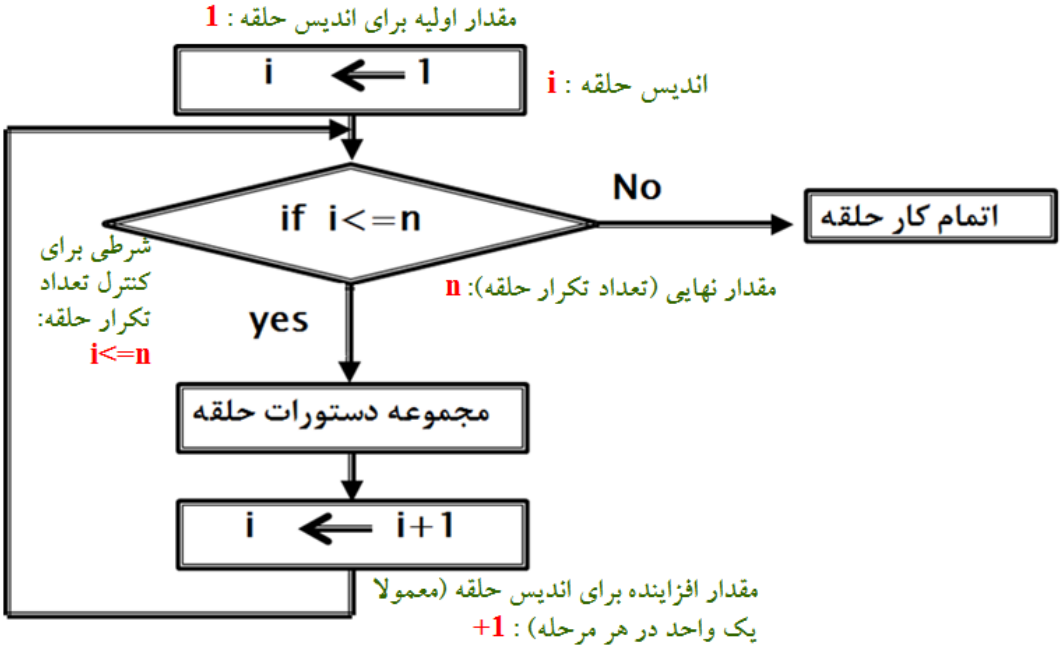
انواع حلقه ها

۱. حلقه های با تکرار مشخص
۲. حلقه های با تکرار نا مشخص

حلقه های با تکرار مشخص: در این نوع حلقه‌ها تعداد تکرار مشخص می‌باشد این حلقه از اجزاء زیر تشکیل می‌شود:

۱ـ اندیس حلقه
۲ـ مقدار اولیه برای اندیس حلقه
۳- مقدار افزاینده یا کاهنده برای اندیس حلقه (معمولا یک واحد در هر مرحله)
۴ـ مقدار نهایی (تعداد تکرا حلقه)
۵ـ شرطی برای کنترل تعداد تکرار حلقه

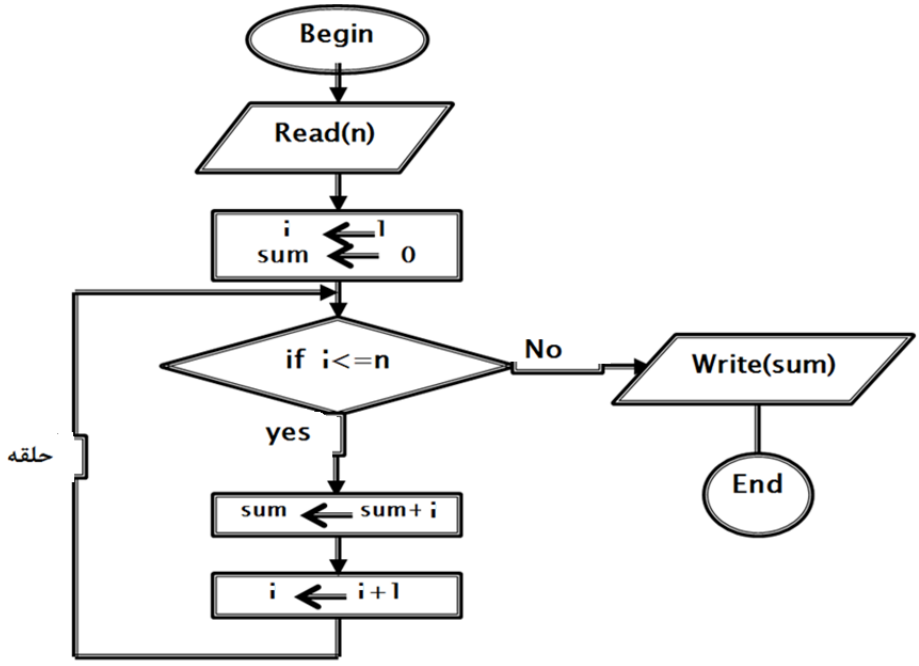
این حلقه‌ها را غالباً با فلوچارت بصورت زیر نمایش می‌دهند:



مثال : فلوچارتی رسم نمائید که عدد n را از ورودی دریافت کرده، مجموع اعداد از یک تا n را محاسبه کند.

n مقدار نهایی

i اندیس حلقه



نمونه اجرای فلوچارت بالا بصورت زیر است:

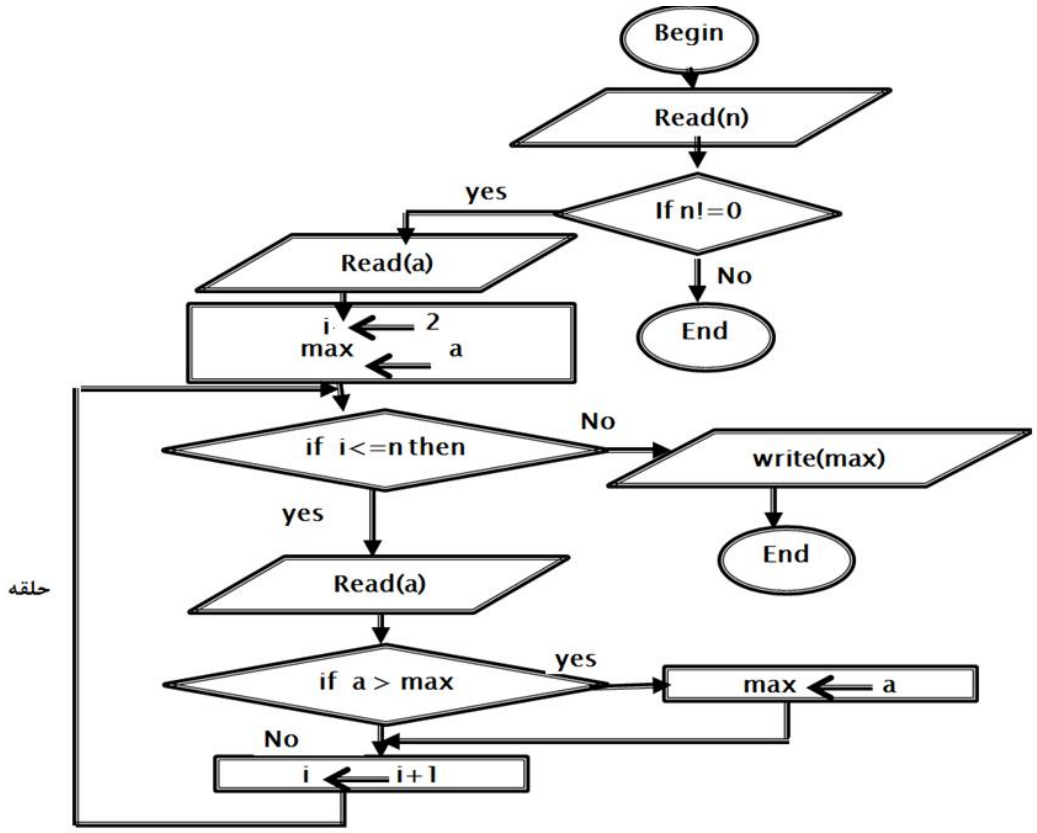
	N	I	sum	خروجی
1	5	1	0	15
2		2	1	
3		3	3	
4		4	6	
5		5	10	
6		6	15	
7				

مثال : فلوچارتی رسم کنید که n عدد از ورودی دریافت کرده، بزرگترین مقدار از بین n عدد را پیدا کرده در خروجی چاپ نماید.

اندیس حلقه i

مقدار نهایی n

بزرگترین مقدار Max



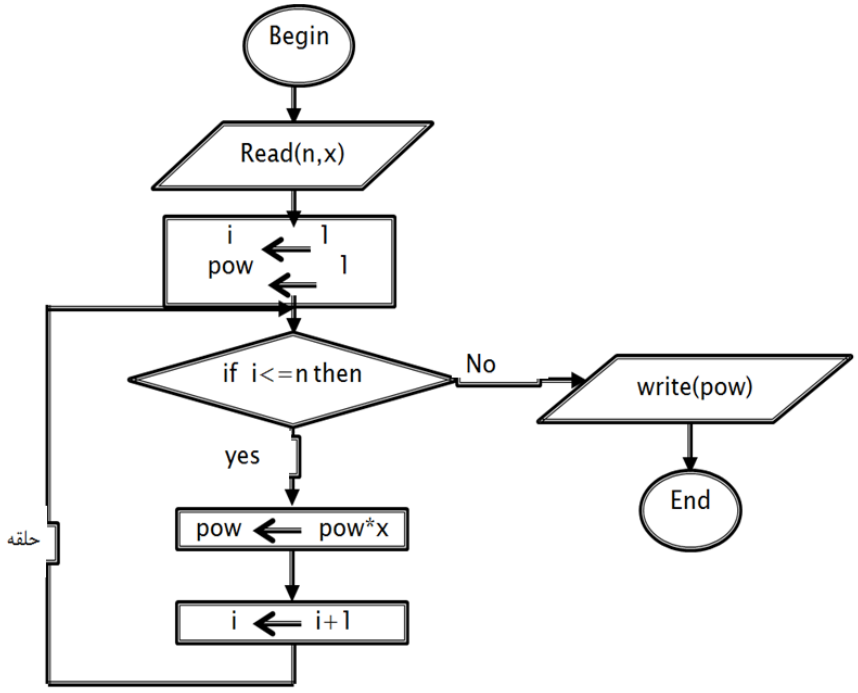
مثال : فلوچارتی رسم نمائید که x , n، دو عدد صحیح مثبت را از ورودی دریافت کرده سپس x به توان n را محاسبه کند.

اندیس حلقه i

مقدار نهایی n

عدد به توان n

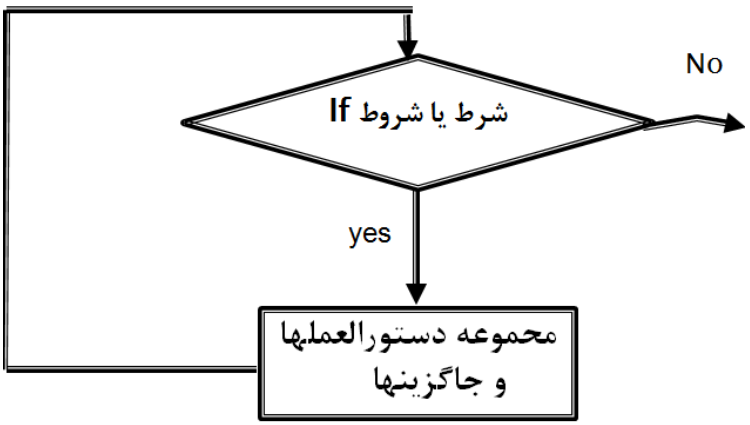
pow



حلقه‌هایی که تعداد تکرار آن ها مشخص نیست (در C++ به حلقه while مشهورند.)

در این حلقه‌ها با توجه به ورودی، تعداد تکرار مشخص می‌شود. و دقیقاً نمی‌توان تعداد تکرار حلقه را بدون ورودی معین کرد. این حلقه ها فقط شامل شرطی هستند که تا زمانیکه برقرار باشد حلقه اجرا می‌شود.

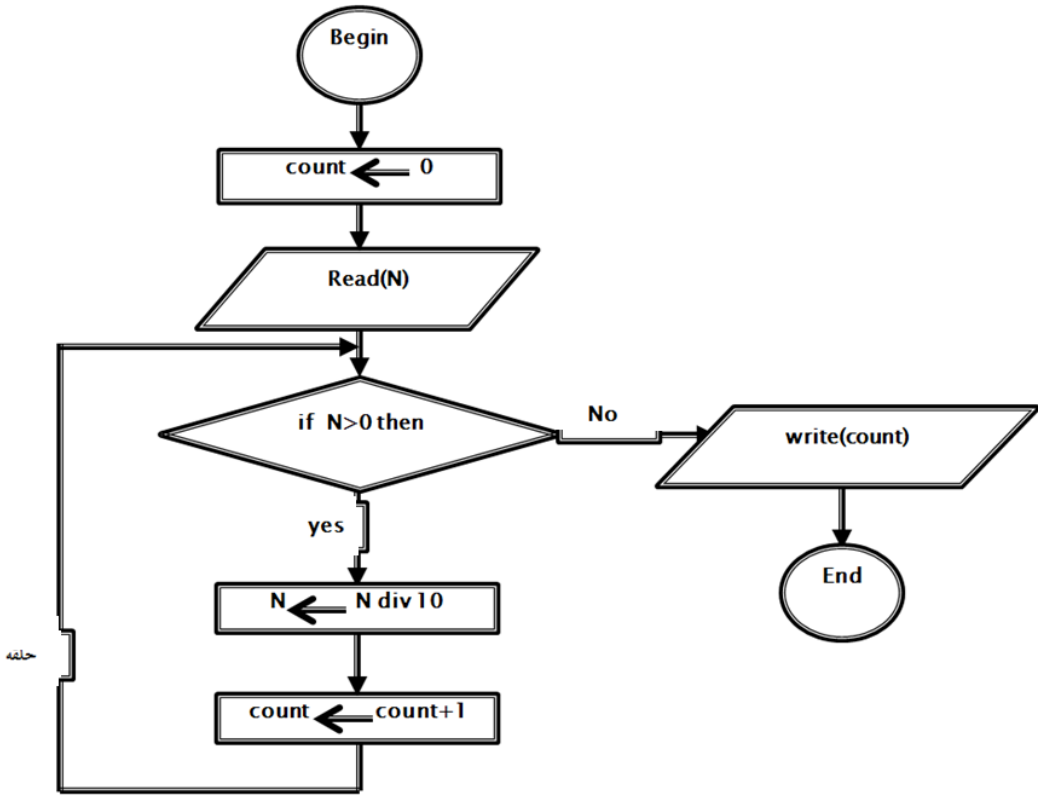
در حالت کلی این نوع حلقه‌ها بصورت زیر نمایش داده می‌شوند:



مثال: فلوچارتی رسم کنید که عددی را از ورودی دریافت کرده سپس تعداد ارقام آن را شمرده در خروجی چاپ نماید.

عدد خوانده شده *N*

تعداد ارقام *count*



مثال : فلوچارتی رسم نمائید که عددی از ورودی دریافت کرده، سری فیبوناچی قبل از آن را تولید نماید.

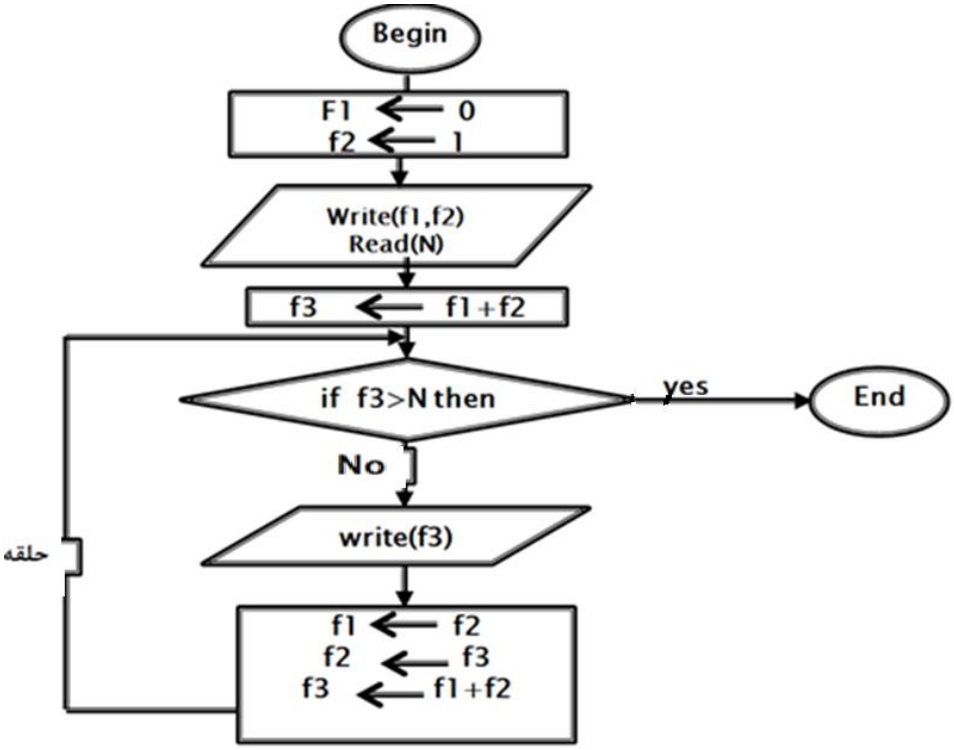
در حالت کلی جملات سری بصورت: $f_k=f_{k-1}+f_{k-2}$

عدد خوانده شده N

جمله اول سری f_1

جمله دوم سری f_2

جمله سوم سری f_3



تمرین

۱- فلوچارتی رسم نمائید که عددی از ورودی دریافت کرده، کامل بودن آن را بررسی نماید. (عدد کامل، عددی است که مجموع مقسوم‌علیه‌های آن با خودش برابر باشد.)

۲- فلوچارتی رسم کنید که N را از ورودی دریافت کرده، N جمله سری فیبوناچی را تولید نماید.

