

به نام خدا

دانشگاه صنعتی بیرجند

گروه مهندسی کامپیوتر و فناوری اطلاعات

جزوه درس مبانی کامپیوتر و برنامه نویسی

وحیده بابائیان

vahidebabaiyan@yahoo.com

vahidebabaiyan.blogfa.com

❖ جایگاه این درس در رشته مهندسی کامپیوتر و مهندسی IT

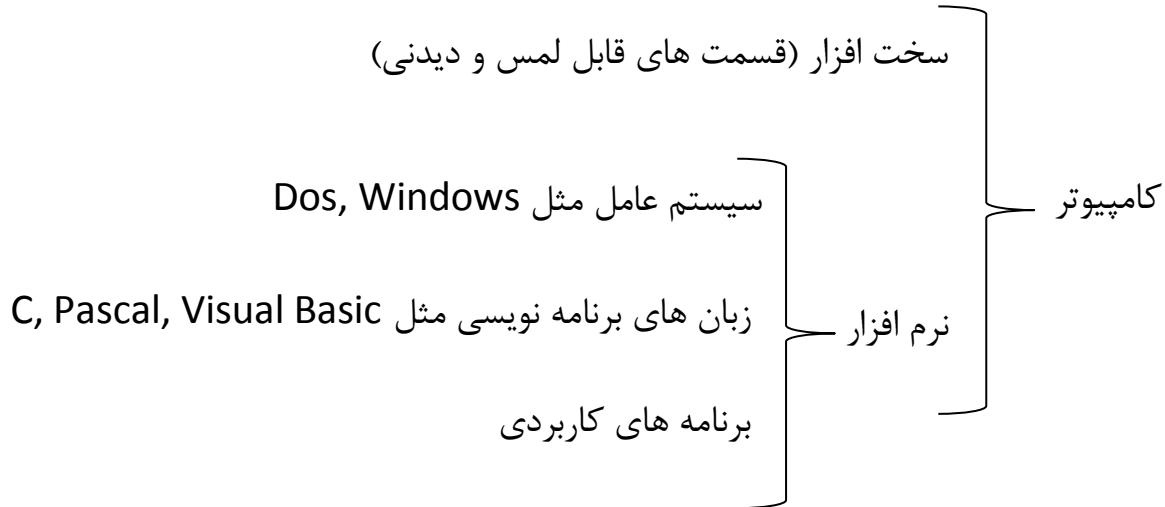
این درس اولین درس دانشگاهی رشته می باشد و نقطه شروعی برای ورود به دنیای جالب برنامه نویسی و علم و فن کامپیوتر هست . بنابراین یاد گیری اصول اولیه برنامه نویسی در این درس از جایگاه ویژه ای برخوردار است.

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 ساخت 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 می‌تواند مقادیر ۰ و ۱ را داشته باشد.

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

۱ کیلو بايت: ۱۰۲۴ بايت

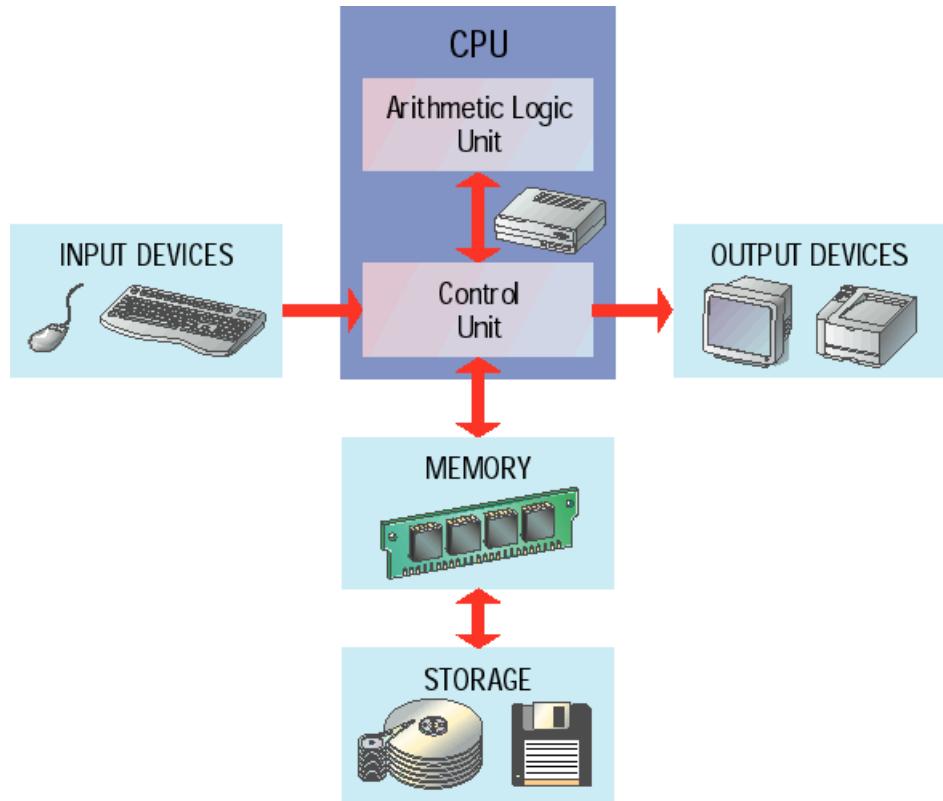
۱ مگا بايت: ۱۰۲۴ کیلو بايت

۱ گیگا بايت: ۱۰۲۴ مگا بايت

اجزای اصلی یک کامپیوتر

هر سیستم کامپیوتری از چهار قسمت اصلی تشکیل می‌شود:

- ۱) واحد پردازش مرکزی (CPU)
- ۲) واحد ورودی (Input)
- ۳) واحد خروجی (Output)
- ۴) حافظه



واحد پردازش مرکزی

ریز پردازنده، تراشه‌ی 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 می‌شود. نام دیگر آن Central Processing Unit می‌باشد.



واحد ورودی

واحدی که داده‌ها را از دستگاه‌های ورودی دریافت کرده و جهت تبدیل آن‌ها به داده‌های قابل فهم کامپیوتر، آن را به واحد کنترل تحویل می‌دهد. قابل ذکر است که واحد کنترل بخشی از CPU می‌باشد. دستگاه‌های ورودی شامل موس، کیبورد، اسکنر، وسایل بازی و هستند.



**CREATIVE
MOUSE**
Optical 3000

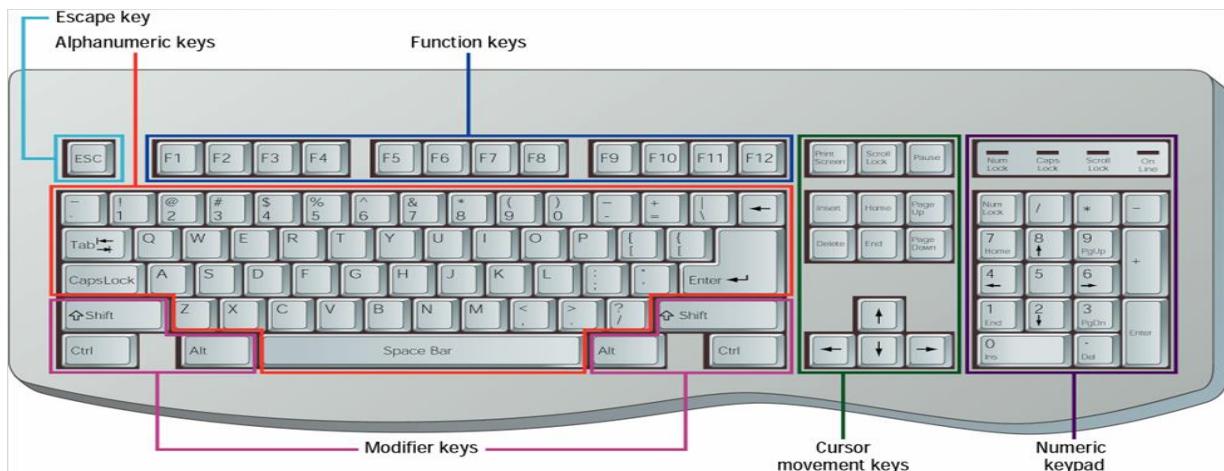
Exceptional Precision and Style at
an Attractive Price



Mouse



Keyboard



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 ای حافظه های اصلی (Main Memory) و حافظه های جانبی یا ثانویه (Secondary Memory) تقسیم می شوند. هر برنامه برای اجرا ابتدا بايستی در حافظه ای اصلی قرار بگیرد و سپس توسط CPU اجرا شود. پس از اجرای برنامه، برای ذخیره اطلاعات برنامه، آن ها را روی حافظه ای جانبی ذخیره می کنند.

وب سایت

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

مروگر 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 مروگر وب Mosaic بود که در سال ۱۹۹۳ شروع به کار کرد.

از انواع دیگر مروگرهای 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 ایمیل پیام خود را کمتر از یک ثانیه به تمام دنیا بفرستیم. صدا، تصویر و فایل های مختلفی می توانند به نامه ای ما پیوست شوند. برخلاف سیستم تلفن، این سیستم یک سیستم افلاین می باشد. یعنی در زمان ارسال نامه لازم نیست طرف مقابل پشت سیستم خود باشد.

موتور جستجو

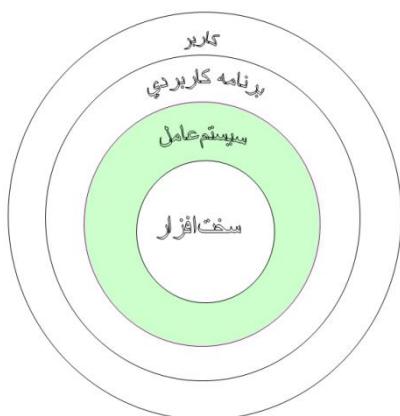
از این وسیله برای جستجوی یک تاپیک یا موضوع خاص در اینترنت استفاده می کنیم. این موتورهای جستجو از طریق مروگرهای وب در دسترس خواهند بود. هر کلمه کلیدی یا عبارتی که وارد کنید با مجموعه ای از نتایج روبرو خواهد شد. در ادامه مثال هایی از چند موتور جستجو اورده شده است:

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 برنامه نویسی

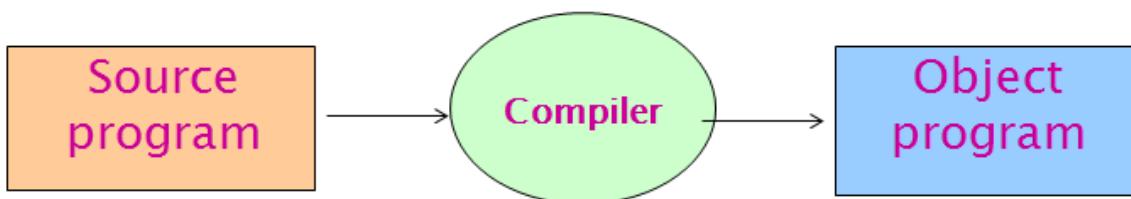
نرم افزارها توسط زبان های برنامه نویسی نوشته می شوند. زبان های برنامه نویسی، یک سیستم ارتباطی هستند که توسط آن ها می توان دستورات لازم را به ماشین انتقال داد.

هر زبان برنامه نویسی به مجموعه ای از علایم، قواعد و دستورالعمل ها گفته می شود که امکان ارتباط با کامپیوتر را جهت بیان کاری یا حل مسئله ای فراهم می کند.

در حالت کلی زبان های برنامه نویسی را به سه دسته زیر تقسیم بندی می کنند:

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

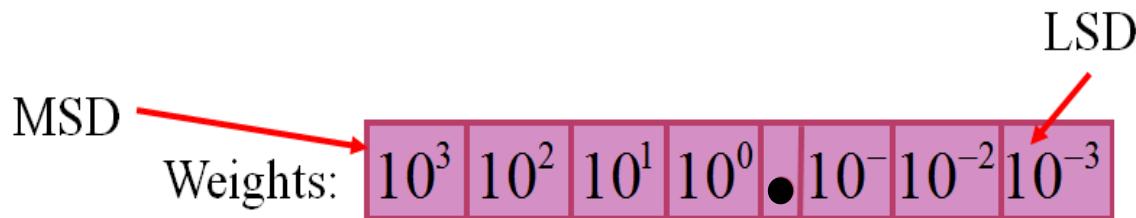
کامپایلر برنامه نوشتہ در یک زبان سطح بالا را به برنامه مقصد تبدیل می کند.



جلسه دوم: آشنایی با سیستم اعداد

مرور سیستم دهددهی

- پایه ۱۰ است و ارقام $0, 1, \dots, 9$ می باشند.
- برای اعداد بزرگتر از ۹، یک رقم با اهمیت تر به سمت چپ اضافه کنید. مثلا: $19 < 193$
- هر محل دارای یک وزن است:



- به عنوان مثال عدد ۱۹۳۶.۲۵ را می توان به صورت زیر نمایش داد:

$$1 \times 10^3 + 9 \times 10^2 + 3 \times 10^1 + 6 \times 10^0 + 2 \times 10^{-1} + 5 \times 10^{-2}$$

سیستم عدد نویسی دودویی

- پایه ۲ است و ارقام $0, 1$ هستند.
- برای اعداد بزرگتر از ۱، یک رقم با اهمیت تر به سمت چپ اضافه کنید. مثلا: $1 < 1011_01$
- هر محل دارای یک وزن است:



- به عنوان مثال عدد 1011_01 را می توان به صورت زیر محاسبه کرد:

$$\begin{aligned}1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} = \\= 1 \times 16 + 0 \times 8 + 1 \times 4 + 1 \times 2 + 1 \times 1 + 0 \times 0.5 + 1 \times 0.25 = 23.25\end{aligned}$$

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

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

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

$$(1011001)_2 = 1 \times 2^0 + 0 \times 2^1 + 0 \times 2^2 + 1 \times 2^3 + 1 \times 2^4 + 0 \times 2^5 + 1 \times 2^6$$

$$= 1 + 0 + 0 + 8 + 16 + 0 + 64 = 89$$

سیستم عدد نویسی دودویی

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

جواب: ۴۸,۴۳۷۵

در دنیای کامپیوتر:

$1024 = 2^{10}$ با K (کیلو) نشان داده می شود. •

$1048576 = 2^{20}$ با M (مگا) نشان داده می شود. •

2^{30} G (گیگا) •

2^{40} T (تر) •

چه تعداد بیت در یک حافظه 16GByte وجود دارد؟

مبناهای ۸ و ۱۶

- مبنای ۸ -

پایه ۸ است و رقم‌ها ۰, ۱, ۲, ۳, ۴, ۵, ۶, ۷ هستند -

$$(236.4)_8 = (158.5)_{10} -$$

$$2 \times 8^2 + 3 \times 8^1 + 6 \times 8^0 + 4 \times 8^{-1} = 158.5 -$$

- مبنای ۱۶ -

پایه ۱۶ است و رقم‌های ۰, ۱, ۲, ۳, ۴, ۵, ۶, ۷, ۸, ۹ از سیستم دهدهی قرض گرفته شده‌اند و از A, B, C, D, E, F

به ترتیب برای نمایش رقم‌های ۱۰, ۱۱, ۱۲, ۱۳, ۱۴, ۱۵ استفاده می‌گردد.

$$(D63FA)_{16} = (877562)_{10} -$$

$$13 \times 16^4 + 6 \times 16^3 + 3 \times 16^2 + 15 \times 16^1 + 10 \times 16^0 = 877562$$

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:

$$\begin{array}{r}
 54 \quad | \quad 2 \\
 54 \quad | \quad 27 \quad | \quad 2 \\
 \underline{\textcircled{0}} \quad \underline{\textcircled{1}} \quad \underline{\textcircled{1}} \quad \underline{\textcircled{0}} \quad \underline{\textcircled{1}} \\
 26 \quad | \quad 13 \quad | \quad 2 \\
 \underline{\textcircled{1}} \quad \underline{\textcircled{0}} \quad \underline{\textcircled{0}} \quad \underline{\textcircled{1}} \\
 12 \quad | \quad 6 \quad | \quad 2 \\
 \underline{\textcircled{1}} \quad \underline{\textcircled{0}} \quad \underline{\textcircled{1}} \\
 6 \quad | \quad 3 \quad | \quad 2 \\
 \underline{\textcircled{0}} \quad \underline{\textcircled{1}} \quad \underline{\textcircled{1}} \\
 2 \quad | \quad 1 \\
 \underline{\textcircled{1}}
 \end{array} \qquad 54 = (110110)_2$$

$$\begin{array}{r}
 13 \quad | \quad 2 \\
 12 \quad | \quad 6 \quad | \quad 2 \\
 \underline{\textcircled{1}} \quad \underline{\textcircled{0}} \quad \underline{\textcircled{1}} \\
 6 \quad | \quad 3 \quad | \quad 2 \\
 \underline{\textcircled{0}} \quad \underline{\textcircled{1}} \\
 2 \quad | \quad 1 \\
 \underline{\textcircled{1}}
 \end{array} \qquad 13 = (1101)_2$$

جمع دودویی

$\frac{1}{1}$	$+$	$\frac{1}{1}$
$\underline{0}$		$\underline{0}$
1	0	

$\frac{1}{0}$	$+$	$\frac{1}{1}$
$\underline{1}$		$\underline{1}$
1		0

$\frac{0}{1}$	$+$	$\frac{1}{1}$
$\underline{1}$		$\underline{1}$
0		1

$\frac{0}{0}$	$+$	$\frac{0}{0}$
$\underline{0}$		$\underline{0}$
0		0

در جمع آخر، ۱، رقمه نقلی است که با بیتهای بعدی جمع می شود.

مثال : جمع زیر را در مبنای ۲ انجام دهید.

$$\begin{array}{r}
 29 = (00011101)_2 \quad 29 \quad | \quad 1 \quad | \quad 1 \\
 17 = (00010001)_2 \quad \underline{17} \quad | \quad ? \quad | \quad 0 \quad 0 \quad 0 \quad 1 \\
 \hline
 (00101110)_2
 \end{array}$$

$$(00101110) = 2 + 4 + 8 + 32 = 46$$

اعتبار نتیجه

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 مثلاً 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 کاملاً 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
- خروجی‌ها

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

<u>ورودی‌ها</u>	<u>انجام محاسبات</u>	<u>خروجی‌ها</u>
a , b	جمع دو عدد	مجموع دو عدد

۰_ شروع
۱- a, b را بخوان.
۲- مجموع a , b را محاسبه و در sum قرار بده.
۳- sum را در خروجی چاپ کن
۴- پایان

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

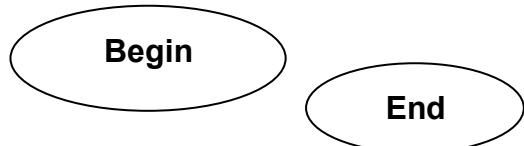
<u>ورودی‌ها</u>	<u>انجام محاسبات</u>	<u>خروجی‌ها</u>
a , b , c	محاسبه مجموع محاسبه میانگین	چاپ مجموع چاپ میانگین

۰_ شروع
۱- سه عدد از ورودی بخوان
۲- مجموع سه عدد را محاسبه و در sum قرار بده.
۳- sum را بر سه تقسیم کرده، در ave قرار بده.
۴- ave , sum را در خروجی چاپ کن.
۵- پایان.

فلوچارت

معمولًاً 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.

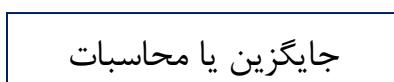
۱. علامت‌های شروع و پایان: که معمولاً از یک بیضی استفاده می‌کنند:



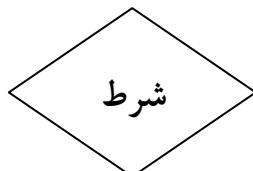
۲. علامت‌های ورودی و خروجی: که معمولاً از متوازی‌الاضلاع استفاده می‌شود:



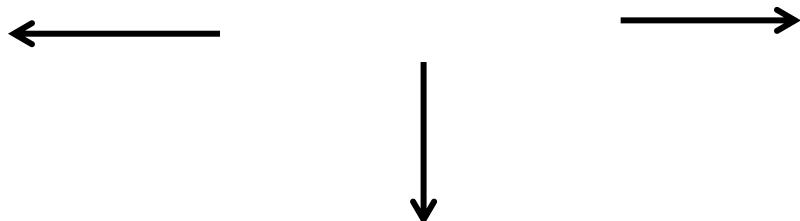
۳. علامت‌های محاسباتی و جایگزینی: برای نمایش دستورات جایگزینی و محاسباتی از مستطیل استفاده می‌کنند:



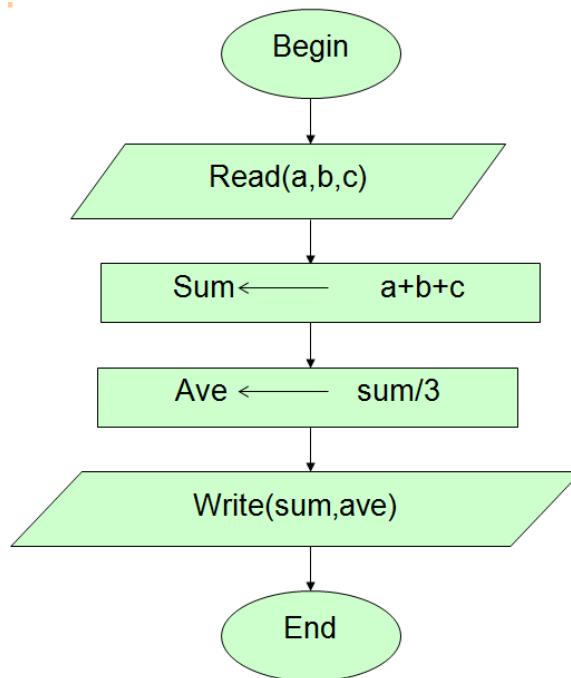
۴. علامت شرط: برای نمایش شرط از لوزی استفاده می‌شود.



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

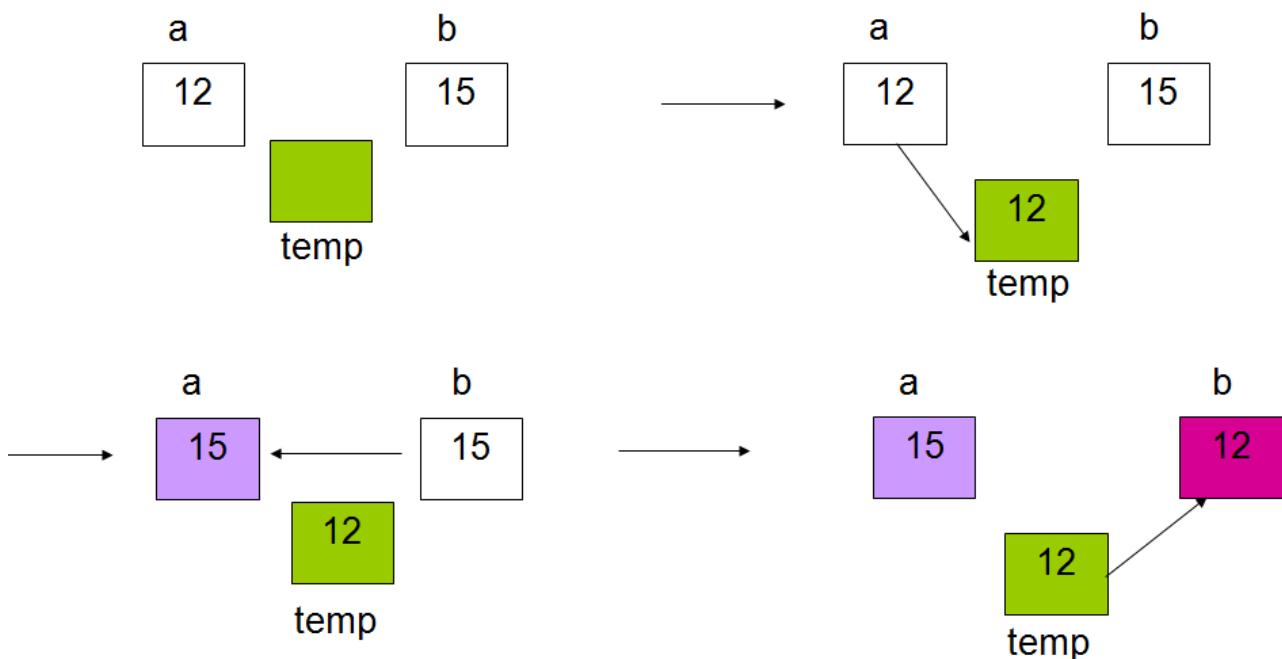


مثال «فلوچارت مجموع سه عدد

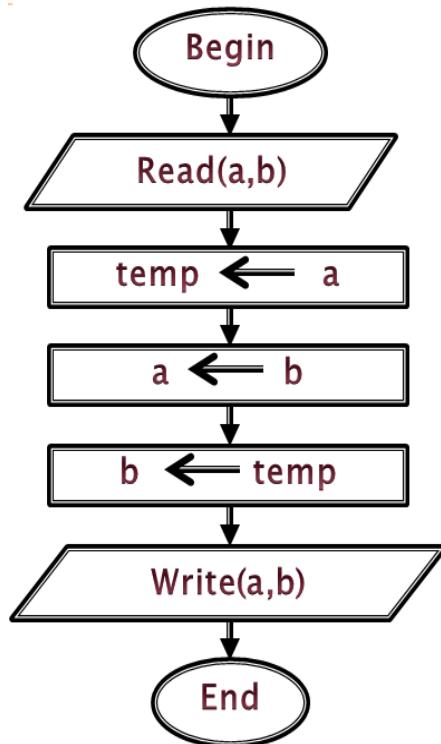


مثال: فلوچارتی رسم نمایید که دو عدد از ورودی دریافت کرده سپس محتويات دو عدد را با هم جابجا نماید.