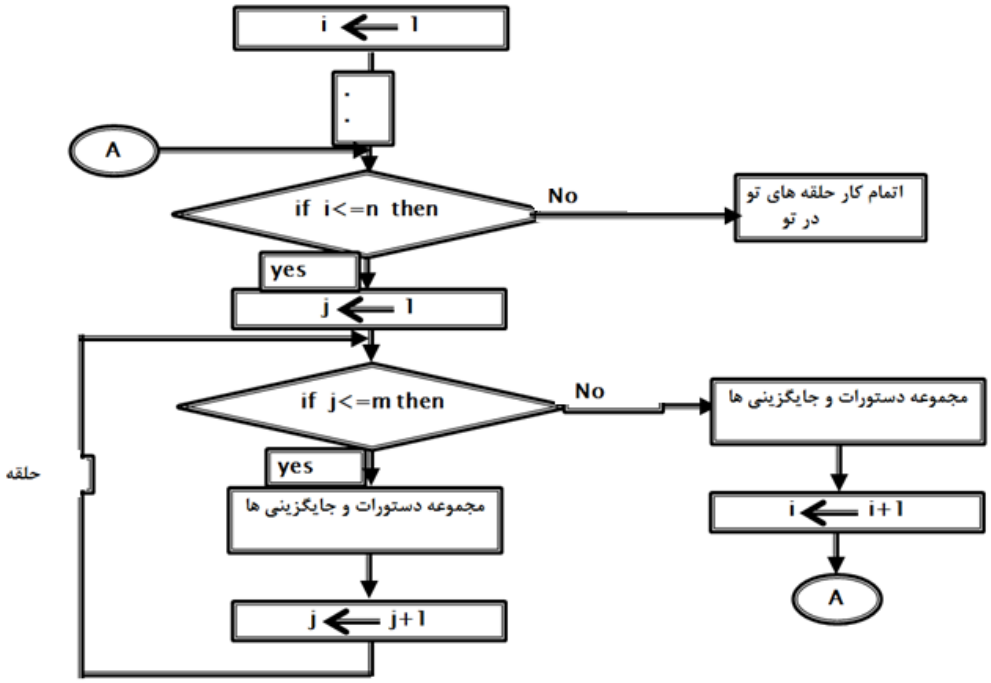
۳ـ فلوچارتی رسم نمائید که دو عدد N , M را از ورودی خوانده، بزرگترین مقسوم‌علیه مشترک دو عدد را محاسبه و چاپ کند.

حلقه‌های تودرتو

الگوریتم‌هایی که تا حال بکار بردیم، فقط شامل یک حلقه بودند. در صورتی که در بسیاری از مسائل ممکن است نیاز به استفاده از چند حلقه در داخل هم باشیم. در این نوع حلقه‌ها باید دقت بیشتری به خرج دهیم، تا مشکلی پیش نیاید. اگر از حلقه‌های نوع اول بصورت تودرتو استفاده کنیم در اینصورت برای هر حلقه شرط نهایی و اندیس اولیه جداگانه باید تعریف کنیم .

در حلقه‌های تودرتو به ازای یکبار تکرار حلقه اولیه، حلقه داخلی به اندازه مقدار نهایی خود تکرار می‌شود. در کل اگر حلقه اولیه n بار تکرار شود و حلقه داخلی m بار، در اینصورت کل حلقه n×m بار تکرار خواهد شد.

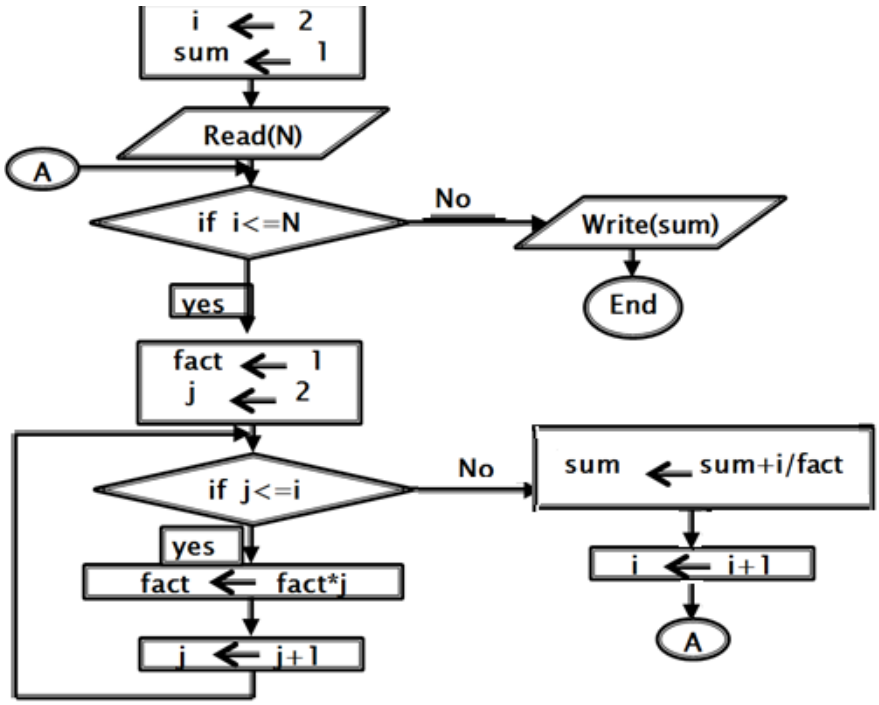
فلوچارت حلقه‌های **تودرتو** را می‌توان بصورت روبرو نشان داد:



مثال : فلوچارتی رسم نمائید که N را از ورودی دریافت کرده، مجموع سری زیر را محاسبه نماید:

$$S = 1 + \frac{2}{2!} + \frac{3}{3!} + ... + \frac{N}{N!}$$

اندیس حلقه اول	I
ورودی	N
محاسبه فاکتوریل	fact
اندیس حلقه داخلی	j
مجموع	Sum



تمرینات آخر فصل

۱- فلوچارتی رسم نمائید که N عدد از ورودی دریافت کرده تعداد اعداد اول و کامل را شمرده در خروجی چاپ نماید.

۲- فلوچارتی رسم نمائید که N , X را از ورودی خوانده مقدار سری زیر را محاسبه کند:

$$S = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + ... + \frac{x^n}{N!}$$

۳- فلوچارتی رسم نمائید که عددی را از ورودی دریافت کرده مقلوب عدد را محاسبه و در خروجی چاپ کند.

۴ـ فلوچارتی رسم کنید که تاریخ تولد شخصی را از ورودی خوانده، سن شخص را با تاریخ روز، محاسبه نموده در خروجی چاپ کند.

۵- فلوچارتی رسم نمائید که (m>n) N ,M را از ورودی دریافت کرده سری فیبوناچی بین N ,M را تولید کرده، در خروجی چاپ کند.

برنامه نویسی به زبان C++

فهرست مطالب:

فصل اول: مقدمات C++

- تاریخچه مختصر
- قانون نامگذاری شناسه ها
- متغیر ها
- اعلان متغیر
- تخصیص مقادیر به متغیر
- داده های از نوع کرکتر
- کرکتر های مخصوص
- رشته ها
- نمایش مقادیر داده ها
- دریافت مقادیر
- عملگر انتساب
- عملگر های محاسباتی
- عملگرهای افزایش و کاهش
- عملگر sizeof
- عملگرهای جایگزینی محاسباتی
- اولویت عملگرها
- توضیحات (Comments)
- توابع کتابخانه
- برنامه در C++

تاریخچه مختصر C++:

این زبان در اوائل دهه ۱۹۸۰ توسط Bjarne stroustrup در آزمایشگاه بل طراحی شده. این زبان عملاً توسعه یافته زبان برنامه نویسی C می باشد که امکان نوشتن برنامه‌های ساخت یافته شئ گرا را می‌دهد.

قانون نامگذاری شناسه‌ها:

۱) حروف کوچک و بزرگ در نامگذاری شناسه‌ها متفاوت می‌باشند.

بنابراین Xy ، XY ، xY ، xy چهار شناسه متفاوت از نظر C++ می‌باشد.

۲) در نامگذاری شناسه‌ها از حروف الفباء، ارقام وزیر خط (underscore) استفاده می‌شود و حداکثر طول شناسه ۳۱ می‌باشد و شناسه بایستی با یک رقم شروع نگردد.

۳) برای نامگذاری شناسه‌ها از کلمات کلیدی نبایستی استفاده نمود. در زیر بعضی از کلمات کلیدی داده شده است.

And	Sizeof	then	xor	Template
Float	False	Friend	While	continue
extern	Private	Switch	Default	Const
delete	typedef	if	this	Virtual

لیست کامل کلمات کلیدی

متغیرها:

متغیر، مکانی در حافظه اصلی کامپیوتر می‌باشد که در آنجا یک مقدار را می‌توان ذخیره و در برنامه از آن استفاده نمود. قانون نامگذاری متغیرها همان قانون نامگذاری شناسه‌ها می‌باشد.

انواع داده ها

نوع داده	حافظه لازم
int	**۴ بایت**
unsigned int	**۴بایت**
long int	**۴ بایت**
unsigned long int	**۴ بایت**
char	**۱بایت**
unsigned char	**۱ بایت**
float	**۴ بایت**
double	**۸ بایت**

اعلان متغیرها:

قبل از آن که در برنامه به متغیرها مقداری تخصیص داده شود و از آن ها استفاده گردد بایستی آن ها را در برنامه اعلان نمود.

چند مثال از اعلان متغیر ها :

برای اعلان متغیرx از نوع int : *int x;*

برای اعلان متغیرهای p و q را از نوع float که هر کدام چهار بایت از حافظه را اشغال می‌کنند:

float p , q ;

برای اعلان متغیر next از نوع کرکتر که می‌توان یکی از ۲۵۶ کرکتر را به آن تخصیص داد و یک بایت را اشغال می‌کند.

char next ;

تخصیص مقادیر به متغیرها

با استفاده از عملگر = می‌توان به متغیرها مقدار اولیه تخصیص نمود.

مثال :

- ✓ در دستورالعمل int x=26; ، x را از نوع int با مقدار اولیه 26 اعلان نموده .
- ✓ در دستورالعمل long int a=67000 , b=260; ، متغیرهای b و a را از نوع long int تعریف نموده با مقادیر بترتیب 260 و 67000 مقدار دهی کرده ایم.

داده‌های از نوع کرکتر

برای نمایش داده‌های از نوع char در حافظه کامپیوتر از جدول ASCII استفاده می‌شود. جدول اسکی به هر یک از ۲۵۶ کرکتر یک عدد منحصر بفرد بین ۰ تا ۲۵۵ تخصیص می‌دهد.

کرکترهای مخصوص

کامپایلر C++ بعضی از کرکترهای مخصوص که در برنامه می‌توان از آنها برای فرمت بندی استفاده کرد را تشخیص می‌دهد. تعدادی از این کرکترهای مخصوص به همراه کاربرد آنها در ادامه آورده شده است .

بعنوان مثال از کرکتر \a می‌توان برای ایجاد صدای beep استفاده نمود.

Char x = '\a' ;

\n	**Newline**	**\a**	**Beep sound**	**\0**	**Null character**
\t	**Tab**	**\"**	**Double quote**	**\?**	**Question mark**
\b	**Backspace**	**\'**	**Single quote**	****	**Back slash**

رشته‌ها

رشته یا string عبارتست از دنباله‌ای از کرکترها که بین " " قرار داده می‌شود. در حافظه کامپیوتر انتهای رشته‌ها بوسیله \0 ختم می‌گردد.

مثال:

- ✓ "BOOK STORE" یک رشته ده کرکتری می‌باشد که با توجه به کرکتر \0 که به انتهای آن در حافظه اضافه می شود جمعـــاً یازده بایت را اشغال می‌کند.

- ✓ دقت نمایید که "w" یک رشته می‌باشد که دو بایت از حافظه را اشغال می‌کند در حالیکه 'w' یک کرکتر می‌باشد که یک بایت از حافظه را اشغال می‌نماید.

نمایش مقادیر داده‌ها

برای نمایش داده‌ها بر روی صفحه مانتور از cout که بدنبال آن عملگر درج یعنی >> قید شده باشد استفاده می‌گردد. بایستی توجه داشت که دوکرکتر > پشت سر هم توسط C++ بصورت یک کرکتر تلقی می‌گردد.

مثال:

- ✓ برای نمایش پیغام good morning بر روی صفحه نمایش :

```
cout << "good morning";
```

- ✓ برای نمایش مقدار متغیر X بر روی صفحه نمایش :

```
cout << x ;
```

دریافت مقادیر متغیرها

به منظور دریافت مقادیر برای متغیرها در ضمن اجرای برنامه از صفحه کلید، از cin که بدنبال آن عملگر استخراج یعنی << قید شده باشد می‌توان استفاده نمود.

مثال:

```
int x;
cout << "Enter a number:" ;
cin >> x;
```

عملگر انتساب

عملگر انتساب = می‌باشد که باعث می‌گردد مقدار عبارت در طرف راست این عملگر ارزیابی شده و در متغیر طرف چپ آن قرار گیرد.

مثال:

```
x=a+b;
x=35 ;
x=y=z=26 ;
```

از عملگرهای انتساب چندگانه نیز می‌توان استفاده نمود. که مقدار سه متغیر z و y و x برابر با 26 می شود.

عملگرهای محاسباتی

در C++ پنج عملگر محاسباتی وجود دارد که عبارتند از :

+	**جمع**
-	**تفریق**
*	**ضرب**
/	**تقسیم**
٪	**باقیمانده**

این عملگرها دو تائی می‌باشند زیرا روی دو عملوند عمل می‌نمایند. از طرف دیگر عملگرهای + و – را می‌توان بعنوان عملگرهای یکتائی نیز در نظر گرفت.

✓ **مثال ۱ :**

در حالتی که هر دو عملوند عملگرهای ٪ ، / ، * ، + ، – از نوع صحیح باشد نتیجه عمل از نوع صحیح می‌باشد.

عبارت	نتیجه
5 + 2	7
5 * 2	10
5 – 2	3
5 % 2	1
5 / 2	2

✓ **مثال ۲ :**

در صورتیکه حداقل یکی از عملوندهای عملگرهای / ، * ، – ، + از نوع اعشاری باشد نتیجه عمل از نوع اعشاری می‌باشد.

عبارت	نتیجه
5.0 +2	7.0
5*2.0	10.0
5.0 / 2	2.5
5.0 -2	3.0
5.0 / 2.0	2.5

عملگرهای افزایش و کاهش

در C++ ، افزایش یک واحد به مقدار یک متغیر از نوع صحیح را افزایش و بطور مشابه کاهش یک واحد از مقدار یک متغیر از نوع صحیح را کاهش می‌نامند.

عملگر کاهش را با - - و عملگر افزایش را با ++ نمایش می‌دهند. چون عملگرهای ++ و - - فقط روی یک عملوند اثر دارند این دو عملگر نیز جزء عملگرهای یکتائی می‌باشند.

سه **دستور** العمل :

```
++x;
x++;
x=x+1;
```

معادل می‌باشند و بطریق مشابه سه دستورالعمل زیر نیز معادل می باشند.

```
- - y;
y=y-1;
y- - ;
```

از عملگرهای ++ و -- می‌توان بدو صورت پیشوندی و پسوندی استفاده نمود. در دستورالعمل‌های پیچیده عملگر پیشوندی قبل از انتساب ارزیابی می شود و عملگر پسوندی بعد از انتساب ارزیابی می‌شود.

مثال:

```
int x=5;
y=++x * 2;
```

پس از اجرای دستورالعمل های فوق : y=12

```
int  x=5;
y=x++ * 2;
```

پس از اجرای دستورالعمل های فوق: y=10

عملگر sizeof

Sizeof ازعملگرهای یکتائی می باشد و مشخص کننده تعداد بایت هائی است که یک نوع داده اشغال می‌کند.

مثال :

```
int   x;
cout << sizeof x ;                مقدار ۴ نمایش داده می‌شود.
cout << sizeof(float) ;           مقدار ۴ نمایش داده می شود.
```

عملگرهای جایگزینی محاسباتی

برای ساده‌تر نوشتن عبارت ها در C++ ، می‌توان از عملگرهای جایگزینی محاسبـــاتی استفاده نمود.

+= -= *= /= %=

اولویت عملگرها

ارزیابی مقدار یک عبارت ریاضی براساس جدول اولویت عملگرها انجام می‌گردد. در ذیل جدول اولویت عملگرها براساس بترتیب از بیشترین اولویت به کمترین اولویت داده شده است.

چپ به راست	پرانتزها	( )
راست به چپ	عملگرهای یکتایی	sizeof ++ -- + -
چپ به راست	عملگرهای ضرب و تقسیم و باقیمانده	% / *
چپ به راست	عملگرهای جمع و تفریق	– +
چپ به راست	عملگرهای درج و استخراج	>> <<
راست به چپ	عملگرهای جایگزینی و انتساب	%= /= *= -= += =

مثال ۱ :

(5+2) *(6+2*2)/2 = ?

با توجه به جدول اولویت عملگرها داریم که

```
7 *(6+2*2)/2
7*(6+4)/2
7* 10 /2
70 /2
35
```

مثال ۲ :

```
int a=6 , b=2, c=8, d=12;
d=a++ * b/c ++;
cout << d << c << b << a;
```

خروجی: 7 2 9 1

توضیحات (Comments)

توضیحات در برنامه باعث خوانائی بیشتر و درک بهتر برنامه می شود. بنابراین توصیه بر آن است که حتی الامکان در برنامه‌ها از توضیحات استفاده نمائیم. در C++، توضیحات بدو صورت انجام می‌گیرد که در اسلایدهای بعد به آن اشاره شده است.

الف: این نوع توضیح **بوسیله** // انجام می‌شود. که کامپیوتر هر چیزی را که بعد از // قرار داده شود تا انتهای آن خط اغماض می‌نماید.

مثال :

```
c=a+b; //c is equal to sum of a  and b
```

ب: توضیح نوع دوم با */ شروع شده و به /* ختم می‌شود و هر چیزی که بین* / و /* قرار گیرد اغماض می‌نماید .