برای حل این مسئله a , b را دو متغیر که در آن ها دو عدد خوانده شده، قرار می گیرد در نظر می گیریم. سپس با استفاده از یک متغیر کمکی محتويات این دو عدد را جابجا می کنیم :



فلوچارت مسئله بالا بصورت زیر خواهد بود:



تمرین

- ۱- فلوچارتی رسم نمایید که طول و عرض مستطیل را از ورودی دریافت کرده محیط و مساحت آنرا محاسبه و چاپ کند.
- ۲- فلوچارتی رسم نمایید که شعاع دایره‌ای را از ورودی دریافت کرده، محیط و مساحت آنرا محاسبه و چاپ نماید.
- ۳- فلوچارتی رسم کنید که سه عدد first , second, third را از ورودی دریافت کرده، محتويات آن‌ها را جابجا نموده، حاصل را در خروجی چاپ کند.
- ۴- فلوچارتی رسم نمایید که دو عدد از ورودی دریافت کرده، سپس محتويات دو عدد را بدون استفاده از متغیر کمکی جابجا کند.
- ۵- فلوچارتی رسم نمایید که عددی (درجه حرارت بر حسب سانتیگراد) را از ورودی دریافت کرده سپس آن را به درجه فارنهایت تبدیل کند.

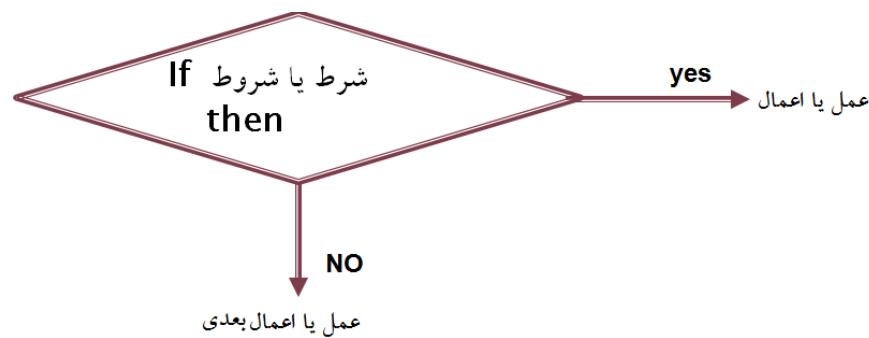
$$f = \frac{9}{5} c + 32$$

دستورالعمل‌های شرطی

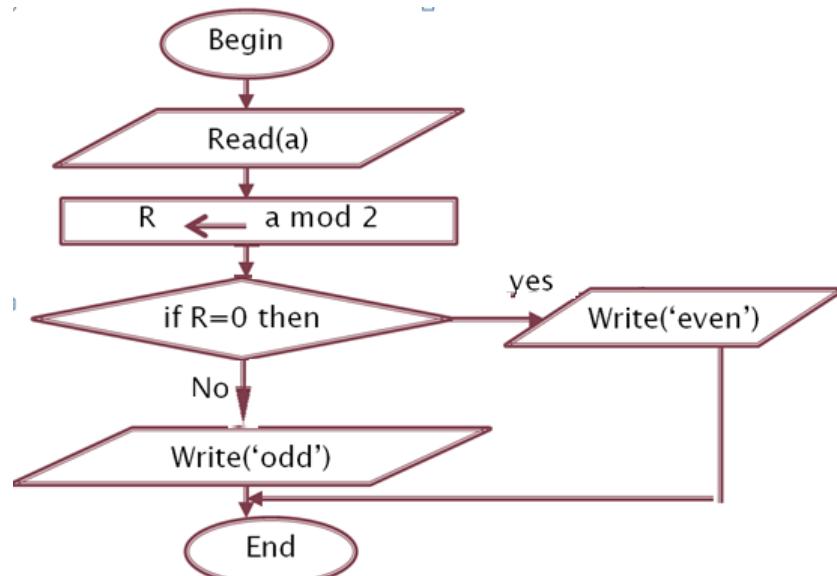
در حل بسیاری از مسائل یا تقریباً تمام مسائل نیاز به استفاده از شروط جزء، نیازهای اساسی محسوب می‌شود. همانطور که ما خودمان در زندگی روزمره با این شرط‌ها سرکار داریم. بطور مثال اگر هوا ابری باشد ممکن است چنین سخن بگوییم:

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

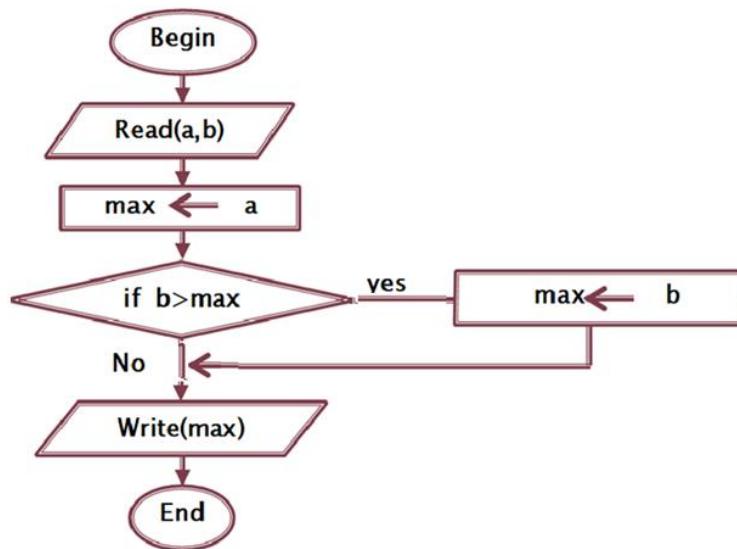
در حالت کلی شرط را بصورت زیر نمایش می‌دهند:



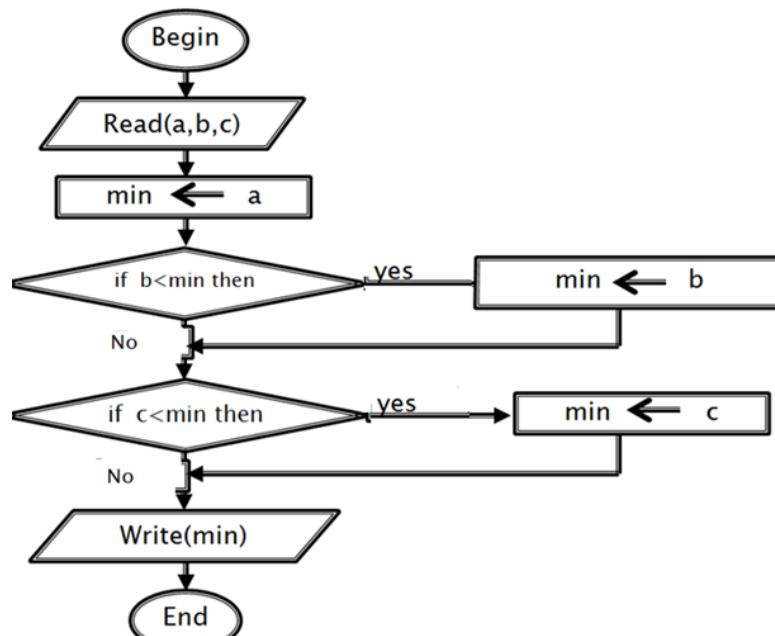
مثال : فلوچارتی رسم نمایید که عددی را از ورودی دریافت کرده، فرد یا زوج بودن آن را تشخیص دهد.



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



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



نمونه اجرای فلوچارت بالا بصورت زیر می باشد:

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

- ۱- فلوچارتی رسم کنید که عددی را از ورودی دریافت کرده، قدر مطلق عدد را در خروجی چاپ کند.
- ۲- فلوچارتی رسم نمایید 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 نمایید که عددی را از ورودی دریافت کرده، بخش پذیری آن بر ۳ و ۵ را بررسی نماید.
- ۴- فلوچارتی رسم نمایید 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.

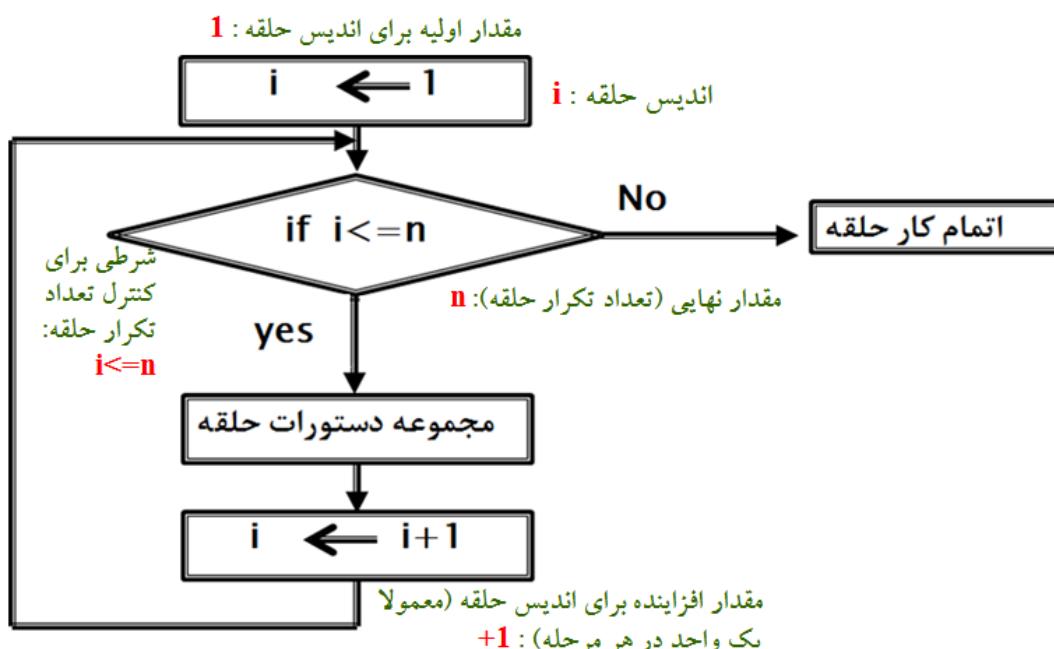
أنواع حلقة ها

۱. حلقه‌های با تکرار مشخص
۲. حلقه‌های با تکرار نا مشخص

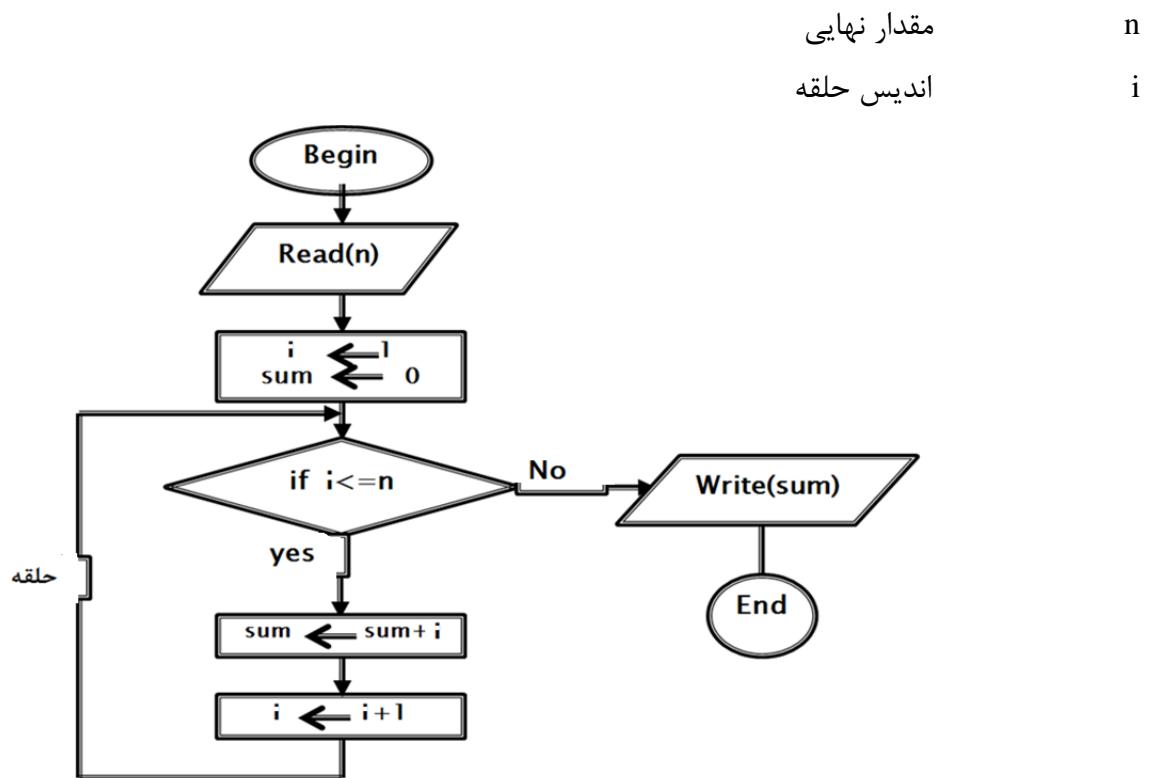
حلقه‌های با تکرار مشخص: در این نوع حلقه‌ها تعداد تکرار مشخص می‌باشد این حلقه از اجزاء زیر تشکیل می‌شود:

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

این حلقه‌ها را غالباً با فلوچارت بصورت زیر نمایش می‌دهند:



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

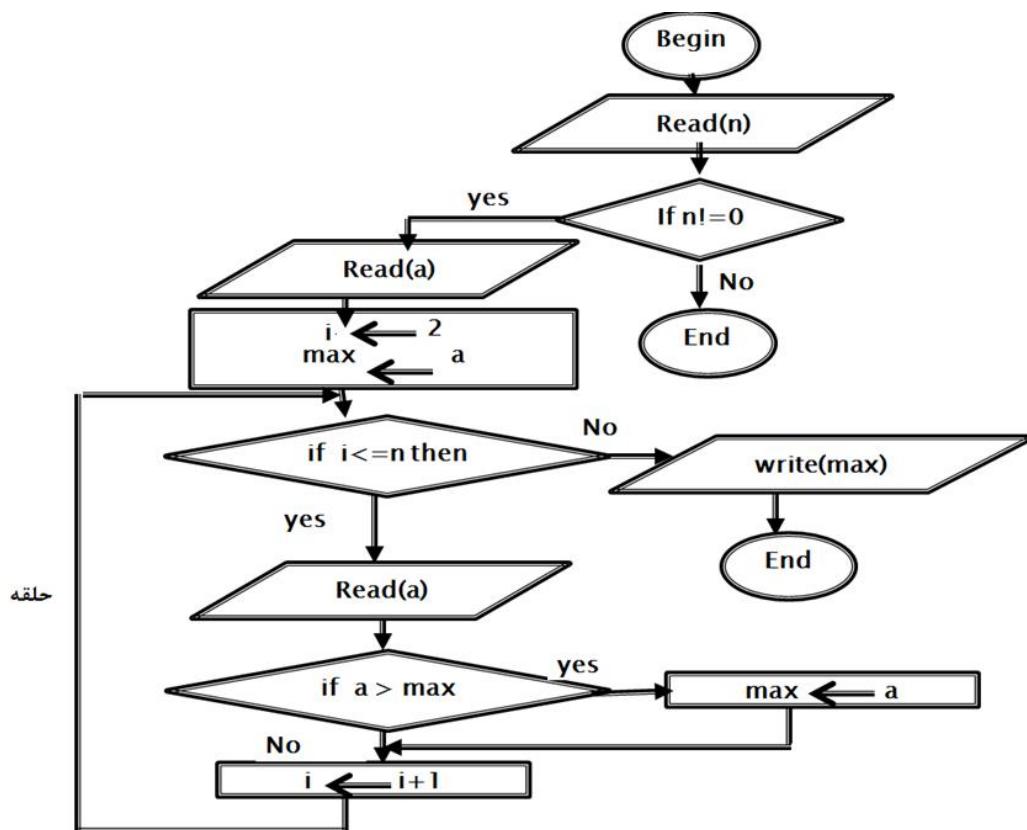


نمونه اجرای فلوچارت بالا بصورت زیر است:

	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 عدد از ورودی دریافت کرده، بزرگترین مقدار از بین n عدد را پیدا کرده در خروجی چاپ نماید.

i
اندیس حلقه
n
مقدار نهایی
Max
بزرگترین مقدار



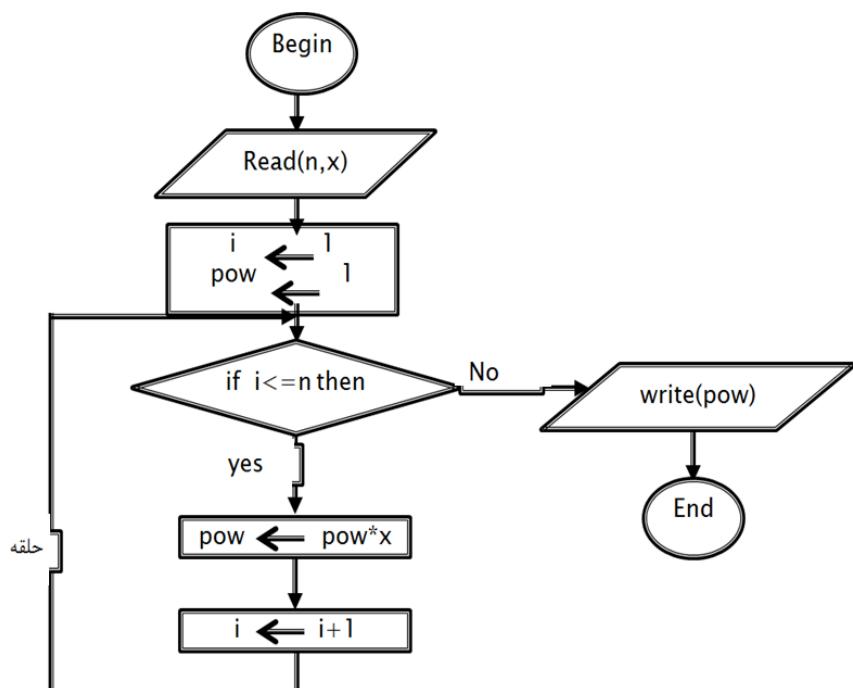
مثال : فلوچارتی رسم نمایید که x ، n ، دو عدد صحیح مثبت را از ورودی دریافت کرده سپس x به توان n را محاسبه کند.

i اندیس حلقه

n مقدار نهایی

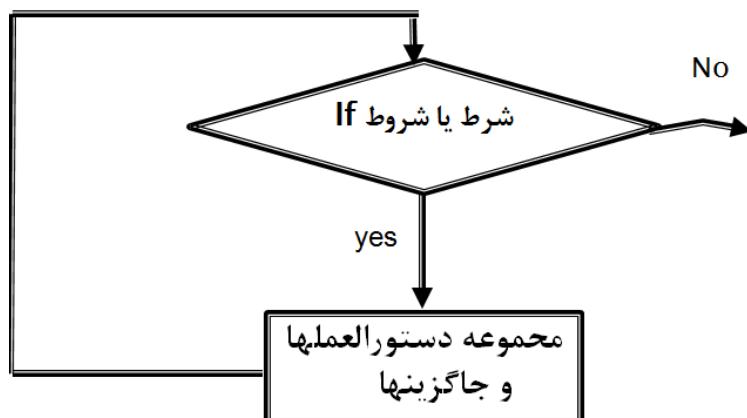
عدد به توان n

pow

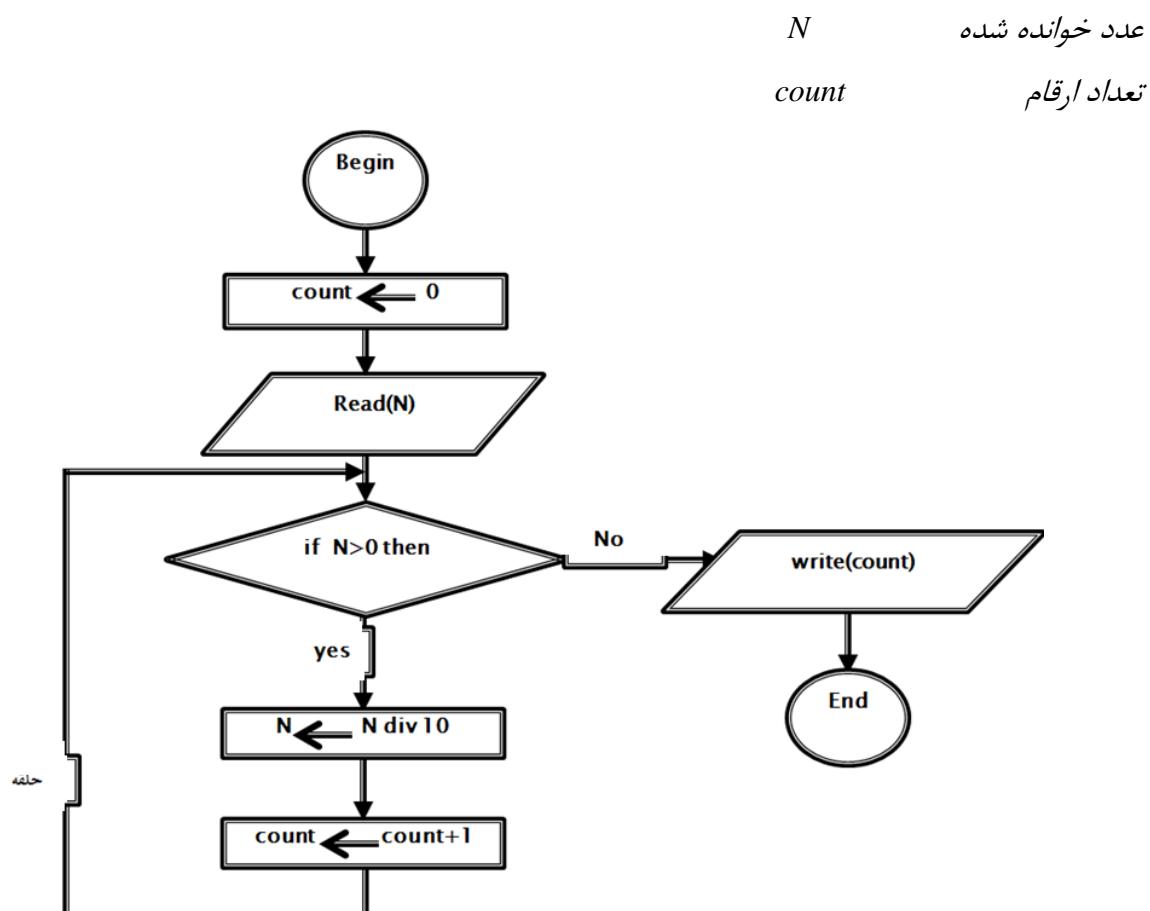


حلقه‌هایی که تعداد تکرار آن‌ها مشخص نیست (در C++ به حلقه while مشهورند). در این حلقه‌ها با توجه به ورودی، تعداد تکرار مشخص می‌شود. و دقیقاً نمی‌توان تعداد تکرار حلقه را بدون ورودی معین کرد. این حلقه‌ها فقط شامل شرطی هستند که تا زمانیکه برقرار باشد حلقه اجرا می‌شود.

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

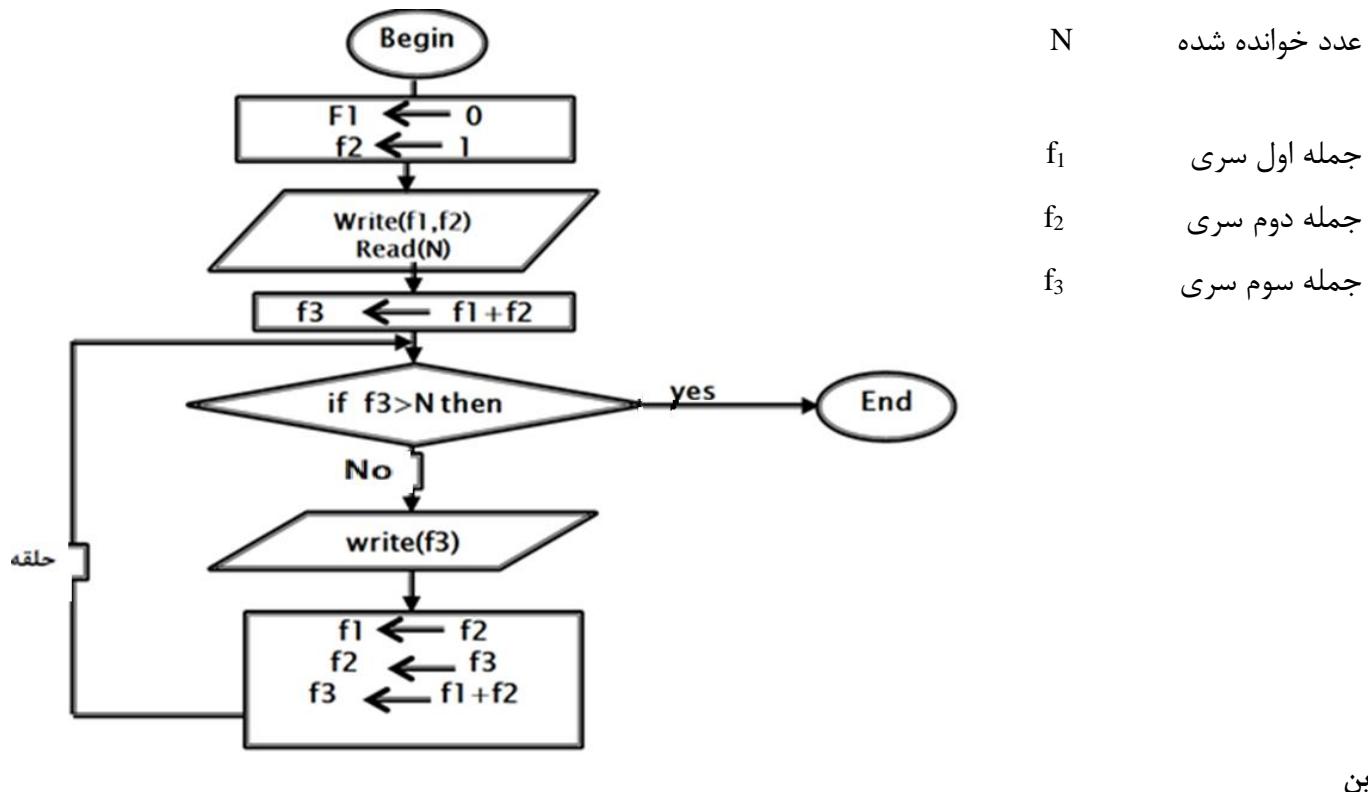


مثال: فلوچارتی رسم کنید که عددی را از ورودی دریافت کرده سپس تعداد ارقام آن را شمرده در خروجی چاپ نماید.



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

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



تمرین

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

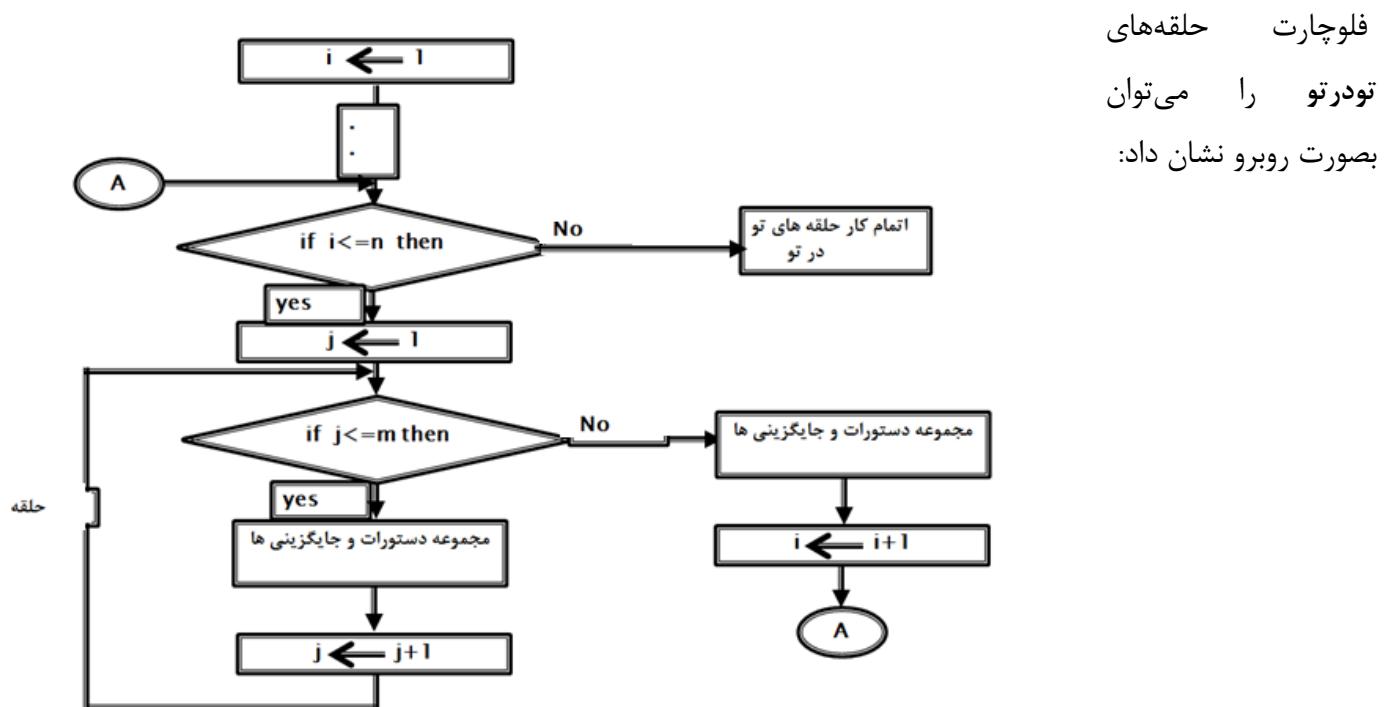
۲- فلوچارتی رسم کنید که N را از ورودی دریافت کرده، N جمله سری فیبوناچی را تولید نماید.

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

حلقه‌های تودر تو

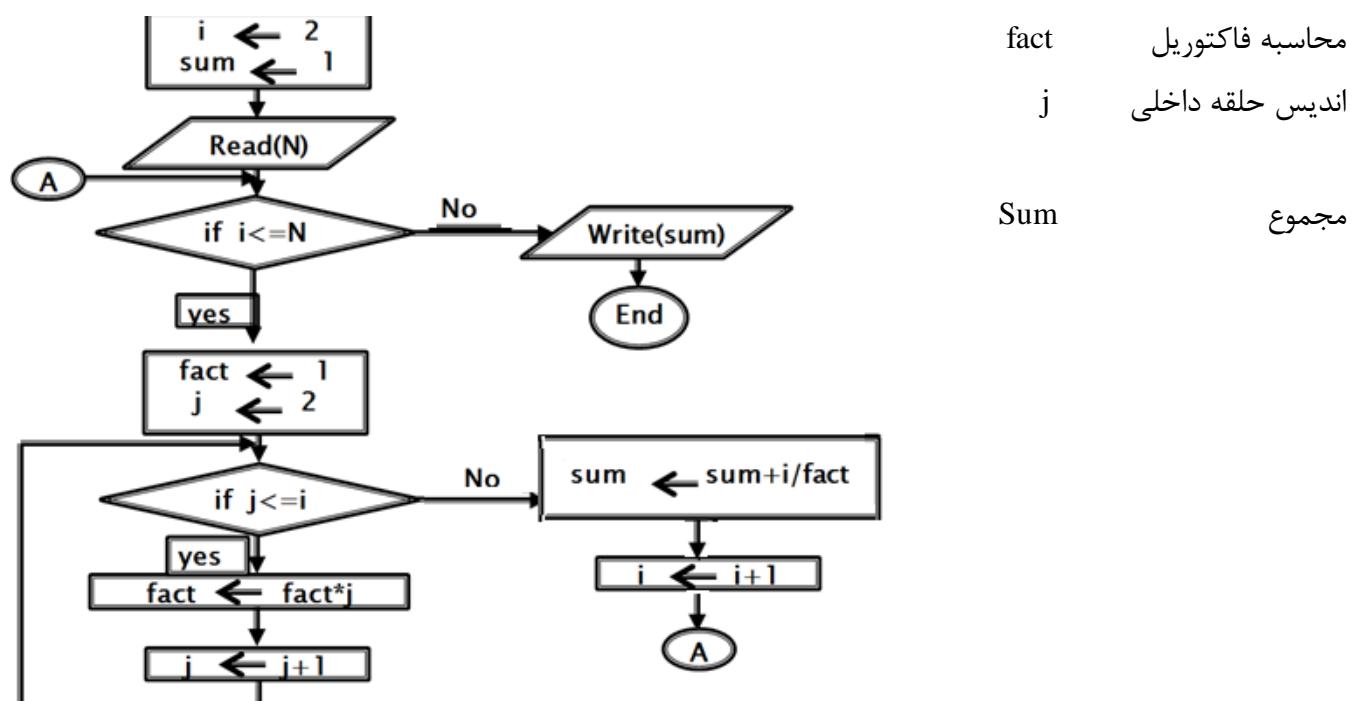
الگوریتم‌هایی که تا حال بکار بردیم، فقط شامل یک حلقه بودند. در صورتی که در بسیاری از مسائل ممکن است نیاز به استفاده از چند حلقه در داخل هم باشیم. در این نوع حلقه‌ها باید دقیق‌تری به خرج دهیم، تا مشکلی پیش نیاید. اگر از حلقه‌های نوع اول بصورت تودر تو استفاده کنیم در اینصورت برای هر حلقه شرط نهایی و اندیس 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 $n \times m$ بار تکرار خواهد شد.



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

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



تمرینات آخر فصل

- ۱- فلوچارتی رسم نماید که N عدد از ورودی دریافت کرده تعداد اعداد اول و کامل را شمرده در خروجی چاپ نماید.
- ۲- فلوچارتی رسم نماید که X, N را از ورودی خوانده مقدار سری زیر را محاسبه کند:
$$S = 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \dots + \frac{x^n}{N!}$$
- ۳- فلوچارتی رسم نماید که عددی را از ورودی دریافت کرده مقلوب عدد را محاسبه و در خروجی چاپ کند.
- ۴- 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 نماید که M, N ($m > n$) را از ورودی دریافت کرده سری فیبوناچی بین M, N را تولید کرده، در خروجی چاپ کند.

برنامه نویسی به زبان C++

فهرست مطالب:

① فصل اول : مقدمات زبان C++

② فصل دوم : ساختار های تصمیم گیری و تکرار

③ فصل سوم : سایر ساختار های تکرار

④ فصل چهارم : اعداد تصادفی

⑤ فصل پنجم : آرایه ها

⑥ فصل ششم : توابع

⑦ فصل هفتم : ساختارها و رشته ها

فصل اول: مقدمات C++

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

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 برنامه های ساخت یافته شی گرا را می دهد.

قانون نامگذاری شناسه ها:

(۱) حروف کوچک و بزرگ در نامگذاری شناسه ها متفاوت می باشند.

بنابراین **Xy**، **X_y**، **XY**، **xY**، **xy** چهار شناسه متفاوت از نظر C++ می باشد.

(۲) در نامگذاری شناسه ها از حروف الفباء، ارقام و زیر خط (underscore) استفاده می شود و حداقل طول شناسه ۳۱ می باشد و شناسه بایستی با یک رقم شروع نگردد.

(۳) برای نامگذاری شناسه ها 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.

أنواع داده ها

نوع داده	حافظه لازم
int	۴ بایت
unsigned int	۴ بایت
long int	۴ بایت
unsigned long int	۴ بایت
char	۱ بایت
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 آن‌ها استفاده گردد بایستی آن‌ها در برنامه اعلان نمود.

چند مثال از اعلان متغیرها :

برای اعلان متغیر x از نوع int :

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` اعلام نموده .
- ✓ در دستورالعمل `long int a=67000, b=260;` ، متغیرهای `b` و `a` را از نوع `long int` `a=67000` و `b=260` تعریف نموده با مقادیر بترتیب `67000` و `260` 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 .