مثال :

```
/ * this is a program
to calculate sum of
n integer numbers */
```

توابع کتابخانه

زبان C++ مجهز به تعدادی توابع کتابخانه می‌باشد. بعنوان مثال تعدادی توابع کتابخانه برای عملیات ورودی و خروجی وجود دارند. معمولاً توابع کتابخانه مشابه ، بصورت برنامه‌های هدف (برنامه ترجمه شده بزبان ماشین) در قالب فایل های کتابخانه دسته بندی و مورد استفاده قرار می‌گیرند. این فایل ها را فایلهای header می‌نامند و دارای پسوند h. می‌باشند.

نحوه استفاده از توابع کتابخانه ای

برای استفاده از توابع کتابخانه خاصی بایستـــی نام فایل header آنرا در ابتدای برنامه در دستور #include قرار دهیم.

#include < header **اسم فایل** >

تابع	نوع	شرح	فایل هدر
abs(i)	*int*	قدرمطلق i	stdlib.h
cos(d)	*double*	کسینوس d	math.h
exp(d)	*double*	e^x	math.h
log(d)	*double*	$\log_e$ d	math.h
log10(d)	*double*	Log_{10} d	math.h
sin(d)	*double*	سینوس d	math.h
sqrt(d)	*double*	جذر d	math.h
strlen(s)	*int*	تعداد کرکترهای رشته s	string.h
tan(d)	*double*	تانژانت d	math.h
toascii(c)	*int*	کداسکی کرکتر c	stdlib.h
tolower(c)	*int*	تبدیل به حروف کوچک	stdlib.h
toupper(c)	*int*	تبدیل به حرف بزرگ	stdlib.h

برنامه در C++

اکنون باتوجه به مطالب گفته شده قادر خواهیم بود که تعدادی برنامه ساده و کوچک به زبان C++ بنویسیم. برای نوشتن برنامه بایستی دستورالعمل ها را در تابع ( )main قرار دهیم و برای اینکار می‌توان به یکی از دو طریقی که در ادامه آمده است، عمل نمود.

روش اول :

```
using namespace std;
#include    <    >
int main( )
 {
دستورالعمل ۱ ;
دستورالعمل ۲ ;
.
.
.
دستورالعمل n;
return 0 ;
}
```

روش دوم:

```
using namespace std;
#include    <    >
void main( )
{
دستورالعمل ۱ ;
دستورالعمل ۲ ;
.
.
.
دستورالعمل n;
}
```

نکات:

به خطاهای برنامه نویسی error می گویند. ما در برنامه نویسی دو نوع خطا داریم:

- خطاهای دستوری (syntax error)
- خطاهای منطقی (logical error).

در خطاهای دستوری (syntax error)، کد برنامه دارای اشکال است که معمولا خود کامپایلر با پیغامی خطا در زمان کامپایل، آن خطا را به ما تذکر می‌دهد و تا زمانی که آن خطا را برطرف نکنیم برنامه اجرا نمی‌شود. (به این خطا، خطای زمان کامپایل نیز می‌گویند.) به عنوان مثال فرض کنید دستور cout را به صورت coot بنویسید، در این صورت کامپایلر با پیغامی، این خطا را به شما هشدار می‌دهد. اما در خطای منطقی (logical error)، کد برنامه مشکلی ندارد، ولی الگوریتم برنامه دارای مشکل است، رایج‌ترین خطای منطقی، خطای تقسیم بر صفر است. (زیرا اگر هر عدد بر صفر تقسیم شود، حاصلی مبهم دارد) این نوع خطاها توسط کامپایلر تشخیص داده نمی‌شود، و حتی ممکن است برنامه اجرا شود، ولی پس از اجرا، برنامه نتایجی به همراه دارد که با پیش بینی ما از برنامه متفاوت خواهد بود و حتی ممکن است در خروجی برنامه مشکل ایجاد کند.

در این قسمت به مفاهیم اولیه کدنویسی به زبان C++ می‌پردازیم.

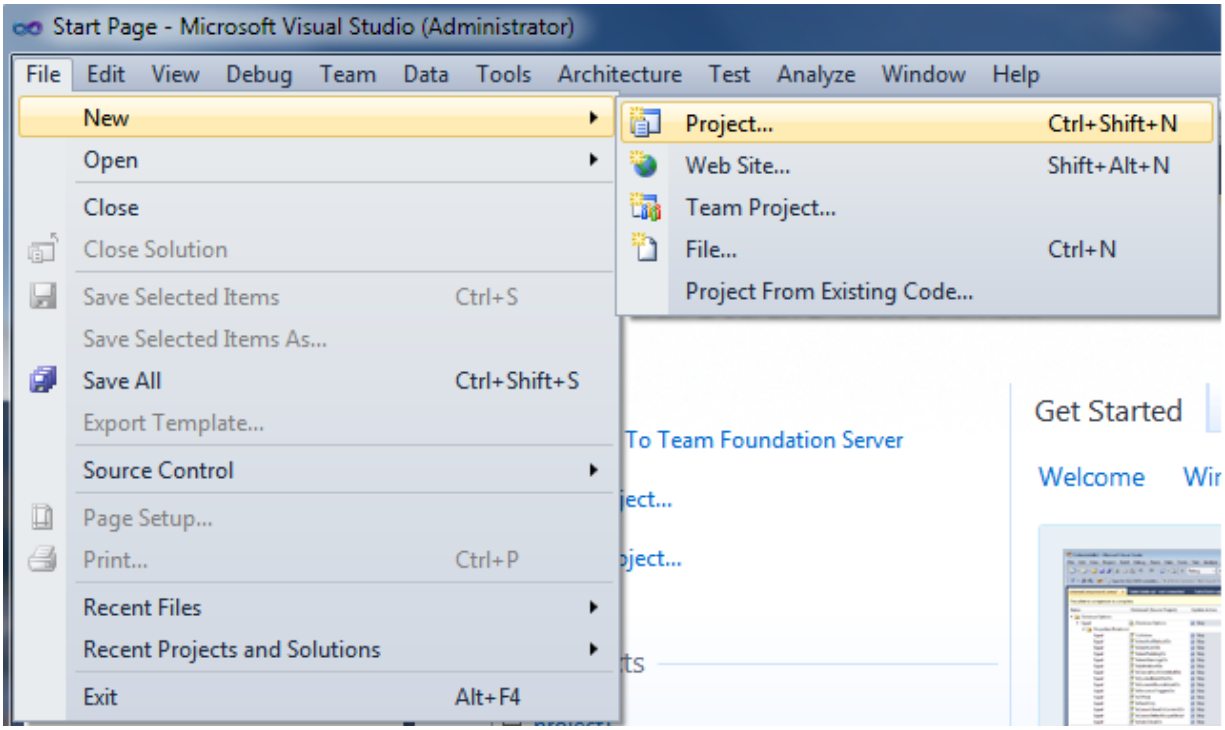
ما برای کدنویسی برنامه‌هایمان، به نرم افزارهای دیگری نیاز داریم تا کدهای برنامه را درون آن بنویسیم و نتایج آن‌ها را برایمان نمایش بدهد، به این برنامه‌ها مترجم (Compiler) گفته می‌شود، که عبارتند از:

Microsoft visual studio, Borland C++, Turbo C++, Notepad ++

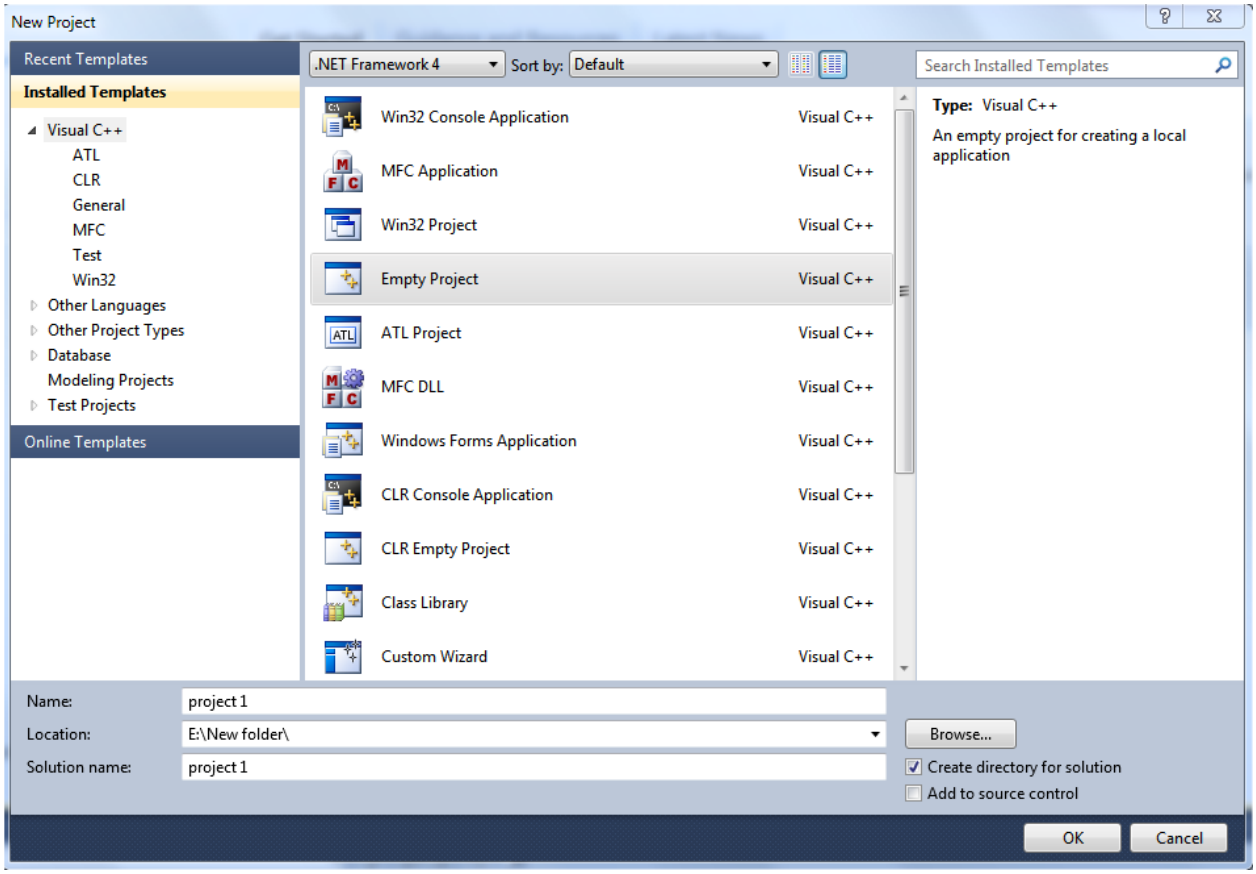
که من استفاده از Microsoft visual studio 2010 را پیشنهاد می کنم، زیرا نوشتن و ویرایش کد در این محیط راحت‌تر است.

ابتدا به روش ساختن یک پروژه جدید در Visual Studio 2010 می‌پردازیم:

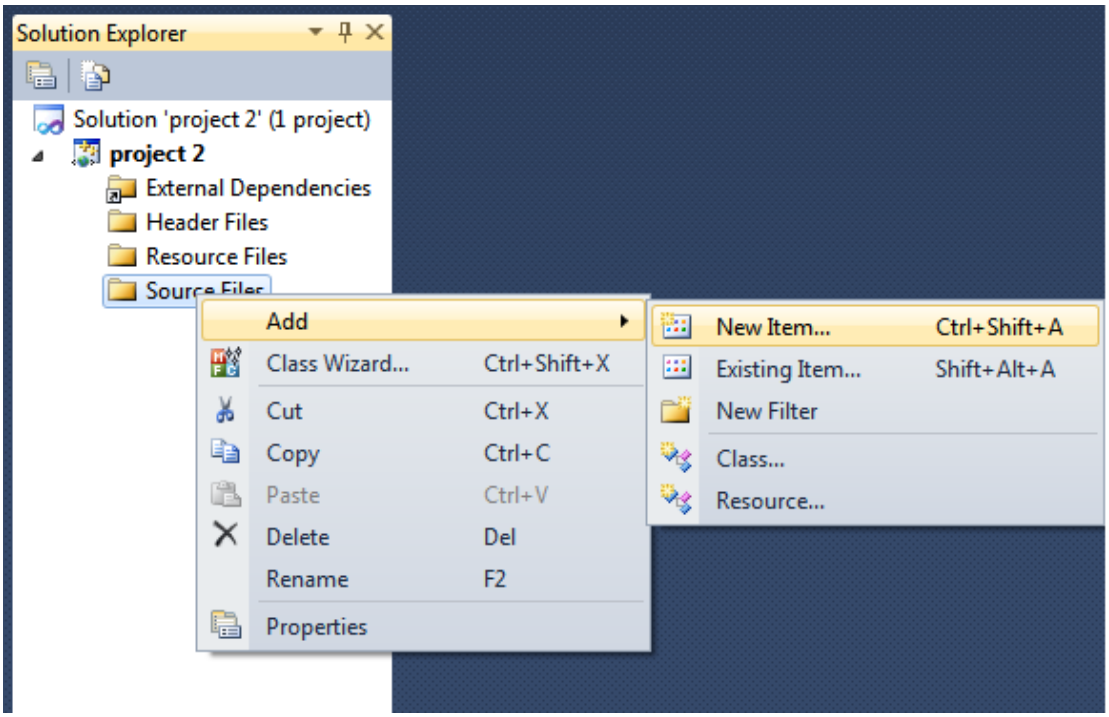
پس از باز کردن نرم افزار Visual Studio 2010 همانند شکل زیر بر روی گزینه File کلیک کرده، سپس با رفتن به روی دکمه‌ی New، گزینه‌ی Project را انتخاب می‌کنیم:



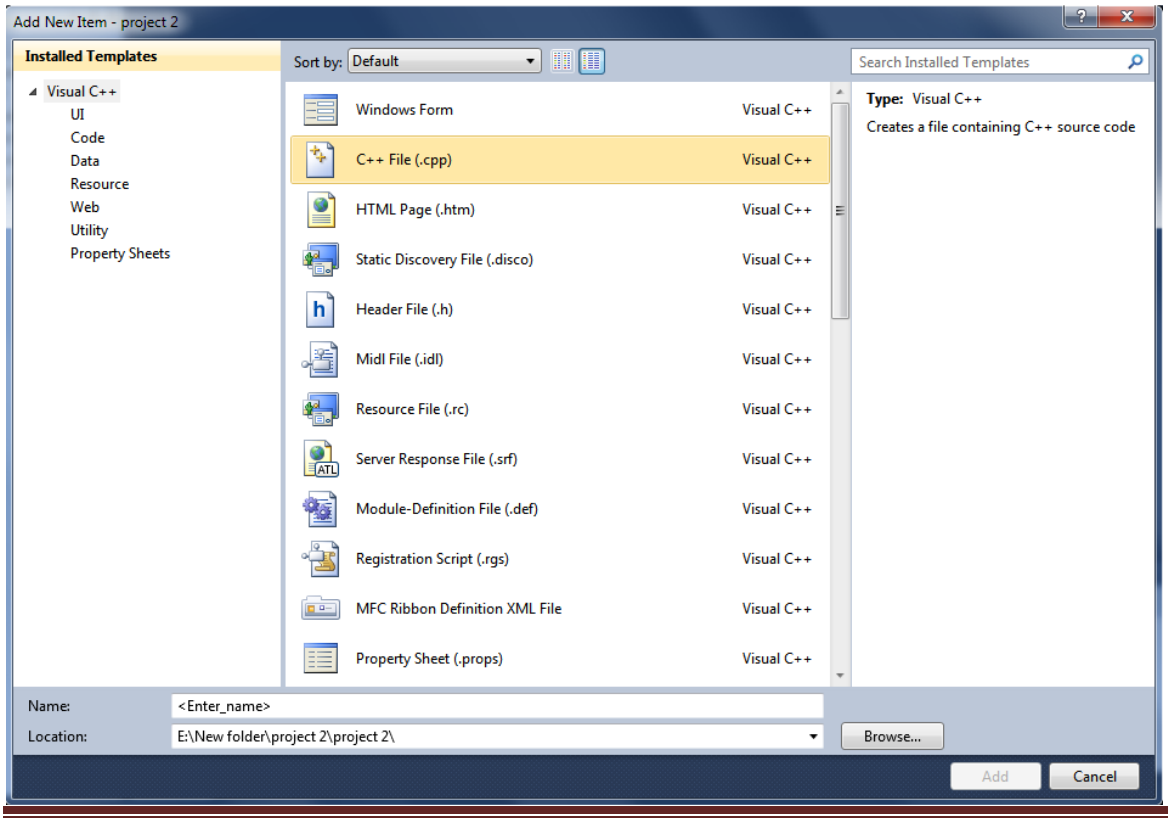
سپس در سمت چپ قسمت Visual C++ را انتخاب کرده و از قسمت سمت راست گزینه Empty Project را انتخاب می‌کنیم، همچنین در پایین در کادر name بک نام به پروژه‌ی خود اختصاص دهید و محل ذخیره پروژه را با استفاده از دکمه‌ی browse مشخص کنید:



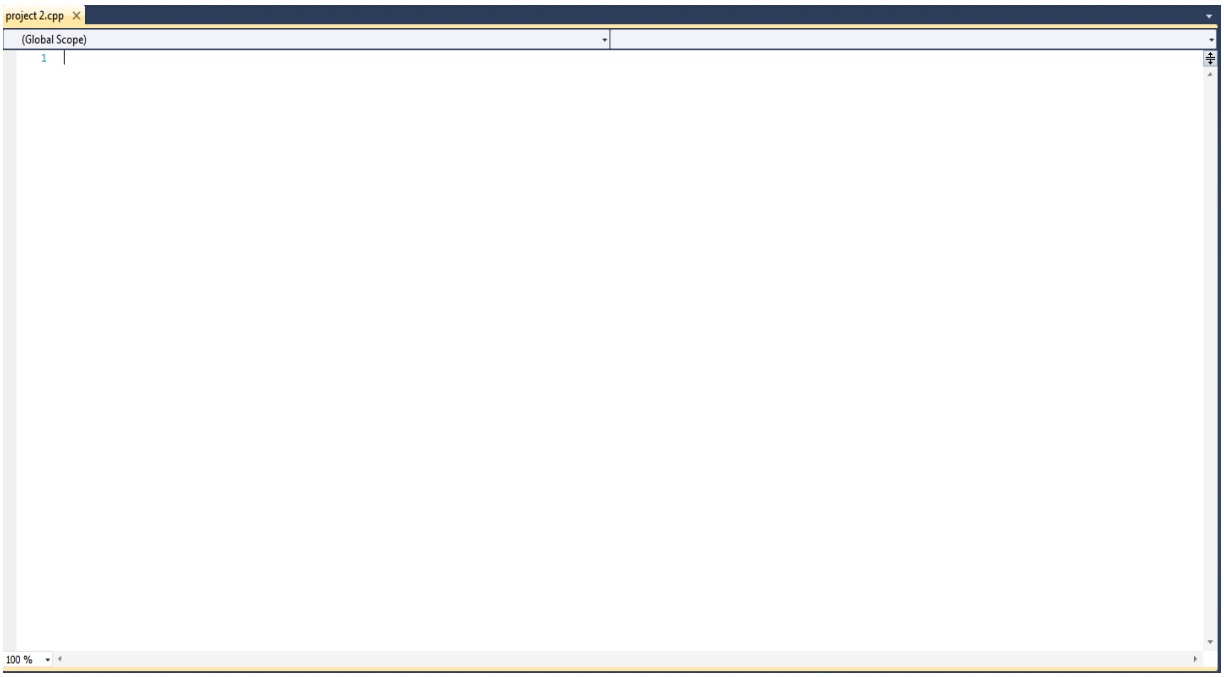
پس از درست شدن پروژه جدید همانند شکل زیر بر روی Source Files کلیک راست کرده و گزینه New item را انتخاب می‌کنیم:



سپس در پنجره‌ی باز شده در سمت راست همانند شکل زیر روی گزینه‌ی C++ File(.cpp) کلیک می‌کنیم و نام مناسب منبع کد را نیز در قسمت name (پایین صفحه) مشخص می‌کنیم:



در نهایت همانند شکل زیر محیط سفید رنگ کد نویسی باز می‌شود، که شما در اینجا کدهای خود را می‌نویسید:



در شکل زیر کد یک برنامه وجود دارد که پیغام welcome to c++! را برای ما در مانیتور، چاپ می‌کند:

```
1  # include <iostream> // allows program to output data to the screen
2  # include <conio.h>
3
4  using namespace std;
5
6  int main()
7  {
8      cout<< "welocome to c++!"; // display message
9
10     getch();
11     return 0; // indicate that program ended successfully
12 }
13
14
```

حال به توضیح قسمت های مختلف کد بالا می‌پردازیم:

نتیجه اجرای این کد نمایش جمله‌ی Welcome to c++! است.

خط ۱ و ۲ دستور های پیش پردازنده است، یعنی قبل از کامپایل (کار تبدیل کدهای برنامه به دستورهای قابل فهم کامپیوتر) برنامه این خط اجرا می‌شود، به عبارت دیگر هر خطی که با # شروع می‌شود یک دستور پیش پردازنده است.

دستور خط ۱ به پیش پردازنده می‌گوید که محتوای سرفایل جریان ورودی/ خروجی را در برنامه قرار دهد.

عبارت سبز رنگی که با علامت // شروع می‌شود، توسط کامپایلر نادیده گرفته می شود، زیرا از این نوع خطوط برای وضوح بخشیدن به برنامه استفاده می‌شود، همان طور که مشاهده می‌کنید، روبروی خط ۱ توضیحی برای علت استفاده از این خط نوشته شده است.(در این باره در نکات تکمیلی توضیح می‌دهم)

خط ۳ یک خط خالی است که کامپایلر آن را نادیده می‌گیرد؛ در کل خطوط خالی در کامپایل برنامه خللی ایجاد نمی‌کند و همچینی به خوانایی کد کمک می‌کند.

خط ۴ به این معنی است که ما از دستورهای استاندارد استفاده می‌کنیم، که باید در تمام برنامه‌ها وجود داشته باشد.

از خط ۶ تا ۱۲، بدنه‌ی اصلی برنامه را تشکیل می‌دهد، که هر برنامه‌ی ما حداقل باید یک بدنه‌ی اصلی داشته باشد.

کلماتی که به رنگ آبی نمایش داده شده است، دستور های **اصلی** C++ هستند، دستور int نوع متغیر است. (که در قسمت‌های بعد توضیح داده خواهد شد.)

دستور ()main یک تابع است که نمایانگر بدنه‌ی اصلی برنامه است، که گیومه خالی به این علت است که برنامه مقداری را برنمی‌گرداند، همین طور بدنه‌ی اصلی **باید** بین دو کروشه باز و بسته {} (خطوط ۷ و ۱۲) قرار گیرد. (البته در قسمت‌های بعد به توضیح این تابع می‌پردازیم.) در حالت کلی دستورهای شامل گیومه، تابع هستند.

خط ۸، خطی است که جمله‌ی !Welcome to c++ را در مانیتور نمایش می‌دهد. (این همه خط کد برای نمایش همین چند یه جمله!) دستور cout همراه با علامت >> قسمت بعد از این علامت را در خروجی نمایش می‌دهد، که می‌تواند شامل متن (همانند این مثال)، یک متغیر و ... باشد، همچنین دستور \n باعث می‌شود، مکان نما به خط جدید برود. (درباره‌ی این نوع دستورها در نکات تکمیلی توضیح خواهم داد)

خط ۱۰ به این دلیل به کار می‌رود تا پنجره‌ای که خروجی برنامه را نمایش می‌دهد تا زمانی که ما کاری برای بستن آن انجام ندهیم، بسته نشود، حتما الان این سوال را از خود پرسیده‌اید که مفهوم بسته نشدن برنامه یعنی چه؟ ببینید به علت اینکه حاصل نتایج کار شما بر روی یک صفحه سیاه! نمایش داده می‌شود و به عبارت دیگر خروجی برنامه به حالت dos است شما باید از پایه تمام کارهای برنامه را بنویسید (همون طور که در قسمت قبل به دقیق بودن کدهای برنامه اشاره کردم)، به همین دلیل باید این دستور هم برای بسته نشدن پنجره بنویسید. (برای درک بهتر می‌توانید این کد را یک بار حذف کرده و نتیجه را ببینید، البته میدونم که یه کم فهم این قسمت سخته به همین دلیل شما فقط برای بسته نشدن پنجره این دستور رو در هر حالت بنویسید!)

خط ۱۱، دستوری است که نشان می‌دهد برنامه به پایان رسیده است، و عملیات ترجمه کد باید خاتمه یابد، البته این خط بیشتر زمانی به کار می‌رود که قرار است شما برنامه‌ی خود را بر روی فضایی اجرا کنید که دارای حافظه‌ی بسیار محدود است، ولی با توجه به اینکه فضای کامپیوتر شما بسیار زیاد است می‌توانید این کد را ننویسید.

نکات تکمیلی

۱. C++ به بزرگی و کوچکی حروف حساس است، یعنی main و Main دو معنی جدا از هم دارند، که **تمام** دستورهای C++ با حروف **کوچک** نوشته می‌شوند.

۲. تمام دستورهای C++ با علامت ; (سِمی کالِن) به پایان می‌رسند، **به جز** دستورهای پیش پردازنده.

۳. تفاوت " و ' : هرگاه **متنی** بین " " (دابِل کوتِیشِن) بیاید به آن رشته یا لیتِرال گفته می‌شود، ولی بین ' ' (کوتِیشِن) یک متن **نمی‌تواند** بیاید، زیرا این علامت برای نمایش **یک** کاراکتر به کار می‌رود.

۴. به کاراکتر \ کاراکتر **کنترلی** یا کاراکتر **گریز** گفته می‌شود، که نشان‌دهنده‌ی کاراکتر خاصی بر روی خروجی است (این کاراکترها در دستور cout استفاده می‌شوند)، که چند نمونه از این نوع کاراکترها عبارتند از:

\n خط جدید: مکان نما را به ابتدای خط بعد منتقل می‌کند.

\t جدول بندی افقی: مکان نما را به محل ستون بعدی (به اندازه‌ی ۴/۱ اینچی) می‌برد.

\r برگشت به اول سطر: مکان نما را در ابتدای خط فعلی قرار می‌دهد ولی آن را تا خط بعد جلو نمی‌برد.

\a زنگ: صدای سیستم به صدا در می‌آید. (عموما برای نوشتن کد خطا به کار می‌رود، البته استفاده های دیگه ای هم داره که میتونید خودتون پیدا کنید!)

۵. دستور های جریان ورودی/ خروجی به ترتیب شامل دستورهای cin/cout هستند، که استفاده از این دستورها را در برنامه های بعدی که می‌نویسیم، توضیح خواهیم داد، البته این دستورها را با این علامت ها به کار می‌روند: >>cout و <<cin

۶. برای اینکه کدی که شما می‌نویسید خوانا باشد و اگر روزی آن را به شخص دیگری بدهید، بتواند آن را بخواند، بهتر است از علامت های توضیحی استفاده کنید که شامل // و /* */ است، علامت // برای توضیح یک خطی به کار می‌رود و علامت های /* */ برای توضیح چند خطی به کار می‌روند، که توضیح های چند خطی با علامت */ شروع و با علامت /* به پایان می‌رسد، یعنی هر عبارتی بین این دو علامت قرار بگیرد، توسط برنامه توضیح تلقی می‌شود.

۷. دستورهایی که با علامت # شروع می‌شوند، که به آنها پیش پردازنده می‌گویند، در اصل هرکدام از آنها یک کتابخانه (library) سی پلاس پلاس را فراخوانی می‌کنند. حال کتابخانه چیست؟ منظور از کتابخانه در C++ یعنی مجموعه تابع ها و دستورهایی که ما با فراخواندن یکی از دستورهای پیش پردازنده آن ها را در اختیار برنامه قرار می‌دهیم، به بیان ساده‌تر، وقتی شما یک دستور پیش پردازنده مثل iostream را در برنامه به عنوان کد پیش پردازنده می‌نویسید، تمام دستورها و تابع های مربوط به کتابخانه ورودی/خروجی در اختیار برنامه قرار می‌گیرد (همون طور که در توضیح خط ۱ کد گفتم، محتوای سرفایل ورودی/خروجی، منظورم همین بودا!) به عنوان مثالی دیگر اگر دستور پیش پردازنده math.h را به صورت زیر در برنامه قرار دهید، تمام تابع ها و عملگرهای ریاضی مثل توان، سینوس، کسینوس و... در برنامه قرار می‌گیرد:# include <math.h>

مثال: برنامه زیر یک حرف انگلیسی کوچک را گرفته به حرف بزرگ تبدیل می‌نماید.

```
using namespace std;
#include    <iostream >
#include    <stdlib. h>
int main( )
{
 char c1 , c2;
cout << "Enter a lowercase letter:"
cin >> c1;
c2 = toupper(c1);
cout << c2 << endl;
return 0; }
```

مثال: دو عدد از نوع اعشاری را گرفته مجموع و حاصلضرب آن ها را محاسبه و نمایش می‌دهد.

```
using namespace std;
#include    <iostream >
int main( )
{
float        x,y,s,p ;
cin >> x >> y ;
s= x+y ;
p=x*y;
cout << s <<endl << p;
return 0 ;
}
```

فصل دوم: ساختارهای تصمیم گیری و تکرار

- عملگر های رابطه ای
- دستورالعمل شرطی
- عملگر های منطقی
- عملگر شرطی
- عملگر کاما
- دستورالعمل For

عملگرهای رابطه ای

از این عملگرها برای تعیین اینکه آیا دو عدد با هم معادلند یا یکی از دیگری بزرگتر یا کوچکتر می‌باشد استفاده می‌گردد. عملگرهای رابطه‌ای عبارتند از:

مساوی	= =
مخالف	! =
بزرگتر	>
بزرگتر یا مساوی	> =
کوچکتر	<
کوچکتر یا مساوی	<=

عملگر شرطی

شکل کلی عملگر شرطی بصورت زیر می‌باشد:

expression _ test ? expression _ true : expression _ false

عملگر شرطی تنها عملگری در C++ می‌باشد که دارای سه عملوند می‌باشد.

مثال۱:

```
int x=10,y=20,b;
b=(x>y) ? x : y ;
```

این دو دستور العمل باعث می شوند که ماکزیمم مقادیر y و x در b قرار بگیرد.

```
x>=10 ? cout << "passed" : cout << "failed" ;
```

اگر مقدار x بزرگتر یا مساوی ده باشد رشتهٔ passed در غیر اینصورت رشته failed نمایش داده می شود.

دستورالعمل شرطی

توسط این **دستور شرطی** را تست نموده و بسته به آنکه شرط درست یا غلط باشد عکس‌العمل خاصی را نشان دهیم.

```
if (عبارت )
{
 1 دستورالعمل ;
...
 n  دستورالعمل;
 }
 else
 {
  1 دستورالعمل ;
 ...
  n  دستورالعمل;
  }
```

مثال ۱:

```
if(x != y)
        { cout << x ;  ++ x ; }
else
        { cout << y ;  - - y ; }
```

در این مثال اگر x و y مخالف هم باشند، x را نمایش داده و سپس به x یک واحد می افزاید. در غیر این صورت یعنی زمانی که x و y مساوی هم هستند، مقدار y را نمایش داده و از y یک واحد کم می کند.

مثال ۲:

برنامه زیر یک عدد اعشاری را از ورودی گرفته جذر آن را محاسبه می‌نماید.

```
using namespace std;
#include   <iostream >
#include    <math . h>
int main( )
{
float  x,s;
cin >> x ;
if( x < 0 )
cout << " x is negative" << endl ;
else
{
s = sqrt(x) ;
cout << s << endl ;
                }
return 0;
}
```

عملگر کاما

تعدادی عبارت را می‌توان با کاما بهم متصل نمود و تشکیل یک عبارت پیچیده‌تری را داد. این عبارت ها به ترتیب از چپ به راست ارزیابی شده و مقدار عبارت معادل عبارت n می‌باشد.

(عبارت n , ... , عبارت 3 , عبارت 2 , عبارت 1)

مثال: اگر داشته باشیم ; int a=2 , b=4 , c=5 عبارت زیر را در نظر بگیرید:

(++ a , a+b, ++ c, c+b)

ابتدا به a یک واحد اضافه شده، سپس a و b جمع می شوند، بعد از آن به c یک واحد اضافه می شود و در نهایت c و b با هم جمع می شوند. مقدار عبارت حاصل برابر است با b+c که معادل 10 می‌باشد.

عملگرهای منطقی

با استفاده از عملگرهای منطقی می‌توان شرط های ترکیبی در برنامه ایجاد نمود. عملگرهای منطقی عبارتست از :

AND

OR

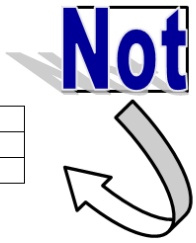
NOT

که در C++ به ترتیب بصورت && ، || و ! نشان داده می شود.

جدول درستی سه عملگر شرطی

a	b	a && b
true	true	True
true	false	False
false	true	False
false	false	False

a	b	a || b
true	true	True
true	false	True
false	true	True
false	false	False



a	!a
true	False
false	True

مثال:

```
if ((x= = 5) ||(y != 0))
   cout << x << endl ;
```

اگر x برابر با 5 یا y مخالف صفر باشد مقدار x نمایش داده شود .

```
if(x)
 x = 0 ;
```

اگر مقدار x مخالف صفر باشد، آنگاه x برابر با صفر شود .

برنامه زیر طول سه پاره‌خط را از ورودی گرفته مشخص می‌نماید که آیا تشکیل یک مثلث می دهد یا خیر؟

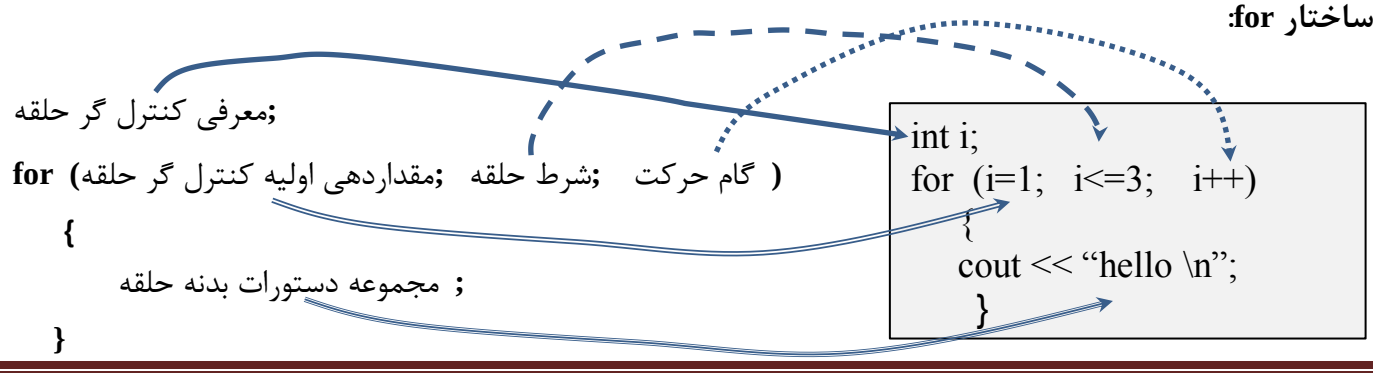
```
 using namespace std;
 #include    <iostream >
int main( )
{
float  a, b, c;
cout << "Enter three real numbers" << endl ;
cin >> a >> b >> c;
if(( a < b + c) &&(b < a+c) &&(c < a+b))
cout << "It is a triangle" ;
else
cout << "Not a triangle" ;
return  0 ;
}
```

دستورالعمل For

از دستور العمل for برای تکرار دستورالعمل ها استفاده می شود. شکل کلی دستور for بصورت زیر می‌باشد:

```
for (عبارت 1 ; عبارت 2 ; عبارت 3)
{
 دستورالعمل 1 ;
 دستورالعمل 2 ;
 .
 .
 .
 n دستورالعمل;
}
```

ساختار for:



مثال

```
void main()
{
cout << “hello \n”;
cout << “hello \n”;
cout << “hello \n”;
}
```

=

```
void main()
{
int i;
for  (i=1;   i<=3;    i++)
   {
   cout << “hello \n”;
```

نکات:

- لزومی ندارد که کنترل گر حلقه حتماً از ۱ شروع شود.

```
int i;

for  (i=5;   i<=7;    i++)
    {
    cout << “hello  \n”;
   }
```

- مقدار دهی اولیه کنترل گر حلقه می تواند خارج از دستور for باشد.

```
int i =1;

for  (  i;   i<=3;    i++)
    {
    cout << “hello  \n”;
   }
```

- مقدار دهی اولیه کنترل گر حلقه می تواند خارج از دستور for باشد. در این صورت می توان جمله اول موجود در عبارت for را خالی گذاشت.

```
int i=1;

for ( i;  i<=3;   i++)
    {
    cout << "hello \n";
   }
```

=

```
int i=1;

for (    ;  i<=3;   i++)
    {
    cout << "hello \n";
   }
```

- گام حرکت می تواند در بدنه دستور for تعریف شود.

```
int i=1;

for ( i;  i<=3;       )
    {
    cout << "hello \n";
    i++;
   }
```

- معرفی کنترل گر حلقه می تواند در داخل دستور for باشد.

```
for (int i= 1;  i<=3;   i++)
    {
    cout << "hello \n";
   }
```

- در دستور for اگر قسمت شرط خالی باشد، حلقه همیشه اجرا خواهد شد. به عبارتی هیچ شرطی برای توقف نداریم

```
for (int i= 1;          ;   i++)
    {
    cout << "hello \n";
   }
```

- وقتی هیچ شرطی نداریم، دو قسمت دیگر دستور for نیز می توانند خالی باشند.

```
for (          ;          ;       )
    {
    cout << "hello \n";
   }
```

- لزومی ندارد که گام حرکت بصورت افزایشی باشد بلکه می تواند بصورت کاهشی نیز باشد.

```
for  (int i= 3;   i >= 1;    i--)
    {
    cout << “hello \n”;
   }
```

تمرین: اعداد ۱۰۰ تا ۱ را به صورت نزولی چاپ نمایید. (با گام حرکت افزایشی و کاهشی جداگانه بنویسید)

- گام حرکت می تواند افزایش یا کاهش بیش از ۱ واحد را داشته باشد.