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

`Char x = '\a';`

<code>\n</code>	Newline	<code>\a</code>	Beep sound	<code>\0</code>	Null character
<code>\t</code>	Tab	<code>\"</code>	Double quote	<code>\?</code>	Question mark
<code>\b</code>	Backspace	<code>\'</code>	Single quote	<code>\ </code>	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 :

✓ یک رشته ده کرکتری می‌باشد که با توجه به کرکتر `\0` که به انتهای آن در حافظه اضافه می‌شود جمعاً 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:

از عملگرهای انتساب چندگانه نیز می‌توان استفاده نمود. که مقدار سه متغیر Z و y و x برابر با 26 می‌شود.

```
x=a+b;
```

```
x=35 ;
```

```
x=y=z=26 ;
```

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

جمع	+
تفريق	-
ضرب	*
تقسيم	/
باقيمانده	%

اين عملگرها دو تائي مى باشند زيرا روی دو عملوند عمل مى نمايند. از طرف ديگر عملگرهاي + و - را مى توان بعنوان عملگرهاي يكتائي نيز در نظر گرفت.

عبارت	نتيجه
$5 + 2$	7
$5 * 2$	10
$5 - 2$	3
$5 \% 2$	1
$5 / 2$	2

✓ مثال ۱ :

در حالتی که هر دو عملوند عملگرهاي % ، / ، * ، + ، - از نوع صحيح باشد
 نتيجه عمل از نوع صحيح مى باشد.

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

✓ مثال ۲ :

در صورتيكه حداقل يکي از عملوندهای عملگرهاي / ، * ، - ، + از نوع اعشاري باشد نتيجه عمل از نوع اعشاري مى باشد.

در C++ ، افزایش يک واحد به مقدار يک متغير از نوع صحيح را افزایش و بطور مشابه کاهش يک واحد از مقدار يک متغير از نوع صحيح را کاهش مى نامند.

عملگر کاهش را با - و عملگر افزایش را با ++ نمایش مى دهند. چون عملگرهاي ++ و - فقط روی يک عملوند اثر دارند اين دو عملگر نيز جزء عملگرهاي يكتائي مى باشند.

عملگرهای افزایش و کاهش

در C++ ، افزایش يک واحد به مقدار يک متغير از نوع صحيح را افزایش و بطور مشابه کاهش يک واحد از مقدار يک متغير از نوع صحيح را کاهش مى نامند.

عملگر کاهش را با - و عملگر افزایش را با ++ نمایش مى دهند. چون عملگرهاي ++ و - فقط روی يک عملوند اثر دارند اين دو عملگر نيز جزء عملگرهاي يكتائي مى باشند.

سه دستور العمل :

++x;

x++;

x=x+1;

معادل می باشند و بطريق مشابه سه دستور العمل زير نيز معادل می باشند.

-- y;

y=y-1;

y--;

از عملگرهای ++ و -- می توان بدو صورت پیشوندی و پسوندی استفاده نمود. در دستورالعمل های پیچیده عملگر پیشوندی قبل از انتساب ارزیابی می شود و عملگر پسوندی بعد از انتساب ارزیابی می شود.

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

از عملگرهای یکتائی می باشد و مشخص کننده تعداد بایت هایی است که یک نوع داده اشغال می کند.

مثال :

int x;

cout << sizeof x ;

مقدار 4 نمایش داده می شود.

cout << sizeof(float) ;

مقدار 4 نمایش داده می شود.

برای ساده‌تر نوشتمن عبارت‌ها در C++، می‌توان از عملگرهای جایگزینی محاسباتی استفاده نمود.

$\% = \quad / = \quad * = \quad - = \quad + =$

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.

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

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 خوانایی بیشتر و درک بهتر برنامه می‌شود. بنابراین توصیه بر آن است که حتی الامکان در برنامه‌ها از توضیحات استفاده نماییم. 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 :

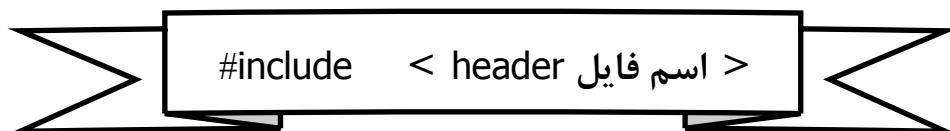
```
/* this is a program  
to calculate sum of  
n integer numbers */
```

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 فایل‌های کتابخانه دسته بندی و مورد استفاده قرار می‌گیرند. این فایل‌ها را فایل‌های header می‌نامند و دارای پسوند h. می‌باشند.

نحوه استفاده از توابع کتابخانه ای

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



تابع	نوع	شرح	فایل هدر
abs(i)	int	قدر مطلق 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

C++ برنامه در

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

روش دوم:	روش اول :
<pre>using namespace std; #include < > void main() { ; ; ; ; ; ; }</pre>	<pre>using namespace std; #include < > int main() { ; ; ; ; ; ; return 0 ; }</pre>

نکات:

به خطاهای برنامه نویسی error می‌گویند. 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 معمولاً 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 cout 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 خطاهای توسعه کامپایلر تشخیص داده نمی‌شود، و حتی ممکن است برنامه اجرا شود، ولی پس از اجرا، برنامه نتایجی به همراه دارد که با پیش‌بینی 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.

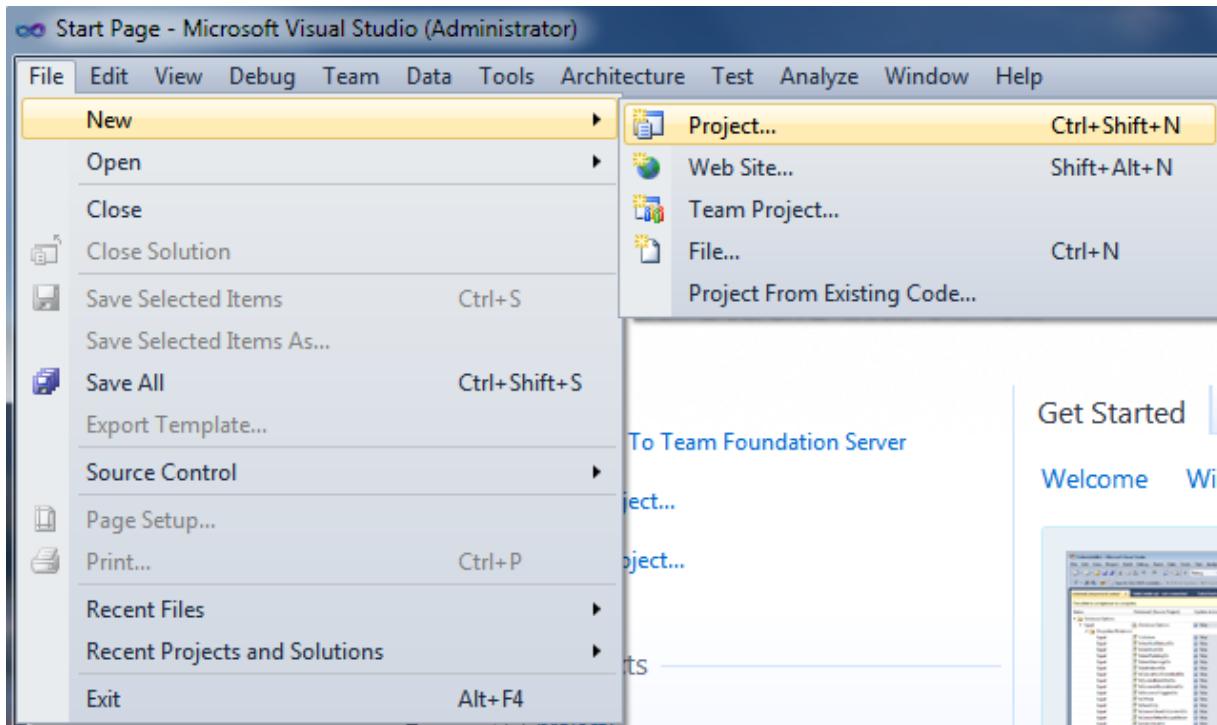
ما برای کدنویسی برنامه‌هایمان، به نرم افزارهای دیگری نیاز داریم تا کدهای برنامه را درون آن بنویسیم و نتایج آن‌ها را برایمان نمایش بدهد، به این برنامه‌ها مترجم (Compiler) گفته می‌شود، که عبارتند از:

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

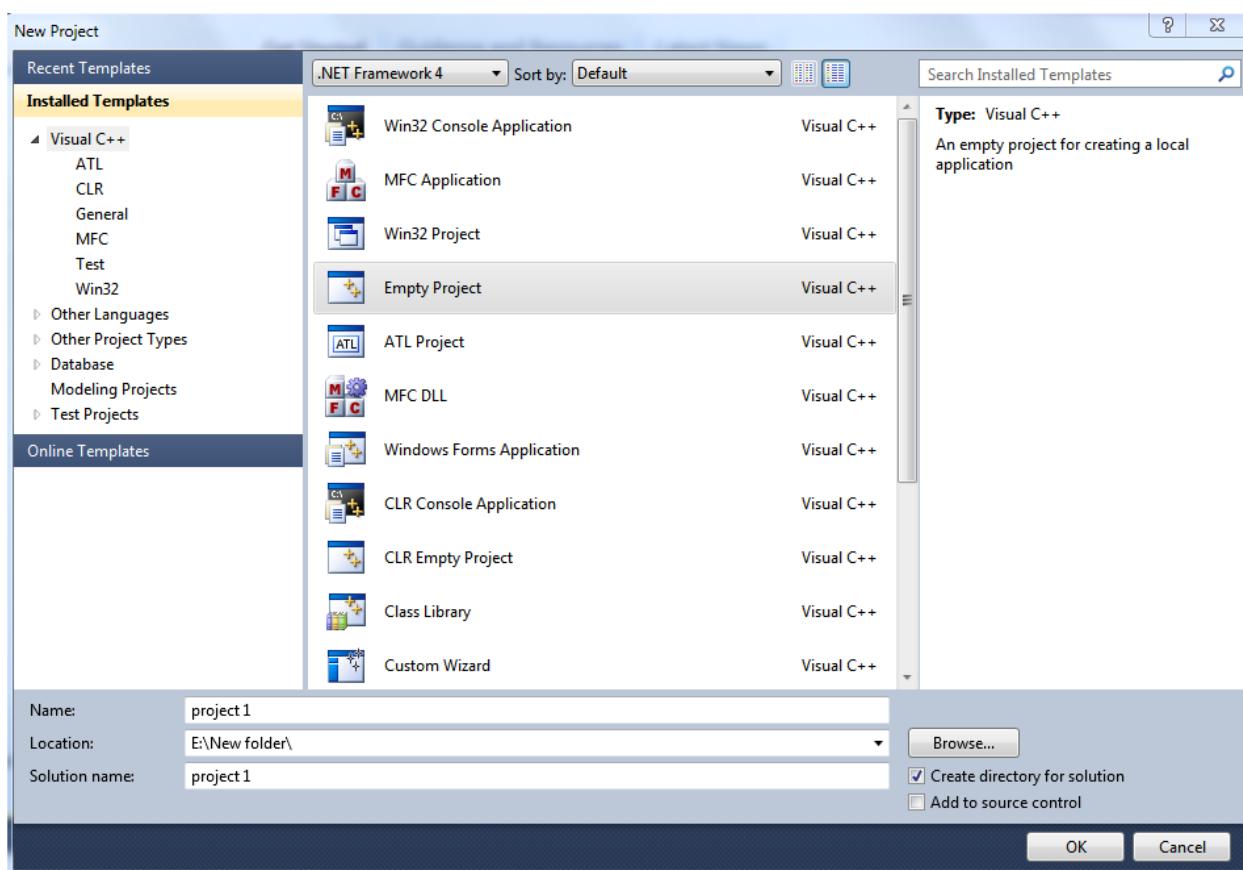
که من استفاده از Microsoft visual studio 2010 را پیشنهاد می‌کنم، زیرا نوشتن و ویرایش کد در این محیط راحت‌تر است.