مثال: چاپ اعداد فرد بین ۱ تا ۱۰۰

```
for  (int k= 1;  k <= 100;    k=k+2)
    {
    cout << k << “\n”;
   }
```

- کنترل گر حلقه می تواند اعشاری یا کاراکتری باشد، لزومی ندارد که حتما عدد صحیح در نظر بگیریم.

```
for  (char ch= ‘a’;   ch <= ‘z’;    ch++)
    {
    cout << ch << “\n”;
   }
```

برنامه زیر عدد صحیح و مثبت n را از ورودی گرفته فاکتوریل آن را محاسبه ونمایش می‌دهد.

```
using namespace std;
#include   <iostream >
int  main( )
{
int  n, i ;
long fact = 1 ;
cout << "Enter a positive integer number";
cin >> n;
for( i=1; i<=n; ++i)
    fact *= i;
 cout << fact << endl;
return 0 ;
}
```

برنامه زیر مجموع اعداد صحیح و متوالی بین ۱ تا n را محاسبه نموده و نمایش می‌دهد.

```
using namespace std;
#include   <iostream >
int     main( )
{
int n, i=1 ;
long s = 0 ;
cin >> n ;
for(; i<=n; i++)
     s += i;
cout << s ;
return  0 ; }
```

برنامه زیر ارقام 0 تا 9 را نمایش می‌دهد.

```
using namespace std;
#include   <iostream >
int main( )
{
int  j=0 ;
for( ; j <= 9 ; )
     cout << j++ << endl;
return 0 ;
}
```

تمرینات:

- برنامه ای که اعداد ذوج بین ۱ تا ۱۰۰۰ را چاپ کند.
- برنامه ای که ۱۰۰ عدد را خوانده، مجموع را محاسبه و چاپ کند.
- برنامه ای که حاصلضرب اعداد ۱ تا ۵۰ را چاپ کند.
- برنامه ای که ۵۰ کاراکتر از صفحه کلید خوانده و تعیین کند که کدام یک حرف کوچک است.
- برنامه ای که ۱۰۰ عدد صحیح را خوانده max و min را چاپ کند.
- برنامه ای که کاراکتر هایی که کد آن ها بین ۱۰۰ تا ۲۰۰ هست، را چاپ نماید.

کاربرد دستور break در دستور for

اگر در بدنه for از جمله ی ;break استفاده شود، ادامه ی اجرای حلقه متوقف شده و حلقه خاتمه می یابد.

```
int i , x;
for(i=1;i<=100;i++)
{cin>>x;
if (x==50) break;
}
```

قطعه کد فوق حداکثر ۱۰۰ عدد صحیح از ورودی می گیرد، ولی اگر در بین اعداد ورودی عدد ۵۰ وارد شود بدون بررسی شرط حلقه از ادامه اجرای دستورات for اجتناب کرده و از حلقه خارج می شود.

مثال:

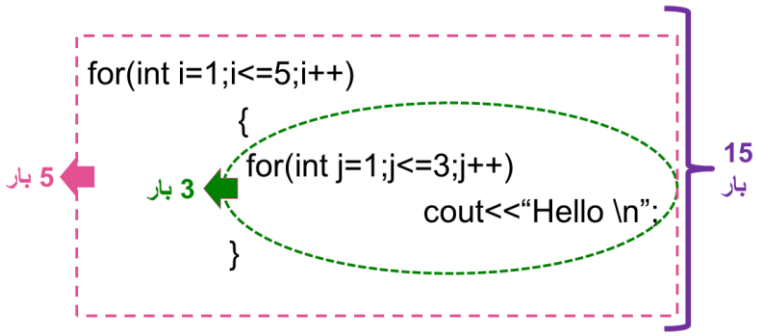
قطعه کدی که تعدادی کاراکتر از صفحه کلید خوانده، بعد از فشردن دکمه ی E تعداد آن ها را مشخص کند:

```
char ch;
int i;
for(i=0;    ;i++)
            {cin>>ch;
            if (ch==‘E’) break;
            }
cout<<i;
```

حلقه for تودرتو

می توان داخل بدنه ی دستور for هر دستور دلخواه دیگری نوشت. به عنوان مثال می توان از یک دستور for در بدنه دستور for استفاده کرد.

قطعه کد زیر عبارت Hello را ۱۵ بار اجرا می کند:



برنامه زیر کلیه اعداد سه رقمی که با ارقام 1 ، 2 ، 3 ایجاد می‌شوند را نمایش می‌دهد.

```
using namespace std;
#include   <iostream >
int main( )
{
int  i,j,k,n;
for(i=1; i<=3; ++i)
    for(j=1; j<=3; ++j)
          for(k=1; k<=3; ++k)
          {
          n=i*100 + j*10+k;
          cout << n << '\n' ;
          }
return 0 ;
}
```

مثال: چاپ جدول ضرب اعداد

```
using namespace std;
#include   <iostream >
int main()
{int i,j;
for(int i=1;i<=10;i++)
                for(int j=1;j<=10;j++)
                        { cout<<i*j<<"    ";
                          if (j==10)   cout<<'\n';
                         }
return 0;
}
```

مثال: برنامه ای که ۲۰ عدد را خوانده و برای هرکدام مجموع اعداد ۱ تا آن عدد را محاسبه کند.

```
using namespace std;
#include   <iostream >
int main()
{
int i,j,x,sum;
for(int i=1;i<=20;i++)
        {sum=0;   cin>>x;
        for(int j=1;j<=x;j++)
                        sum+=j;
         cout<<"sum of 1 to "<<x<<" :"<<sum<<endl;
        }
return 0;
}
```

کاربرد حلقه for با دو اندیس

مثال: برنامه ای بنویسید که ستون اعداد زیر را چاپ کند:

روش اول:

```
for(int i=1;i<=20;i++)
   cout<<i<<", "<<21-i<<endl;
```

روش دوم:

```
for(int i=1, j=20;   i<=20;   i++, j--)
   cout<<i<<", "<<j<<endl;
```

```
1, 20
2, 19
3, 18
4, 17
  .
  .
  .
20, 1
```

دستور continue; در for

اگر دستور continue; در حلقه for استفاده شود، جملاتی از حلقه که هنوز اجرا نشده اند، بدون اجرا مانده و ادامه اجرا از انتهای حلقه اغاز خواهد شد.

```
int main()
{
int x, p=1;
for(int i=1;   i<=20;    i++)
          {cin>>x;
           if (!x) continue;
            p*=x;
           }
cout<<p;
return 0;
}
```

برنامه فوق ۲۰ عدد از ورودی خوانده و حاصلضرب اعداد غیر صفر را در متغیر p محاسبه کرده و در نهایت نمایش می دهد.

فصل سوم: سایر ساختارهای تکرار

- دستورالعمل while
- دستورالعمل do while
- دستورالعمل break
- دستورالعمل continue
- دستورالعمل switch
- تابع cin.get( )
- عملگر static_cast < > ( )
- جدول اولویت عملگرها

دستورالعمل while

از این دستور العمل مانند دستورالعمل for برای تکرار یک دستورالعمل ساده یا ترکیبی استفاده می‌گردد. شکل کلی این دستور العمل بصورت زیر می‌باشد.

```
while( شرط)
{
 ; دستورالعمل ۱
 ; دستورالعمل ۲
    .
    .
 ; دستورالعمل n
}
```

تفاوت دستورهای while و for

دستورالعمل for زمانی استفاده می شود که تعداد دفعات تکرار از قبل مشخص و معین باشد. در صورتیکه تعداد دفعات تکرار مشخص نباشد بایستی از دستورالعمل while استفاده نمود. دقت کنید با پیاده سازی مکانیزم های خاصی می توان بدون توجه به دفعات تکرار از هر دو دستور بصورت مشابه استفاده کرد.

مثال:

```
int x=0
while(x<5)
cout << x ++<< endl;
```

با اجرای قطعه برنامه فوق مقادیر فوق نمایش داده می شود :

```
0
1
2
3
4
```

- نکته: همانند دستور for اگر داخل بدنه دستور while فقط یک جمله باشد می توان } و { را حذف کرد.

مثال: نمایش اعداد ۰ تا ۱۰۰:

```
int main( )
{
int j = 0;
while(j<=100)
                cout<<j++<<endl;

return 0 ;
}
```

تمرین: برنامه زیر را تفسیر کنید:

```
int main( )
{
int j = 0;
while(j<10)
      J++;
      cout<<j;
return 0 ;
}
```

مثال: برنامهٔ فوق n مقدار از نوع اعشاری را گرفته میانگین آن ها را محاسبه و در متغیر avg قرار می‌دهد.

```
using namespace std;
#include   <iostream >
int main( )
{
int count = 0 , n;
float  x, sum = 0 , avg ;
cin >> n ; /*  تعداد مقادیر ورودی n*/
while(count < n){
cin >> x ;
sum += x ;
++ count ; }
avg = sum / n ;
cout << avg << endl;
return 0 ; }
```

دستورالعمل do while

این دستور العمل نیز برای تکرار یک دستورالعمل ساده یا ترکیبی استفاده می‌شود.

شکل کلی این دستورالعمل بصورت روبرو می‌باشد.

```
معرفی کنترل گر حلقه;
مقدار دهی اولیه کنترل گر حلقه;

do
    { دستورالعمل ۱ ;
      دستورالعمل ۲ ;
      .
      .
     دستورالعمل n ;
     گام حرکت;
     } while(شرط);
```

تفاوت دستورهای do while و while

در دستورالعمل while ابتدا مقدار شرط ارزیابی شده اما در دستورالعمل do while ابتدا دستورالعمل اجرا شده سپس مقدار شرط ارزیابی می‌گردد. بنابراین دستورالعمل do while حداقل یک بار انجام می شود .

مثال: برنامه فوق، ارقام 0 تا 9 را روی ده خط نمایش می‌دهد.

```
using namespace std;
#include <iostream>
int main( )
{
int count = 0;
do
cout << count ++<<endl ;
while(count <= 9);
return 0 ; }
```

دستورالعمل break

این دستورالعمل باعث توقف دستورالعمل های تکرار(for , while ,do while) شده و کنترل به خارج از این دستورالعمل ها منتقل می‌نماید.

چند مثال:

```
int main( )
{
int count = 0 ;
while( 1 )
{
count ++ ;
if(count > 10 )
break ;
}
cout << "counter :   " << count << "\n";
return 0 ;
}
```

```
counter : 11
```

```
int main( )
{
float x, s=0.0 ;
cin >> x ;
while(x <= 1000.0) {
if(x < 0.0){
cout << "Error-Negative Value" ;
break;
}
s += x ;
cin >> x ;}
cout << s << endl ;
return 0 ; }
```

جمع تعدادی عدد که بین 0 و 1000 هستند. اگر بین اعداد وارد شده عدد منفی وارد شود، بواسطه دستور break; حلقه خاتمه می یابد.
اگر عدد وارد شده بزرگتر از 1000 باشد نیز شرط حلقه برآورده نشده و حلقه خاتمه می یابد

```
void  main( )
{
int count;
float x, sum = 0;
cin >> x ;
for(count = 1; x < 1000 . 0; ++ count )
{
if(x < 0.0) {
cout << "Error – Negative value " <<endl;
break ;
}
sum += x ;
cin >> x ;   }
cout << sum << '\n' ; }
```

جمع تعدادی عدد اعشاری که بیشتر از 0 و کمتر از 1000 هستند را محاسبه می کند. اگر بین اعداد وارد شده عدد منفی وارد شود، بواسطه دستور break; حلقه خاتمه می یابد.
اگر عدد وارد شده بزرگتر یا مساوی 1000 باشد نیز شرط حلقه برآورده نشده و حلقه خاتمه می یابد

```
int main( )
{
float x , sum = 0.0 ;
do
{
cin >> x ;
if(x < 0.0)
    {
cout << "Error – Negative Value" << endl ;
break ;
    }
sum += x ;
} while(x <= 1000.0);
cout << sum << endl ;
return 0 ; }
```

جمع تعدادی عدد که بین 0 و 1000 هستند. اگر بین اعداد وارد شده عدد منفی وارد شود، بواسطه دستور break; حلقه خاتمه می یابد.
اگر عدد وارد شده بزرگتر از 1000 باشد نیز شرط حلقه برآورده نشده و حلقه خاتمه می یابد.
نکته: این حلقه حداقل یک بار اجرا می شود.... حتی اگر عدد اول بزرگتر از 1000 باشد نیز در مجموع شرکت داده خواهد شد. سپس از حلقه خارج خواهیم شد.

دستورالعمل continue

از دستورالعمل continue می‌توان در دستورالعمل های تکرار do while ، while ، for استفاده نمود. این دستورالعمل باعث می‌شود که کنترل به ابتدای دستورالعمل های تکرار منتقل گردد.

چند مثال:

```
int main( )
{
float  x, sum = 0.0 ;
Do
 {
cin >> x ;
if(x < 0 . 0)
      {
cout  << "Error" << endl ;
continue ;
      }
sum += x ;
} while(x <= 1000.0 );
cout << sum ;
return 0 ; }
```

جمع تعدادی عدد که کوچکتر و مساوی 1000 هستند را بدست می اورد. اگر بین اعداد وارد شده عدد منفی وارد شود، در حاصل جمع شرکت نخواهد کرد و اجرای دستورالعمل ها به انتهای حلقه هدایت خواهد شد.
اگر عدد وارد شده بزرگتر از 1000 باشد نیز شرط حلقه برآورده نشده و حلقه خاتمه می یابد.
نکته: این حلقه حداقل یک بار اجرا می شود.... حتی اگر عدد اول بزرگتر از 1000 باشد نیز در مجموع شرکت داده خواهد شد سپس از حلقه خارج خواهیم شد.

```
int main( )
{
int n , navg = 0 ;
float  x, avg, sum = 0 ;
cin >> n ; / * عبارت از تعداد اعداد ورودی n * /
for(int count = 1 ; count <=n; ++ count )
{
cin >> x ;
if(x ==   0 ) continue ;
sum += x ;
++ navg ;
}
avg = sum / navg;
cout << avg << endl ;
return 0 ;
 }
```

n عدد از ورودی خوانده. در صورتیکه صفر نباشد ان را
در مجموع اعداد شرکت می دهد.
در نهایت میانگین اعداد غیر صفر را محاسبه می کند

دستورالعمل switch

همانطور که می دانید از دستورالعمل شرطی(if else) می توان بصورت تودرتو استفاده نمود ولی از طرفی اگر عمق استفادهٔ تو در تو از این دستورالعمل زیاد گردد، درک آنها مشکل می شود . برای حل این مشکل C++ ، دستورالعمل switch که عملاً یک دستورالعمل چند انتخابی می‌باشد را ارائه نموده است.

شکل کلی دستور العمل Switch:

```
switch(عبارت)
{
case    valueone : statement;
                   break;
case     valuetwo: statement;
                  break;
:

case       valuen : statement;
                    break;
default: statement ;
}
```

مثال:

```
void main( )
{
unsigned  int  n;
cin >> n ;
switch(n) {
            case 0:
            case 1:
            case 2:
                    cout << "Less Than Three" << endl;
                    break;
            case 3:
                    cout << "Equal To Three" << endl ;
                    break;
            default:
                    cout << "Greater Than Three" << endl;
          }
}
```

```
void main( )
{
unsigned int n ;
cin >> n;
switch(n)
{
    case  0:
            cout << "ZERO" << endl ;
            break;
    case  1:
            cout << "one" << endl ;
            break ;
    case 2:
            cout << "two" << endl ;
            break;
    default :
            cout << "default" << endl;
}       / * end of switch statement * /
}
```

تابع cin.get() :

این تابع یک کرکتر را از صفحه کلید می‌گیرد. برای استفاده از این تابع در ابتدای برنامه بایستی داشته باشیم:

```
#include      <iostream >
```

مثال: قطعه برنامه ذیل یک کرکتر را از صفحه کلید گرفته و نمایش می‌دهد.

```
char       x;
x = cin.get( );
cout << x ;
```

مثال: برنامهٔ ذیل یک سطر متن انگلیسی که به CTRL Z ختم می شود را گرفته دقیقاً نمایش می‌دهد.

```
#include      <iostream >
int main( )
{
char  x;
while((x = cin.get()) !=EOF)
cout << x ;
return 0 ;
 }
```

EOF به معنی End of File می‌باشد که در iostream.h تعریف شده و مقدار آن برابر با ۱- می‌باشد. مقدار آن در سیستم عامل DOS عبارتست از ctrl z .