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

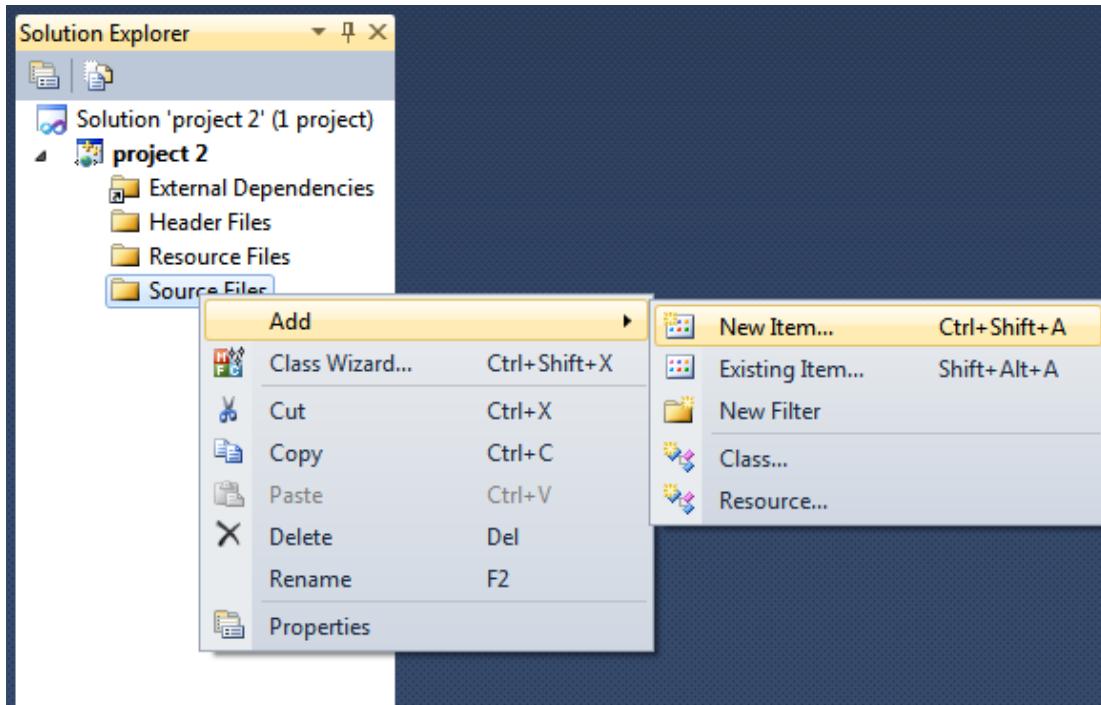
پس از باز کردن نرم افزار 2010 Visual Studio همانند شکل زیر بر روی گزینه File کلیک کرده، سپس با رفتن به روی دکمه‌ی New، گزینه‌ی Project را انتخاب می‌کنیم:



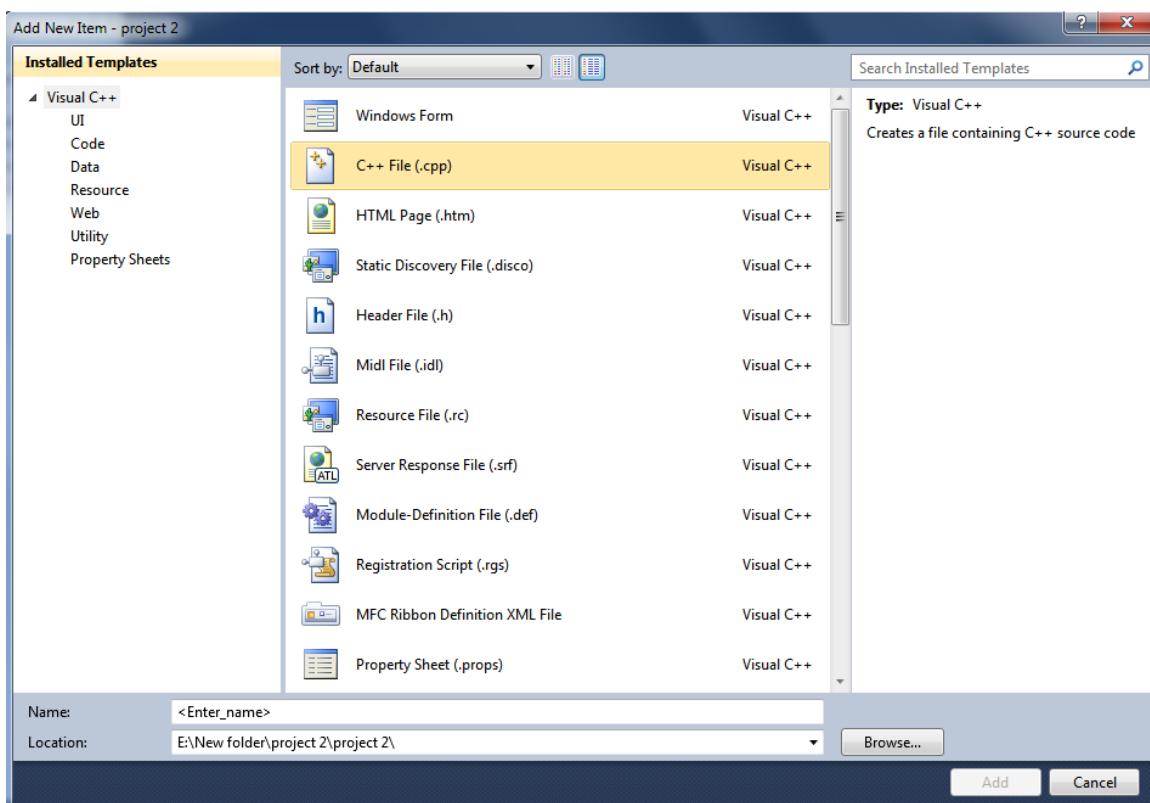
سپس در سمت چپ قسمت Visual C++ را انتخاب کرده و از قسمت سمت راست گزینه Empty Project را انتخاب می‌کنیم همچنین در پایین در کادر name بک نام به پروژه‌ی خود اختصاص دهید و محل ذخیره پروژه را با استفاده از دکمه‌ی browse مشخص کنید:



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



سپس در پنجره‌ی باز شده در سمت راست همانند شکل زیر روی گزینه 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.

عبارة سبز رنگی که با علامت // شروع می‌شود، توسط کامپایلر نادیده گرفته می‌شود، زیرا از این نوع خطوط برای وضوح بخشیدن به برنامه استفاده می‌شود، همان طور که مشاهده می‌کنید، روی 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 خلی 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، بدنی اصلی برنامه را تشکیل می‌دهد، که هر برنامه‌ی ما حداقل باید یک بدنی 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 بدنی اصلی برنامه است، که گیومه خالی به این علت است که برنامه مقداری را برنمی‌گرداند، همین طور بدنی 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 خواهند 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، حتماً 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 تكميلي

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 ; (سیمی کالن) به پایان می‌رسند، به جز دستورهای پیش پردازنه.

۳. تفاوت " و ' : هرگاه متنی بین " (دابل کوئیشن) باید به آن رشته یا لیتل ال گفته می‌شود، ولی بین ' (کوئیشن) 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. (عموماً برای نوشتن کد خطا به کار می‌رود، البته استفاده‌های دیگه ای هم داره که می‌توانید خودتون پیدا کنید!)

۵. دستورهای جریان ورودی/ خروجی به ترتیب شامل دستورهای cin/cout هستند، که استفاده از این دستورها را در برنامه‌های بعدی که می‌نویسیم، توضیح خواهیم داد، البته این دستورها را با این علامت‌ها 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 علامت‌های توضیحی استفاده کنید که شامل // و /* */ است، علامت // برای توضیح یک خطی به کار می‌رود و علامت‌های /* */ برای توضیح چند خطی به کار می‌رود، که توضیح‌های 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 پردازند می‌گویند، در اصل هر کدام از آنها یک کتابخانه (library) سی پلاس پلاس را فراخوانی می‌کنند. حال کتابخانه چیست؟ منظور از کتابخانه در C++ یعنی مجموعه تابع‌ها و دستورهایی که ما با فراخواندن یکی از دستورهای پیش پردازند آن‌ها را در اختیار برنامه قرار می‌دهیم، به بیان ساده‌تر، وقتی شما یک دستور پیش پردازند مثل `iostream` را در برنامه به عنوان کد پیش پردازند می‌نویسید، تمام دستورها و تابع‌های 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 بود!) به عنوان مثالی دیگر اگر دستور پیش پردازند `math.h` را به صورت زیر در برنامه قرار دهید، تمام تابع‌ها و عملگرهای ریاضی مثل `توان`، `سینوس`، `کسینوس` و... در برنامه قرار می‌گیرد:

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

عملگرها رابطه ای

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

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:

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.

مثال ۱ :

```
int x=10,y=20,b;
```

```
b=(x>y) ? x : y ;
```

این دو دستور العمل باعث می‌شوند که ماکریم مقدار y و x در b قرار بگیرد.

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

اگر مقدار x بزرگتر یا مساوی ده باشد رشتة passed در غیر اینصورت رشتة failed نمایش داده می‌شود.

دستورالعمل شرطی

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

```

if ( عبارت )
{
    دستورالعمل 1 ;
...
    دستورالعمل ;
}
else
{
    دستورالعمل 1 ;
...
    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 می‌افزاید. در غیر این صورت یعنی زمانی که x و y مساوی هم هستند، مقدار y را نمایش داده و از y یک واحد کم می‌کند.

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 عبارت‌ها 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 می‌شوند، بعد از آن به c یک واحد اضافه می‌شود و در نهایت c و b با هم جمع می‌شوند. مقدار عبارت حاصل برابر است با $b+c$ که معادل 10 می‌باشد.

عملگرهای منطقی

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

AND

OR

NOT

که در C++ به ترتیب بصورت `&&` ، `||` و `!` نشان داده می‌شود.

جدول درستی سه عملگر شرطی

And

a	b	<code>a && b</code>
true	true	True
true	false	False
false	true	False
false	false	False

a	b	<code>a b</code>
true	true	True
true	false	True
false	true	True
false	false	False

Or

a	<code>!a</code>
true	False
false	True

Not

```
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 پاره خط را از ورودی گرفته مشخص می‌نماید که آیا تشکیل یک مثلث می‌دهد 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 ;
}
```

For دستورالعمل

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

(عبارت 1) ؛ عبارت 2 ؛ عبارت 3) for

```
{
    دستورالعمل 1 ;
    دستورالعمل 2 ;
    .
    .
    .
    دستورالعمل n ;
}
```

ساختار for

؛ معرفی کنترل گر حلقه

) گام حرکت ؛ شرط حلقه ؛ مقداردهی اولیه کنترل گر حلقه (for

{

؛ مجموعه دستورات بدنه حلقه

}

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

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 می دهد.

```
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 ۰ تا ۹ 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 **break**; استفاده شود، ادامه 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:

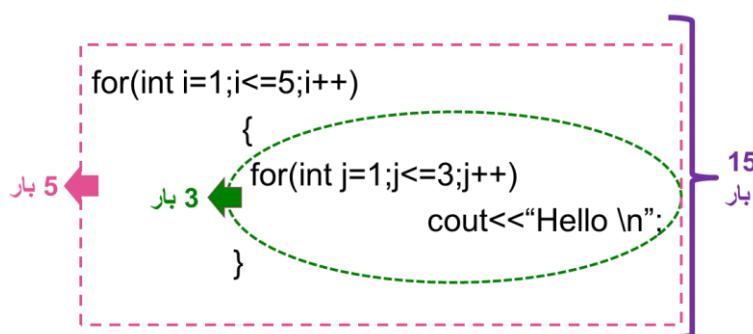
قطعه کدی که تعدادی کاراکتر از صفحه کلید خوانده، بعد از فشردن دکمه **E** تعداد آن ها را مشخص کند:

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

حلقه **for** تودر تو

می توان داخل بدنه **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 ۱ ، ۲ ، ۳ 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 هر کدام 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;
```

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

روش دوم:

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

دستور for در continue;

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

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

.

.

n ; دستورالعمل

}

تفاوت دستورهای for و while

دستورالعمل for زمانی استفاده می‌شود 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 می‌توان 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 می‌شود :

0
1
2
3
4

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

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

return 0 ;
}
```

مثال: نمایش اعداد ۰ تا ۱۰۰:

تمرین: برنامه زیر را تفسیر کنید:

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

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

```
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

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

;معرفی کنترل گر حلقه

;مقدار دهنده اولیه کنترل گر حلقه

شکل کلی این دستورالعمل بصورت روبرو می‌باشد.

do

{ ; دستورالعمل ۱

؛ دستورالعمل ۲

.

.

;

گام حرکت

} while();

تفاوت دستورهای while و do while

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

مثال: برنامه فوق، ارقام ۰ تا ۹ را روی ده خط نمایش می‌دهد.

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( )
{
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 ۰ و ۱۰۰۰ هستند. اگر بین اعداد وارد شده عدد منفی وارد شود، بواسطه دستور حلقه خاتمه می‌یابد.
اگر عدد وارد شده بزرگتر از ۱۰۰۰ باشد نیز شرط حلقه برآورده نشده و حلقه خاتمه می‌یابد.

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

counter : 11

```

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 می توان در دستورالعمل های تکرار for ، while ، do while استفاده نمود. این دستورالعمل باعث می شود 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 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 ;
}

```

ن 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 استفاده 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 می باشد را ارائه نموده است.

شكل 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: برنامه ذیل یک سطر متن انگلیسی که به Z CTRL ختم می‌شود را گرفته دقیقاً نمایش می‌دهد.

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

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

مثال: در قطعه برنامه ذیل از تابع `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: برنامه ذیل یک سطر متن انگلیسی را گرفته کرکترهای خالی (blank) آن را حذف نموده و نمایش می‌دهد.