مثال: در قطعه برنامه ذیل ازتابع cin.get() و دستور switch استفاده شده است.

```
char         x;
x = cin.get( );
switch(x)  {
case  ′ r ′ :
case  ′ R ′ :
            cout << ″RED″ << ″\n″ ;
            break ;
case  ′ b ′ :
case  ′ B ′ :
             cout << ″BLUE″ << endl ;
             break ;
case  ′ y ′ :
case  ′ Y ′ :
            cout <<  ″YELLOW″ << endl;
   }
```

مثال: برنامهٔ ذیل یک سطر متن انگلیسی را گرفته کرکترهای خالی (blank) آن را حذف نموده و نمایش می دهد.

```
using namespace std;
#include      <iostream >
int main( )
{
char next;
while((next = cin.get( ) ) !=EOF)
if(next != ' ' )
cout  << next ;
return 0 ;
 }
```

عملگر static_cast

از این عملگر برای تبدیل موقت یک نوع data به نوع دیگر استفاده می‌شود. این عملگر یک عملگر یکتائی می‌باشد.

مثال ۱:

```
int   x = 25 ;
float  y ;
y = static_cast < float >(x) ;
```

مقدار x موقتاً بصورت اعشاری در می آید و در نتیجه مقدار y برابر با 25.0 می شود. بایستی توجه داشت که نوع متغیر x عوض نمی شود بلکه موقتاً مقدار آن بصورت اعشاری در آمده است.

مثال ۲:

```
float x = 14.75 ;
cout << static_cast < int >(x) << endl;
cout << x ;
```

ابتدا مقدار ۱۴ نمایش داده می شود و سپس مقدار ۱۴,۷۵ نمایش داده می شود.

جدول اولویت عملگرها

چپ به راست	( )
راست به چپ	**Static_cast < >( ) ++ -- + - sizeof**
چپ به راست	* / %
چپ به راست	+ -
چپ به راست	<< >>
چپ به راست	< <= > >=
چپ به راست	== !=
راست به چپ	? :
راست به چپ	= += -= *= /= %=
چپ به راست	,

تمرینات:

۱. برنامه ای که یک سکه ۱۰۰ ریالی را به سکه های ۲و۵ و۱۰و۲۰ و ۵۰ ریالی خرد کند.

۲. برنامه ای که خروجی زیر را چاپ کند:

۳. برنامه ای که حاصل عبارت زیر را بدست اورد:

$$S = 1 + \frac{2}{2!} + \frac{3}{3!} + ... + \frac{N}{N!}$$

فصل چهارم: اعداد تصادفی

- تولید اعداد تصادفی
- تعریف نوع داده (typedef)
- داده های از نوع شمارشی

اعداد تصادفی

مقادیر تصادفی یا شانسی در اکثر برنامه‌های کاربردی در زمینه شبیه سازی و بازی های کامپیوتری نقش مهمی را ایفا می‌نمایند. برای ایجاد یک عدد تصادفی صحیح بین ۰ و ۳۲۷۶۷ بایستی از تابع ()rand استفاده نمائیم.

مثال: برنامه زیر 10 عدد تصادفی بین 0 و 32767 را ایجاد می‌نماید.

```
using namespace std;
#include      <iostream >
int main( )
{
for(int j=1; j<=10; ++j)
cout << rand( ) << '\n' ;
return 0 ;
}
```

اگر برنامهٔ فوق را چندبار اجرا نمائیم جواب یکسانی را از کامپیوتری می گیریم. برای تصادفی کردن اعداد می‌بایستی از تابع ( )srand استفاده نمائیم. این تابع به یک آرگومان صحیح از نوع unsigned نیاز دارد. به این آرگومان seed گفته می‌شود.

مثال: برنامه زیر 10 عدد تصادفی بین 0 و 32767 را ایجاد می‌نماید. (( )srand)

```
using namespace std;
#include      <iostream >
int main( )
{
unsigned    seed;
cout << "Enter seed value :  " ;
cin >> seed ;
srand(seed);
for(int  j=1; j<=10; ++j)
cout << rand( ) << ' \n ';
return 0  ;
 }
```

مثال: برنامه زیر نتیجه پرتاب دو تاس را نمایش می‌دهد.

```
using namespace std;
#include      <iostream >
int main( )
{
unsigned     seed, d1, d2;
cout << "Enter seed: " ;
cin >> seed ;
srand(seed) ;
d1= 1+rand( )% 6 ;
d2= 1+rand( )% 6 ;
cout << d1 << ″         ″<< d2 ;
return 0 ;
 }
```

مثال: برنامه زیر 10 اعداد شانسی بین 0 و ۱ را نمایش می‌دهد.

```
using namespace std;
#include    <stdlib.h>
#include    < iostream >
int main( )
{
unsigned seed ;
cout << "Enter seed: " ;
cin >> seed ;
srand(seed) ;
for(int i=1; i<=10; ++i)
cout << rand( ) / 32768.0 << endl ;
return 0 ;
}
```

تعریف نوع داده (typedef)

از typedef می‌توان برای تعریف نوع داده‌های جدید که معادل نوع داده‌های موجود باشد استفاده نمود. شکل کلی عبارتست از:

```
typedef    type    newtype;
```

نشان دهنده نوع داده موجود ← type

اسم جدید ← newtype

مثال:

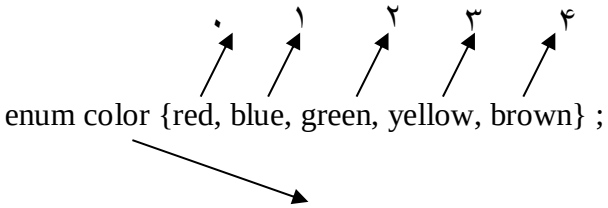
```
typedef  int  integer;
```

حال می‌توان y و x را بصورت روبرو تعریف نمود: **integer x,y;**

داده‌های از نوع شمارشی

بمنظور معرفی داده‌های از نوع شمارشی از کلمه enum استفاده می‌گردد.

مثال :



color یک نوع داده شمارشی می‌باشد.

چند مثال:

emum status {married, devorced, vidow, single}; status a ; a= single ;

enum days {sat, sun, mon, tue, wed, thr, fri};

نکته:

بایستی در نظر داشت که داده‌های از نوع شمارشی در عملیات ورودی و خروجی شرکت نمی‌نمایند. به عبارت دیگر مقادیر داده‌های از نوع شمارشی بایستی در برنامه تعیین نمود. دستورالعل های ورودی و خروجی مانند cin و cout در مورد داده‌های شمارشی نمی‌توان استفاده نمود.

فصل پنجم: آرایه ها

- آرایه یک بعدی
- آرایه دو بعدی (ماتریس ها)

آرایه یک بعدی:

آرایه یک فضای پیوسته از حافظه اصلی کامپیوتر می‌باشد که می‌تواند چندین مقدا را در خود جای دهد.

کلیه عناصر یک آرایه از یک نوع می‌باشند.

عناصر آرایه بوسیله اندیس آن ها مشخص می‌شوند.

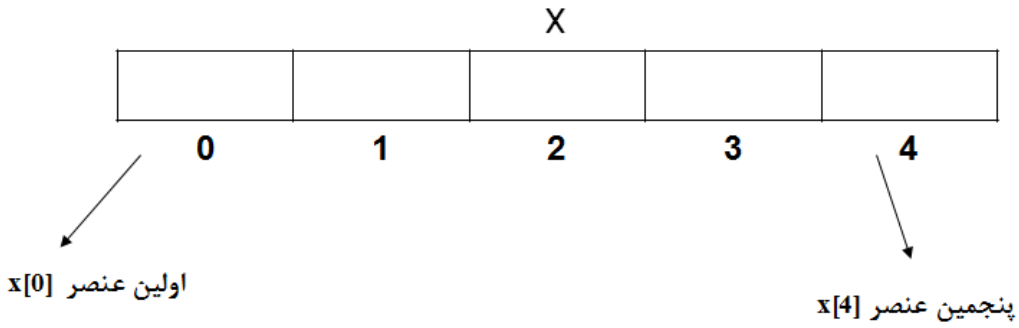
در C++ ، اندیس آرایه از صفر شروع می‌شود.

آرایه‌ها در برنامه‌نویسی در مواردی کاربرد دارند که بخواهیم اطلاعات و داده‌ها را در طول اجرای برنامه حفظ نمائیم.

مثال:

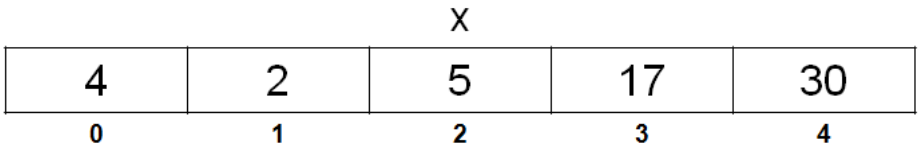
ایجاد آرایه یک بعدی از نوع int

```
int   x[5] ;
```



تخصیص مقادیر اولیه به عناصر آرایه :

```
int  x[5]= {4, 2, 5, 17, 30};
```



دریافت مقادیر عناصر آرایه :

```
int  x[5];
for(int    i=0; i<=4; ++i)
cin >> x[ i ] ;
```

نمایش مقادیر عناصر آرایه :

```
for (int   i=0; i<5; ++i) cout << x[ i ] ;
```

نکته: اگر تعداد مقادیر اولیه کمتر از تعداد عضوهای آرایه باشد عضوهای باقیمانده بطور اتوماتیک، مقدار اولیه صفر می‌گیرند.

مثال:

```
int   x[5] = {12, 5, 7};
```

x

12	5	7	0	0
0	1	2	3	4

نکته: بایستی توجه داشت که آرایه‌ها به صورت ضمنی مقدار اولیه صفر نمی‌گیرند. برنامه نویس باید به عضو اول آرایه، مقدار اولیه صفر تخصیص دهد تا عضوهای باقی‌مانده بطور اتوماتیک، مقدار اولیه صفر بگیرند.

```
int   x[5] = {0} ;
```

x

0	0	0	0	0
0	1	2	3	4

دستور زیر یک آرایه یک بعدی شش عنصری از نوع float ایجاد می‌نماید.

```
float x[  ] = {2.4, 6.3, -17.1, 14.2, 5.9, 16.5} ;
```

x

2.4	6.3	-17.1	14.2	5.9	16.5
0	1	2	3	4	5

برنامه زیر 100 عدد اعشاری را گرفته تشکیل یک آرایه می دهد سپس مجموع عناصر آرایه را مشخص نموده نمایش می‌دهد.

```
using namespace std;
#include      <iostream >
int main( )
{
const int arrsize = 100 ;
float x[ arrsize], tot = 0.0 ;
for(int   j=0; j<arrsize; j++)
cin >> x[ j ];
for(j=0; j<arrsize; j++)
tot += x[ j ] ;
cout << tot ;
return 0 ;
}
```

برنامه زیر 20 عدد اعشاری را گرفته تشکیل یک آرایه داده سپس کوچکترین عنصر آرایه را مشخص و نمایش می‌دهد.

```
using namespace std;
#include      <iostream >
#include   <conio.h>
int main( )
{
float   x[20], s;
int  j ;
for(j=0; j<20 ; ++j) cin >> x[ j ];
s = x[0 ] ;
for(j=1; j<20; ++j)
if (x[ j] <s) s = x[ j ];
cout << s << endl;
return 0;
}
```

برنامه زیر 100 عدد اعشاری را گرفته بروش حبابی (Bubble sort) بصورت صعودی مرتب می‌نماید.

```
using namespace std;
#include <conio.h>
#include <iostream>

int main ( )
{
const int n=10;
float  x[n] , temp;
int   i,j ;
for(i=0; i<n; ++i)
       cin >> x[i];
for(i=n-1; i>0; i--)
 for(j=0 ; j<i; j++)
             if(x[j] > x[j+1])
             {
                   temp = x[j] ;
                   x[j] = x[j+1];
                   x[j+1] = temp ;
             }
for(i=0; i< n ; i++)
         cout << x[i] << endl;
getch();
return 0 ;
}
```

آرایه‌های دوبعدی (ماتریس‌ها)

ماتریس ها بوسیله آرایه‌های دوبعدی در کامپیوترنمایش داده می شوند.

int a[3][4];

	ستون 0	ستون 1	ستون 2	ستون 3
سطر 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
سطر 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
سطر 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

تخصیص مقادیر اولیه به عناصر آرایه:

int a[3][4]={ {1,2,3,4}, {5,6,7,8}, {9,10,11,12} } ;

	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12

نکته۱:

int a[3][4]= { {1}, {2,3} , {4,5,6} } ;

	0	1	2	3
0	1	0	0	0
1	2	3	0	0
2	4	5	6	0

نکته۲:

int a[3][4]= {1, 2, 3, 4,5 } ;

	0	1	2	3
0	1	2	3	4
1	5	0	0	0
2	0	0	0	0

نکته۳:

در یک آرایهٔ دواندیسی، هر سطر، در حقیقت آرایه‌ای یک اندیسی است. در اعلان آرایه‌های دواندیسی ذکر تعداد ستون ها الزامی است.

int a[ ][4]={1,2,3,4,5};

	0	1	2	3
0	1	2	3	4
1	5	0	0	0

مثال: برنامه زیر یک ماتریس 4*3 را گرفته مجموع عناصر آن را مشخص نموده و نمایش می‌دهد.

```
using namespace std;
#include       <iostream >
#include  <conio.h>
int main( )
{
float   x[3][4], total= 0.0;
int   i, j ;
// generate matrix x.
for(i=0; i<3; ++i)
for (j=0; j<4; j++)
cin >> x[ i ][ j ];
// calculate the sum of elements.
for(i=0; i<3; ++i)
for(j=0; j<4; j++)
total + = x [ i ][ j ];
cout << "total =  " << total << endl;
return 0 ;
}
```

تمرین:

برنامه ای بنویسید که دو ماتریس را از ورودی خوانده و حاصلضرب آن را محاسبه و چاپ نماید.
توجه کنید که دو ماتریس زمانی قابل ضرب هستند که تعداد ستون های ماتریس اول با تعداد سطرهای ماتریس دوم برابر باشد.

راهنمایی: ابعاد ماتریس بایستی با مقادیر ثابت در نظر گرفته شوند.

مثلاً با دستورات زیر می توان ماتریسی با اندازه بعد m ساخت.

```
const int m = 5;
int array [m];
```

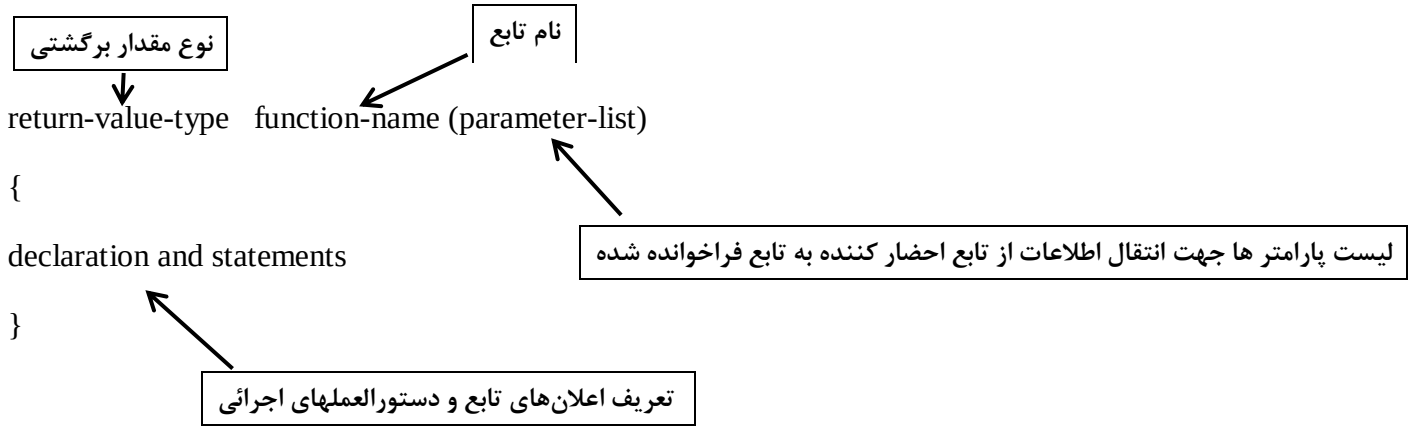
$$[\ddots]_{m\times n} \times [\ddots]_{n\times p} = [\ddots]_{m\times p}$$

فصل ششم: توابع

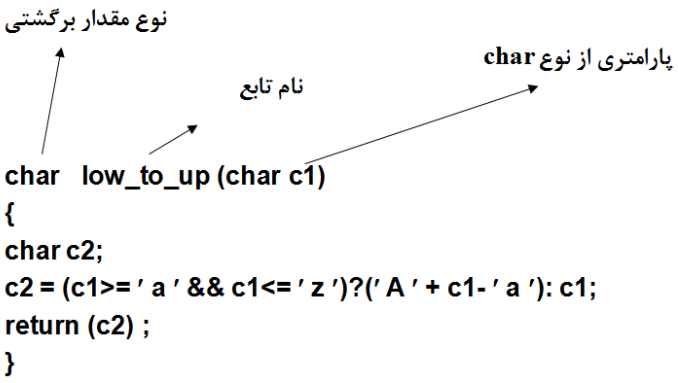
- تعریف تابع
- تابع بازگشتی
- توابع درون خطی
- انتقال پارامترها از طریق ارجاع
- کلاس های حافظه (storage classes)
- سربارگذاری توابع

تعریف توابع

استفاده از توابع در برنامه‌ها به برنامه‌نویس این امکان را می‌دهد که بتواند برنامه‌های خود را به صورت قطعه قطعه برنامه بنویسد. تا کنون کلیه برنامه‌هائی که نوشته‌ایم فقط از تابع ( )main استفاده نموده‌ایم. **شکل کلی توابع بصورت زیر** می‌باشند:



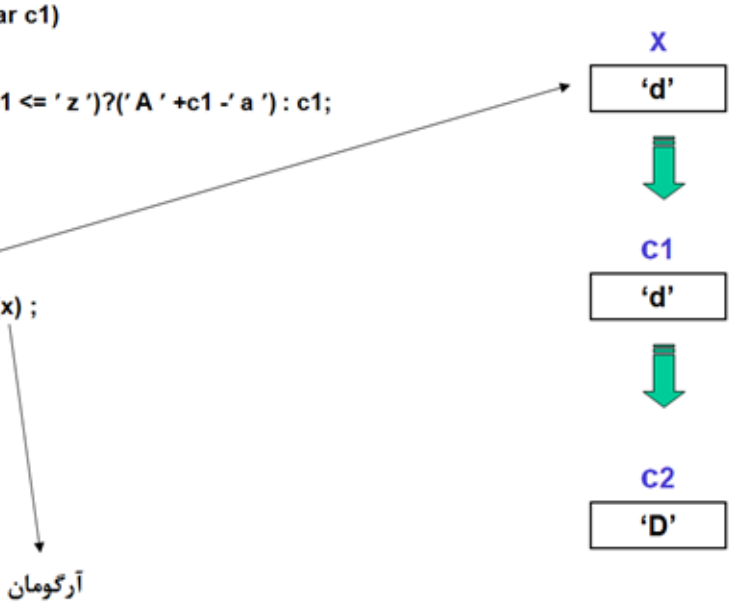
مثال: تابع زیر یک حرف کوچک را به بزرگ تبدیل می‌نماید.




```
char   low_to_up (char c1)
{
char c2;
c2 = (c1>= ' a ' && c1<= ' z ')?(' A ' + c1- ' a '): c1;
return (c2) ;
}
```

برنامه کامل که از تابع قبل جهت تبدیل یک حرف کوچک به بزرگ استفاده می‌نماید.

```
#include  <iostream >
char low_to_up(char c1)
{
char  c2;
c2=(c1 >= ' a ' && c1 <= ' z ')?(' A ' +c1 -' a ') : c1;
return c2;
}
int main( )
{
char x;
x=cin.get( );
cout << low_to_up(x) ;
return 0;
 }
```

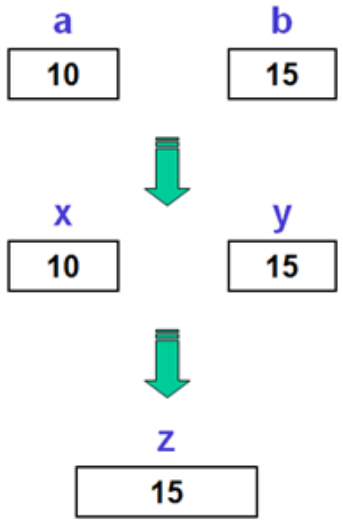


تابع maximum دو مقدار صحیح را گرفته بزرگ ترین آن ها را برمی گرداند.

```
int   maximum(int x, int y)
{
int z ;
z=(x >= y)? x : y;
return z;
}
```

برنامه کامل که از تابع maximum جهت یافتن ماکزیمم دو مقدار صحیح استفاده می نماید.

```
#include  <iostream >
int   maximum(int x , int y)
{
int z ;
z=(x > y)? x : y ;
return z;
}
int main( )
{
int a, b ;
cin >> a >> b ;
cout << maximum(a,b);
return 0;
}
```



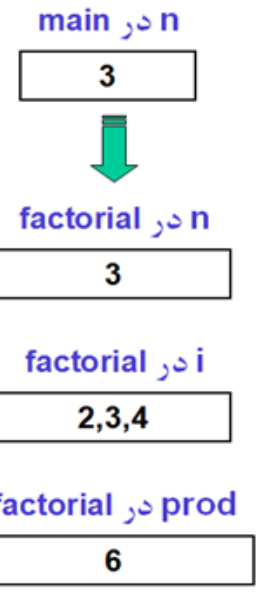
آرگومانهای تابع maximum a, b

نکته: اسامی پارامترها و آرگومان های یک تابع می‌توانند هم نام باشند.

برنامه زیر یک مقدار مثبت را گرفته فاکتوریل آن را محاسبه نموده نمایش می‌دهد.

$$x!=1*2*3*4*...*(x-1)*x$$

```
#include   <iostream>
long int factorial(int n)
{
long int prod=1;
if(n>1)
for(int  i=2; i<=n; ++i)
prod *=i;
return(prod);
}
int main( )
{
int n;
cin >> n ;
cout << factorial(n) ;
return 0 ;
}
```



نکته: وقتی در تابعی، تابع دیگر احضار می‌گردد بایستی تعریف تابع احضار شونده قبل از تعریف تابع احضار کننده در برنامه ظاهر گردد.

نکته: اگر بخواهیم در برنامه‌ها ابتدا تابع main ظاهر گردد بایستی prototype تابع یعنی پیش نمونهٔ تابع که شامل نام تابع، نوع مقدار برگشتی تابع، تعداد پارامترهائی را که تابع انتظار دریافت آن را دارد و انواع پارامترها و ترتیب قرارگرفتن این پارامترها را به اطلاع کامپایلر برساند.

مثال:

```
using namespace std;
#include   <iostream >
#include   <conio.h>
long int factorial(int); // function prototype
int main( )
{
int n;
cout << "Enter a positive integer" << endl;
cin >> n;
cout << factorial(n) << endl;
return 0 ;
}
```

```
long int factorial(int n)
{
long int prod = 1;
if(n>1)
for (int i=2; i<=n; ++i)
prod *= i;
return(prod);
}
```

نکته: در صورتی که تابع مقداری بر نگرداند نوع مقدار برگشتی تابع را void اعلان می‌کنیم. و در صورتیکه تابع مقداری را دریافت نکند بجای parameter- list از void یا ( ) استفاده می گردد.

مثال:

```
#include   <iostream>
#include   <conio.h>
void maximum(int , int) ;
int main( )
{
int x, y;
cin >> x >> y;
maximum(x,y);
return 0;
}
void maximum(int x, int y)
{
int z ;
z=(x>=y) ? x : y ;
cout << "max value \n" << z<< endl;
return ;
}
```

تابع مقداری بر نمی گرداند.

احضار بوسیله مقدار (Call By Value)

```
#include   <iostream>
int modify(int);
int main( )
{
int a=20;
cout << a << endl;
modify(a) ;
cout << a << endl;
return 0 ;
}
int modify(int a)
{
a *= 2;
cout << a << endl;
return 0;
}
```

a در main
20

a در modify
20

a در modify
40

خروجی برنامه :

20

40

20

در این نوع احضار تابع حافظه‌های مورد استفاده آرگومان ها و پارامترها از هم متمایزند و هرگونه تغییر در پارامترها باعث تغییر در آرگومان های متناظر نمی‌گردد.

تابع بازگشتی (recursive functions)

توابع بازگشتی یا recursive توابعی هستند که وقتی احضار شوند باعث می‌شوند که خود را احضار نمایند.

مثال۱: نحوه محاسبه فاکتوریل از طریق تابع بازگشتی

n != 1*2*3*…*(n-1) *n

f(n) = n !

$$f(n) = \begin{cases} 1 & n=0 \quad \text{اگر} \\ n * f(n-1) & \text{در غیر اینصورت} \end{cases}$$

n!= 1* 2* 3* … *(n-2) *(n-1) *n

n!=(n-1)! *n

پیاده سازی تابع بازگشتی محاسبه ی فاکتوریل:

```
# include   <iostream >
long int factorial(int) ;
int main( )
{
    int n ;
    cout << " n= " ;
    cin >> n ;
    cout << endl << " factorial = " << factorial(n) << endl;
    return 0 ;
}
long int factorial(int n)
{
    if(n<=1)
            return(1);
    else
            return(n *factorical(n-1) ) ;
}
```

مثال۲: نحوه محاسبه n امین مقدار دنباله فیبوناچی از طریق تابع بازگشتی

0 , 1, 1, 2, 3, 5, 8, 13, 21 , 34, …

$$\text{جمله n ام} \rightarrow \quad fib(n)=\begin{cases} 1 & \text{اگر } n=1 \\ 1 & \text{اگر } n=2 \\ fib(n-1)+fib(n-2) & \text{در غیر اینصورت} \end{cases}$$

برنامه زیر n امین مقدار دنبالهٔ فیبوناچی (fibonacci) را مشخص و نمایش می‌دهد.

```
# include   <iostream >
long int fib(long int); // forward declaration
int main( )
{
    long int r ;
    int n ;
    cout << " Enter an integer value " << endl ;
    cin >> n ;
    r = fib(n) ;
    cout << r << endl ;
    return 0 ;
}
long int fib(long int n)
{
if (n==0) return 0;
if(n = = 1  ||  n= = 2)
            return 1 ;
    else
            return(fib(n-1) + fib(n-2) ) ;
}
```

مثال۳: برنامه زیر یک خط متن انگلیسی را گرفته آن را وارون نموده نمایش می‌دهد.

```
# include   <iostream >
void reverse(void) ; // forward declaration
int main( )
{
    reverse( ) ;
    return 0 ;
}
```

```
void reverse(void)
// read  a line of characters and reverse it
{
    char c ;
    if(( c=cin.get( )) != ' \n ')
            reverse( );
    cout << c ;
    return ;
}
```

نکته: استفاده از **آرایه‌ها** بعنوان پارامتر تابع مجاز است.

مثال: در برنامه زیر تابع modify آرایه a را بعنوان پارامتر می‌گیرد.

```
# include   <iostream >
void  modify(int [  ] ); // forward declaration
int main( )
{
int  a[5] ;
for(int j=0; j<=4; ++j)
a[ j ] = j+1 ;
modify(a) ;
for(int j=0; j<5; ++j)
cout << a[ j ] << endl ;
return 0 ;
}
void modify(int a[  ]) // function definition
{
for(int j=0; j<5; ++j)
a[ j ] += 2 ;
for(int j=0; j<5; ++j)
cout << a[ j ] << endl ;
return ;
}
```

خروجی :

```
3
4
5
6
7

3
4
5
6
7
```

نکته: در این مثال از فراخوانی با مقدار استفاده نشده است. برای همین تغییرات انجام شده در تابع ()modify باعث تغییر در مقادیر ارایه در تابع اصلی یا ()main شده است. این مبحث، تحت عنوان فراخوانی از طریق ارجاع در ادامه توضیح داده می شود.

مثال: در صورتیکه آرایه بیش از یک بعد داشته باشد بعدهای دوم به بعد بایستی در تعریف تابع و پیش نمونه تابع ذکر گردد.

```
#include   <iostream>
void printarr(int [   ][ 3 ]);
int main( )
{
     int  arrl [2][3] = { {1,2,3}, {4,5,6} } ,arr2 [2][3]= {1,2,3,4,5} ,  arr3 [2][3]={ {1,2}, {4} };
     printarr(arr1);
     cout << endl ;
     printarr(arr2);
     cout << endl ;
     printarr(arr3);
     return 0 ;
}
void printarr(int a[  ][3] )
{
     for(int i=0; i<2; i++)
     {
          for(int j=0; j<3; j++)
               cout << a[ i ][ j ] << ' ';
               cout << endl ;
     }
}
```

خروجی :

```
1    2    3
4    5    6
1    2    3
4    5    0
1    2    0
4    0    0
```

توابع درون خطی (inline)

کلمه inline بدین معنی است که به کامپایلر دستور می‌دهد که یک کپی از دستورالعمل های تابع در همان جا (در زمان مقتضی) تولید نماید تا از احضار تابع ممانعت بعمل آورد.

اشکال توابع inline

بجای داشتن تنها یک کپی از تابع ، چند کپی از دستورالعمل های تابع در برنامه اضافه می‌شود که باعث بزرگ شدن اندازه یا طول برنامه می‌شود. بنابراین از inline برای توابع کوچک استفاده می‌گردد.

مثالی از توابع درون خطی:

```
#include   <iostream >
inline float cube(const float s) {return s*s*s; }
int main( )
{
float side ;
cin >> side ;
cout << side << cube(side) << endl ;
return 0 ;
}
```

انتقال پارامترها از طریق ارجاع

تاکنون وقتی تابعی را احضار می‌کردیم یک کپی از مقادیر آرگومان ها درپارامترهای متناظر قرار می‌گرفت . این روش احضار بوسیله مقدار یا call by value نامیده شد. در انتقال پارامترها از طریق ارجاع در حقیقت حافظه مربوط به آرگومان ها و پارامترهای متناظر بصورت اشتراکی مورد استفاده قرار می‌گیرد. این روش call by reference نامیده می‌شود.

در این روش پارامترهائی که از طریق call by reference عمل می‌نمایند در پیش نمونه تابع قبل از نام چنین پارامترهائی از & استفاده می‌شود. واضح است که در تعریف تابع نیز به همین طریق عمل می‌شود.

```
#include   <iostream>
int vfunct(int);
void rfunct (int &) ;
int main( )
{
     int x=5, y=10;
     cout << x << endl << vfunct(x) << endl << x << endl ;
     cout  << y << endl ;
     rfunct(y) ;
     cout << y << endl ;
     return 0 ;
}
int vfunct(int a)
{
     return a *= a ;
}
void rfunct(int &b)
{
     b *= b ;
}
```

مثال :

x	y
5	10

خروجی :

5
25
5
10

مقدار آرگومان x تغییر نمی کند.

x	y b
5	10 100

ادامه خروجی :

100

نکته: وقتی پارامتری بصورت call by reference اعلان می‌گردد این بدان معنی است که با تغییر مقدار این پارامتر در تابع احضار شده مقدار آرگومان متناظر نیز تغییر می‌نماید.

مثال: برنامه‌زیر با استفاده از fswap دو مقدار اعشاری را مبادله می‌نماید.

```
# include   <iostream >
void fswap(float & , float & );
int main( )
{
float a=5.2, b=4.3;
cout << a << endl << b ;
fswap( a , b) ;
cout << a << endl << b ;
return 0 ;
}
void fswap(float &x , float & y)
{
float t;
t = x ;
x = y ;
y = t ;
}
```

کلاس‌های حافظه (storage classes)

متغیرها به دو طریق متمایز مشخص می‌شوند: یکی بوسیله نوع (type) آن ها و دیگری بوسیله کلاس حافظه آن ها.

نوع متغیر قبلاً اشاره شد. بعنوان مثال int ، float ، double ، . . . ولی کلاس حافظهٔ یک متغیر در مورد طول عمر و وسعت و دامنهٔ متغیر بحث می‌نماید.

بطور کلی کلاس حافظه متغیرها به چهار دستهٔ تقسیم می‌گردد :

۱. automatic
۲. static
۳. external
۴. register

automatic: متغیرهای automatic در درون یک تابع تعریف می‌شوند و در تابعی که اعلان می‌شود بصورت متغیرهای محلی برای آن تابع می‌باشند. حافظه تخصیص داده شده به متغیرهای automatic پس از اتمام اجرای تابع از بین می رود به عبارت دیگر وسعت و دامنهٔ متغیرهای از نوع automatic تابعی می‌باشد که متغیر در آن اعلان گردیده است.

Static: متغیرهای static نیز در درون توابع تعریف می شوند و از نظر وسعت و دامنه شبیه متغیرهای automatic هستند ولی در خاتمهٔ اجرای تابع، حافظه وابسته به این نوع متغیرهـــا از بین نمی‌رود بلکه برای فراخوانی بعدی تابع باقی می‌ماند.

بایستی توجه داشت که اگر در توابع به متغیرهای از نوع static مقدار اولیه تخصیص ندهیم مقدار صفر بصورت اتوماتیک برای آن ها در نظر گرفته می‌شود.

```
#include   <iostream>
// program to calculate successive fibonacci numbers
long int fib(int) ;
int main( )
{
int n ;
cout << " how many fibonacci numbers?" ;
cin >> n ;
cout << endl ;
for(int j=1; j<=n; ++j )
cout << j << "   " << fib(j) << endl ;
return 0 ;
}
long int fib(int count)
{
static long int t1 = 1, t2=1;
long int  t ;
t =(count <3) ?1 : t1 + t2 ;
t2 = t1 ;
t1 = t ;
return(t) ;
}
```

External: متغیرهای از نوع external متغیرهائی هستند که در بیرون از توابع اعلان می شوند و وسعت و دامنه فعالیت آن ها کلیه توابعی می‌باشد که در زیر دستور اعلان متغیر قرار دارد.

مثال:

```
# include   <iostream >
int w; // external variable
functa(int x, int y)
{
cout << w ;
w = x + y ;
cout << endl << w << enld;
return x%y ;
}
```

```
int main( )
{
int a, b, c, d;
cin >> a >> b ;
c=functa(a, b) ;
d=functa(w, b+1);
cout << endl << c << endl << d << endl << w ;
return 0 ;
}
```

بایستی توجه داشت که اگر در توابع به متغیرهای از نوع external مقدار اولیه تخصیص ندهیم مقدار صفر بصورت اتوماتیک برای آن ها در نظر گرفته می‌شود.

Register: وقتی متغیری از نوع register اعلان می‌شود از کامپیوتر عملاً درخواست می‌شود که به جای حافظه از یکی از رجیسترهای موجود استفاده نماید.

کاربرد کلاس register: معمولاً از نوع رجیستر برای شاخص‌های دستور تکرار و یا اندیس های آرایه‌ها استفاده می‌شود. بایستی توجه داشت که متغیرهای از نوع رجیستر قابل استفاده در دستور cin نمی‌باشند.

سربارگذاری توابع (function overloading)

در C++ این امکان وجود دارد که دریک برنامه بتوانیم از چند توابع هم نام استفاده نمائیم مشروط بر این که پارامترهای این توابع متفاوت باشند. (از نظر تعداد پارامتر و یا نوع پارامترها و ترتیب آن ها)

مثال:

```
# include   <iostream >
float addf(float , int);
int addf(int , int);
int main( )
{int a=5, b=10 ;
float d=14.75 ;
cout << addf(a , b) << endl;
cout << addf(d ,  b) << endl;
return 0 ;}

int addf(int x, int y)
{return x+y ;}

float addf(float x, int y)
{return x+y ;}
```

تمرینات:

۱. برنامه ای بنویسید که ۳ عدد اعشاری را خوانده و به تابعی ارسال کند و تابع میانگین آن ها را محاسبه کرده و برگرداند.

۲. برنامه ای که ضرایب معادله ی درجه دومی را خوانده، آن ها را به تابعی ارسال کند. تابع معادله را حل کند و جواب ها را در خروجی چاپ کند. (تابع جواب ها را به تابع اصلی برنمی گرداند.)

۳. برنامه ای بنویسید که عددی را از ورودی خوانده و هریک از ارقام آن را در یک سطر چاپ کند. تفکیک و چاپ ارقام توسط تابع بازگشتی صورت گیرد.

۴. برنامه ای بنویسید که حاصلضرب دو عدد صحیح را به کمک جمع کردن محاسبه کند. برای این کار از تابع بازگشتی استفاده کنید.

$$a\times b = \begin{cases} a & \text{اگر } b=1 \\ a\times(b-1)+a & \text{اگر } b>1 \end{cases}$$

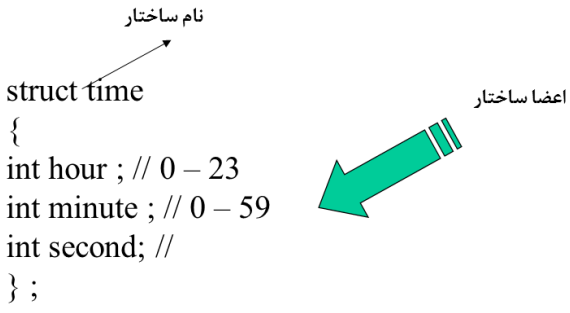
فصل هفتم: ساختار ها

- ساختارها
- Union ها
- رشته ها و توابع مربوطه

ساختارها

ساختارها شبیه آرایه‌ها بوده بدین صورت که یک نوع داده گروهی است که فضای پیوسته از حافظه اصلی را اشغال می‌نماید. اما عناصر ساختار الزاماً از یک نوع نمی‌باشند بلکه اعضای یک ساختار می‌توانند از نوع‌های مختلفه از قبیل char ، int ، float ، ... باشند.

تعریف ساختار:




```
struct account
        {
         int acc_no ;
        char acc_type;
        char name[80] ;
        float balance ;
        };
```

ساختار account دارای چهار عضو می‌باشد:

acc_no	شماره حساب از نوع int
acc_type	نوع حساب از نوع char
name	مشخصات صاحب حساب از نوع رشتهٔ 80 کرکتری
balance	مانده حساب از نوع float

به دو صورت می توان اعلان یک متغیر از نوع ساختار را نمایش داد :

روش دوم	روش اول

struct account {
int acc_no ;
char acc_type;
char name[80];
float balance;
};
account cust1, cust2, cust3;
	struct account {
	int acc_no;
	char acc_type;
	char name[80];
	float balance;
	} cust1, cust2, cust3;

به ساختارها می‌توان مقدار اولیه نیز تخصیص داد:

account cust = {4236, 'r', "Nader Naderi" , 7252.5**};**

دسترسی به عناصر یک ساختار: به منظور دسترسی به عناصر یک ساختار از عملگر . استفاده می‌گردد. عملگر . جزء عملگرهای یکتائی می‌باشد.

مثال:

cust .acc_no = 4236
cust .acc_type = 'r'
cust . name = "Nader Naderi"
cust . balance = 7252.5

نکته: عضو یک ساختار خود می‌تواند یک ساختار دیگر باشد.

```
struct date {
        int month;
        int day;
        int year;
        };
struct account {
        int acc_no ;
        char acc_typer;
        char name[80];
        float balance ;
        date lastpay ;
        };
```

اگرداشته باشیم

```
account x, y ;
```

آنگاه عضو lastpay بوسیله

```
x.lastpay.day
x.lastpay.month
x.lastpay.year
```

مشخص می‌گردد.

نکته: می‌توان آرایه‌ای تعریف نمود که هر عضو آن یک ساختار باشد و حتی به آن ها مقادیر اولیه تخصیص نمود.

```
struct struc1 {
        char name[40];
        int pay1;
        int pay2;
        } ;
struc1 cust[ ]= {"nader", 3000 , 40000,
                 "sara", 4200, 6000,
                 "susan", 3700, 25000,
                  "saman", 4800 , 2000, };
```

برنامه زیر هر عدد مختلط را بصورت یک ساختار در نظر گرفته، دو عدد مختلط را می‌گیرد و مجموع آن ها را مشخص و نمایش می‌دهد.

```
#include  <iostream>
int main( )
{
struct complex{
                float  a;
                float  b; } x, y, z;
cout << "enter 2 complex numbers" << endl ;
cin >> x.a>>x.b ;
cout << endl;
cin >> y.a >> y.b;
z.a = x.a + y.a ;
z.b = x.b + y.b ;
cout << endl << z.a << "  "<< z.b;
return 0 ;
}
```

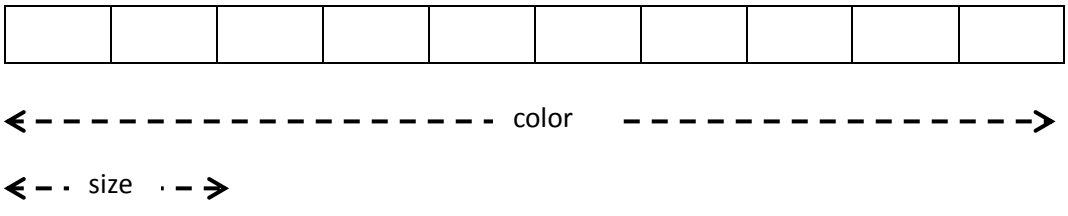
Union

union از نظر ساختاری شبیه struct می‌باشد. با این تفاوت که عضوهائی که تشکیل union می دهد همگی از حافظه مشترکی در کامپیوتر استفاده می‌نمایند. تمام اعضای داده ای در یونیون از یک محل شروع می شوند.

بنابراین استفاده از union باعث صرفه‌جوئی در حافظه می‌گردد.

مثال:

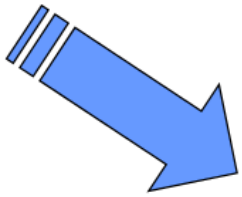
```
union id
        {
         char color [10];
         int size;
        } x , y;
```



هر کدام از متغیرهای x و y یک رشته 10 کرکتری یا یک مقدار از نوع int می‌باشد و کامپیوتر یک بلوک حافظه که بتواند رشته ۱۰ کرکتری را در خود جای دهد ، برای color و size در نظر می‌گیرد.

مثال:

```
union xpq
 {
int x ;
char y[2] ;
} p ;
```



x

بایت دوم	بایت اول
y[1]	y[0]

رشته‌ها و توابع مربوطه

رشته‌ها در C++ ، آرایه‌ای از کرکترها می‌باشند که با کرکتر '\0' ختم می شوند.

char name[ ]= " sara";

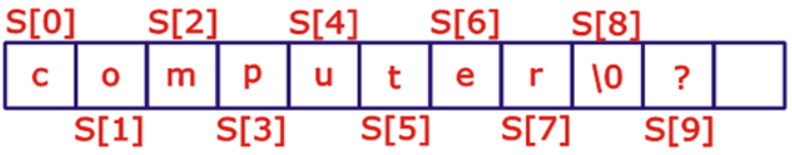
s
a
r
a
\0

نحوه رفتار با رشته‌ها همانند رفتار با آرایه‌ها می‌باشد ولی یک تفاوت اساسی در انتهای رشته‌ها می‌باشد که یک علامت نشان‌دهنده انتهای رشته‌ها می‌باشد ولی در آرایه‌ها هیچ علامتی انتهای آرایه‌ها را مشخص نمی‌کرد.

علامت '\0' انتهای رشته‌ها مشخص می‌کند. مثلاً در دستور

$char \quad S[10];$

و اگر رشته را با 'computer' مشخص نماییم به صورت زیر در حافظه جا می‌گیرد.



مقداردهی اولیه به رشته ها

```
char S1[ ] = "IT";

char S2[10] = "is";

char S3[ ] = {'N','e','w',' ','L','i','v','e','\0'};
```

اگر همانند دو دستور اول عمل کنیم، خود کامپایلر اتوماتیک در آخر رشته '\0' قرار می دهد ولی در حالت **سوم** باید خودمان به رشته '\0' بدهیم.

نکته: رشته ها همیشه به خاطر '\0' یک کاراکتر بیشتر فضا می گیرند.

خواندن و نوشتن رشته:

برای خواندن رشته ها از ورودی طبق رویه زیر عمل می کنیم.

```
char str[51];

cin >> str;

cout << "string is :" << str;
```

در این مثال یک رشته با طول ۵۰ فرض کردیم (یک کارکتر بخاطر '\0' در نظر نگرفتیم) و بوسیله *cin* از ورودی خواندیم و با تابع *cout* در خروجی چاپ نمودیم.

کاربر اگر کلید *Enter* را در مثال فشار دهد رشته *str* مقدار می‌گیرد یا اگر *space* یا *tab* را فشار دهد دوباره *cin* مقدار می‌گیرد.

استفاده از متد ( )get

تابع *get* را می‌توان به شکل زیر بکار برد.

cin.get ((['جداکننده'] و طول رشته و نام رشته)

قسمت 'جداکننده' اختیاری می‌باشد و می‌تواند نباشد. این قسمت یک کارکتر را به عنوان انتهای رشته مشخص می‌کند.

طول رشته: طول حداکثر مکانی است که رشته می تواند بگیرد.

مثلاً:

```
char str[30];
cin.get(str,10);
cin.get(str , 10 , '.');
```

در دستور دوم حداکثر می‌توانیم با طول ۱۰ به *str* از ورودی مقدار دهیم و دستور سوم بجای *Enter* که آخر رشته باشد با '.' مشخص می‌شود.

تفاوت get و cin

همانگونه که گفته شد انتهای ورودی دستور *cin* را یا *Enter* مشخص می‌کند یا *space* ولی در *cin* می‌توانیم فاصله و *tab* را نیز به رشته بدهیم. مثلاً

```
char S[101];
cin.get(S,50);
cin >> S;
```

اگر کاربر رشته "*IT is now live*" را به برنامه بدهد تابع ( ) *get* همه را در *C* جا می‌دهد ولی *cin* فقط "*IT*" را در *S* قرار می‌دهد.

انتساب رشته ها:

دو رشته را نمی توان بصورت مستقیم در داخل یکدیگر بریزیم.

دستور $strcpy(S1,S2)$ مقدار $S2$ را در داخل $S1$ کپی می‌نماید.

```
char S1[ ]="computer";
char S2[20];
strcpy(S2,S1);
```

مقایسه ی رشته ها:

برای مقایسه رشته نمی‌توانیم به این صورت عمل نماییم. $if(S1==S2)$

چون $S2,S1$ اشاره‌گر به رشته می‌باشند ولی دستوری بنام $strcmp(S1,S2)$ وجود دارد که دو رشته $S2,S1$ را مقایسه می‌نماید.

$S_1 \mid S_2$	$S_1 = S_2$	$S_1 > S_2$	$S_1 < S_2$
$strcmp(S_1,S_2)$	۰	عدد مثبت	عدد منفی

عدد برگردانده شده از تابع $strcpy$ طبق جدول بالا می‌باشد. منظور از اینکه دو رشته از هم کوچکتر باشد یا بزرگتر به این صورت می‌باشد که سمت چپ‌ترین کارکتر را با کد اسکی مقایسه می‌کند اگر بزرگتر بود که آن رشته از رشته دوم بزرگتر می‌باشد. اگر کوچکتر بود، آن رشته کوچکتر می‌باشد و اگر مساوی بود کارکتر بعدی را نگاه می‌کند.

الحاق دو رشته:

برای الحاق دو رشته از تابع $strcat(S1,S2)$ استفاده می‌نماییم. این تابع $S2$ را به انتهای $S1$ وصل می‌نماید.

مثال: برنامه ذیل پنج اسم را بصورت 5 رشته در نظر گرفته آن ها را به ترتیب حروف الفباء مرتب نموده نمایش می‌دهد.

```
using namespace std;
#include <string.h>
#include <conio.h>
#include <iostream>

void sort(char  [    ][10 ]);
int main( )
{
char name[5][10] = {"sara", "afsaneh", "babak", "saman", "naser" };
sort(name);// display sorted strings
for(int i=0; i<5; ++i)
        cout << name[ i ] << endl;
getch();
return 0;
}
void sort(char  name[ 5][ 10])
{
char  t[10];
for(int i=4; i>0;i--)
for(int j=0; j<i; j++)
if(strcmpi(name[j], name[j+1])> 0)
         {// interchange  the two strings
        strcpy(t, name[j]);
        strcpy(name[j], name[j+1]);
        strcpy(name[j+1], t);
       }
return  ;
}
```

تابع strcmpi(s1, s2)

رشته‌های s1 و s2 را با هم مقایسه نموده (بدون توجه به حروف کوچک و بزرگ) اگر رشته s1 برابر با رشته s2 باشد مقدار صفر و اگر رشته s1 کوچکتر از رشته s2 باشد یک مقدار منفی در غیر اینصورت یک مقدار مثبت بر می‌گرداند.

```
char s1[10] = "ALI";
char s2[10]="ali";
cout << strcmpi(s1, s2) << endl;
```

تابع strcmp(s1, s2)

رشته‌های s1 و s2 را با هم مقایسه نموده اگر s1 برابر با s2 باشد مقدار صفر و اگر رشته s1 کوچکتر از رشته s2 باشد یک مقدار منفی در غیر اینصورت یک مقدار مثبت برمی‌گرداند.

```
char s1[10]= "ALI";
char s2[10]="ali";
cout << strcmp(s1, s2) << endl;
```

تابع strncmp(s1, s2,n)

حداکثر n کرکتر از رشتهٔ s1 را با n کرکتر از رشتهٔ s2 مقایسه نموده در صورتیکه s1 کوچکتر از s2 باشد یک مقدار منفی، اگر s1 مساوی با s2 باشد مقدار صفر در غیر اینصورت یک مقدار مثبت برمی گرداند.

```
char s1[10]= "ali reza";
char s2[10]="ali";
cout << strncmp(s1, s2,3) << endl;
```


```
char s1[10]= "ali reza";
char s2[10]="ali";
cout << strncmp(s1, s2,4) << endl;
```

تابع strcat(s1, s2)

دو رشتهٔ s1 و s2 را بعنوان آرگومان گرفته رشتهٔ s2 را به انتهای رشتهٔ s1 اضافه می‌نماید. کرکتر اول رشتهٔ s2 روی کرکتر پایانی '\0' رشتهٔ s1 نوشته می‌شود ونهایتاً رشتهٔ s1 را برمی گرداند.

```
char s1[20]= "ali ";
char s2[20]="reza";
cout << strcat(s1, s2) << endl;
```

```
ali reza
```

تابع strncat(s1, s2,n)

دو رشته s1 و s2 ومقدار صحیح و مثبت n را بعنوان آرگومان گرفته، حداکثر n کرکتر از رشتهٔ s2 را در انتهای رشتهٔ s1 کپی می‌نماید. اولین کرکتر رشته s2 روی کرکتر پایانی '\0' رشتهٔ s1 می‌نویسد ونهایتاً مقدار رشتهٔ s1 را برمی گرداند.

```
char s1[20]= "ali ";
char s2[20]="reza";
cout << strncat(s1, s2,2) << endl;
```

```
ali re
```

تابع strlen(s)

رشتهٔ s را بعنوان آرگومان گرفته طول رشته را مشخص می‌نماید.

```
char s1[10]= "ali";
cout << strlen(s1);
```

```
3
```

```
char s1[10]= "ali ";
cout << strlen(s1);
```

```
4
```

تابع strcpy(s1,s2)

دو رشتهٔ s1 و s2 را بعنوان آرگومان گرفته رشتهٔ s2 را در رشتهٔ s1 کپی می‌نماید و نهایتاً مقدار رشتهٔ s1 را بر می‌گرداند.

```
char s1[20]= "ali ";
char s2[10]="reza";
cout << strcpy(s1, s2) << endl;
```

reza

```
char s1[20];
char s2[10]="ali";
cout << strcpy(s1, s2) << endl;
```

ali

تابع strncpy(s1, s2,n)

دو رشتهٔ s2 , s1 و مقدار صحیح و مثبت n را بعنوان آرگومان گرفته، حداکثر n کرکتر را از رشتهٔ s2 در رشتهٔ s1 کپی نموده، نهایتاً مقدار رشتهٔ s1 را برمی گرداند.

```
char s1[20]= "ali ";
char s2[10]="reza";
cout << strncpy(s1, s2,3) << endl;
```

rez

```
char s1[20];
char s2[10]="amir ali";
cout << strncpy(s1, s2,4) << endl;
```

amir

نکته: برای استفاده از توابع مربوط به رشته‌ها بایستی حتماً در ابتدا برنامه #include <string.h> را قرار دهیم.

مثال:

```
#include      <iostream >
#include      <string.h>
#include      <conio.h>
int main( )
{
char s1[30]= ″happy birthday″;
char  s2[30]= ″happy holidays ″;
cout << strcmp(s1, s2) << endl;
cout << strncmp(s1, s2, 7) << endl ;
return 0;
 }
```

مثال:

```
#include      <iostream >
#include      <string.h>
#include      <conio.h>
int main( )
{
char  s[10] = ″sara″;
cout << strlen(s);
return 0;
}
```

تابع زیر معادل تابع کتابخانه strcmp می باشد.

```
int nikstrcmp(char s[] , char t[] )
{
int i=0;
while (s[i]==t[i] )
    if ( s[i++]='\0' )
            return 0;
return (s[i]-t[i]);
}
```