```
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 می‌آید و در نتیجه مقدار y برابر با 25.0 می‌شود. باستی توجه داشت که نوع متغیر x عوض نمی‌شود 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 می‌شود و سپس مقدار ۱۴,۷۵ نمایش داده می‌شود.

جدول اولویت عملگرها

()	چپ به راست
Static_cast < >() ++ -- + - sizeof	راست به چپ
* / %	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 ۵۰ و ۲۰ و ۱۰ و ۵ ریالی خرد کند.
۲. برنامه ای که خروجی زیر را چاپ کند:



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

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

فصل چهارم: اعداد تصادفی

➤ تعریف نوع داده (`typedef`)

➤ تولید اعداد تصادفی

➤ داده های از نوع شمارشی

اعداد تصادفی

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

مثال: برنامه زیر ۱۰ عدد تصادفی بین ۰ و ۳۲۷۶۷ را ایجاد می نماید.

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

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

مثال: برنامه زیر ۱۰ عدد تصادفی بین ۰ و ۳۲۷۶۷ را ایجاد می نماید. ((`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 عدد شانسی بین 0 و 1 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 ;
}
```

تعريف نوع داده (typedef)

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

typedef type newtype;



مثال:

```
typedef int integer;
```

integer x,y;

حال می‌توان y و x را بصورت رو برو تعریف نمود:

داده‌های از نوع شمارشی

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

مثال :

```
enum color {red, blue, green, yellow, brown} ;
```

color یک نوع داده شمارشی می‌باشد.

چند مثال :

```
enum 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 مقدار داده‌های از نوع شمارشی بایستی در برنامه تعیین نمود. دستورالعمل 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 می باشد که می تواند چندین مقدار را در خود جای دهد.

کلیه عناصر یک آرایه از یک نوع می باشند.

عناصر آرایه بوسیله اندیس آن ها مشخص می شوند.

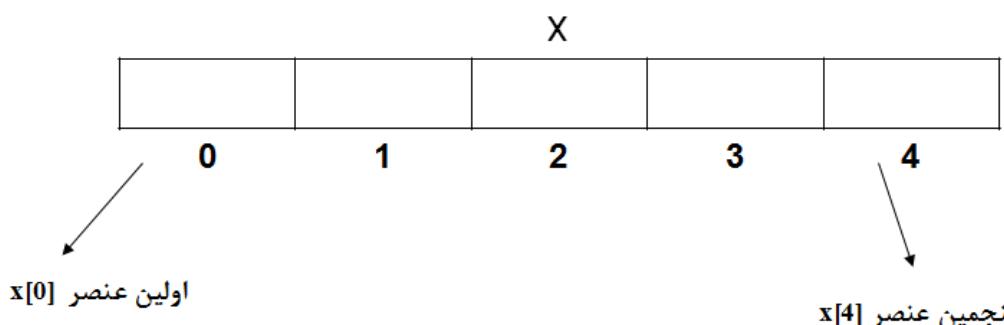
در C++ ، اندیس آرایه از صفر شروع می شود.

آرایه ها در برنامه نویسی در مواردی کاربرد دارند که بخواهیم اطلاعات و داده ها را در طول اجرای برنامه حفظ نمائیم.

مثال:

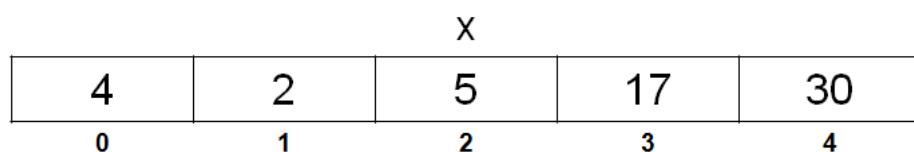
ایجاد آرایه یک بعدی از نوع int

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



تخصیص مقادیر اولیه به عناصر آرایه :

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



دريافت مقادير 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: با استی 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 می‌دهد 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)

ماتریس‌ها بوسیله آرایه‌های دوبعدی در کامپیوتر نمایش داده می‌شوند.

```
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

: نکته ۱

```
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

: نکته ۲

```
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

: نکته ۳

در یک آرایه دو اندیسی، هر سطر، در حقیقت آرایه‌ای یک اندیسی است. در اعلان آرایه‌های دو اندیسی ذکر تعداد ستون‌ها الزامی است.

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

مثال: برنامه زیر یک ماتریس 3×4 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 می‌توان ماتریسی با اندازه بعد m ساخت.

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

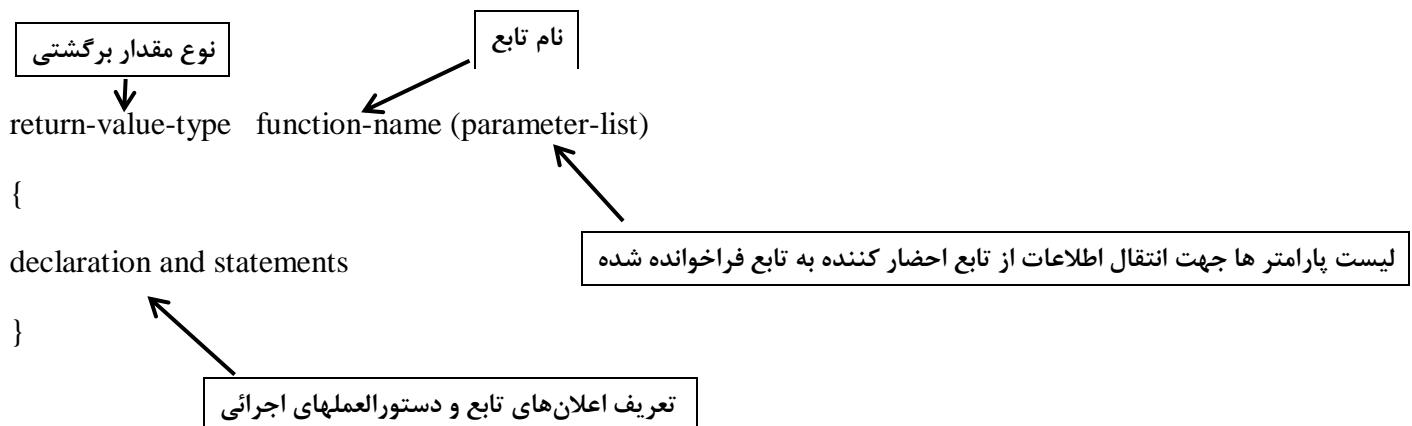
$$[\cdot\cdot]_{m \times n} \times [\cdot\cdot]_{n \times p} = [\cdot\cdot]_{m \times p}$$

فصل ششم: توابع

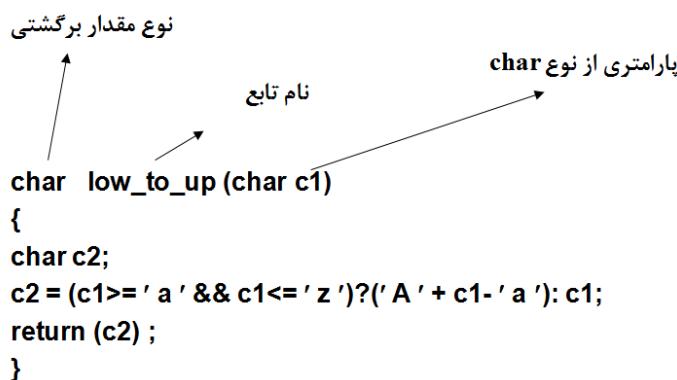
- تابع درون خطی
- سربارگذاری توابع
- تابع بازگشتی
- کلاس های حافظه (storage classes)
- تعریف تابع
- انتقال پارامترها از طریق ارجاع

تعریف توابع

استفاده از توابع در برنامه ها به برنامه نویس این امکان را می دهد که بتواند برنامه های خود را به صورت قطعه قطعه برنامه بنویسد. تا کنون کلیه برنامه هائی که نوشته ایم فقط از تابع main استفاده نموده ایم. شکل کلی توابع بصورت زیر می باشد:



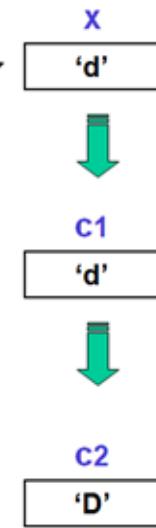
مثال: تابع زیر یک حرف کوچک را به بزرگ تبدیل می نماید.



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;
}
```

آرگومان

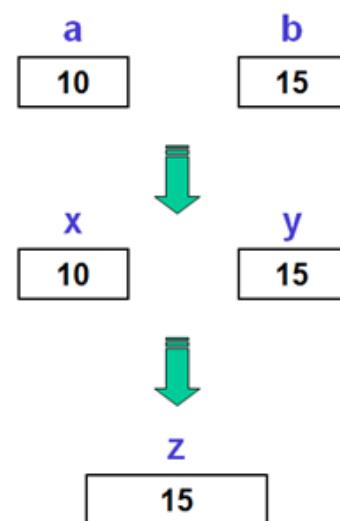


```
int maximum(int x, int y)
{
int z ;
z=(x >= y)? x : y;
return z;
}
```

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

```
#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 a, b

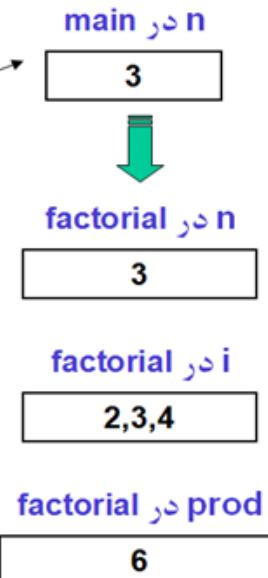


نکته: اسامی پارامترها و آرگومان های یک تابع می‌توانند هم نام باشند.

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

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

```
#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 با استفاده از تابع احضار شونده قبل از تعریف تابع احضار کننده در برنامه ظاهر گردد.

نکته: اگر بخواهیم در برنامه‌ها ابتدا تابع main ظاهر گردد با استفاده از prototype تابع یعنی پیش نمونه تابع که شامل نام تابع، نوع مقدار برگشتی تابع، تعداد پارامترهایی 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 می‌گردد.

```

#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 ;
}

```

مثال:

تابع مقداری بر نمی‌گرداند.

احصار بواسیله 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;
}

```

main در a

20

modify در a

20

modify در a

40

خروجی برنامه :

20
40
20

در این نوع احضار تابع حافظه‌های مورد استفاده آرگومان‌ها و پارامترها از هم متمایزند و هرگونه تغییر در پارامترها باعث تغییر در آرگومان‌های متناظر نمی‌گردد.

تابع بازگشتی (recursive functions)

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

مثال ۱: نحوه محاسبه فاکتوریل از طریق تابع بازگشتی

$$n! = 1 * 2 * 3 * \dots * (n-1) * n$$

$$f(n) = n!$$

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

$$n! = 1 * 2 * 3 * \dots * (n-2) * (n-1) * n$$

$$n! = (n-1)! * n$$

پیاده سازی تابع بازگشتی محاسبه‌ی 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 * factorial(n-1)) ;
}
```

مثال ۲: نحوه محاسبه n امین مقدار دنباله فیبوناچی از طریق تابع بازگشته

0 , 1, 1, 2, 3, 5, 8, 13, 21 , 34, ...

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

برنامه زیر n امین مقدار دنباله 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)) ;
}
```

مثال ۳: 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 می‌شود.

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 arr1 [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

تابع درون خطی (inline)

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

ashkal_tavab_inline

به جای داشتن تنها یک کپی از تابع ، چند کپی از دستورالعمل های تابع در برنامه اضافه می شود که باعث بزرگ شدن اندازه یا طول برنامه می شود. بنابراین از inline برای توابع کوچک استفاده می گردد.

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 آرگومان‌ها در پارامترهای 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 آرگومان‌ها و پارامترهای متناظر بصورت اشتراکی مورد استفاده قرار می‌گیرد. این روش call by reference نامیده می‌شود.

در این روش پارامترهایی که از طریق call by reference عمل می‌نمایند در پیش نمونه تابع قبل از نام چنین پارامترهایی 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	100

← مقدار آرگومان X تغییر نمی کند.

ادامه خروجی :

x	y	b
5	10	100

نکته: وقتی پارامتری بصورت call by reference اعلان می‌گردد این بدان معنی است که با تغییر مقدار این پارامتر در تابع حاضر شده مقدار آرگومان متناظر نیز تغییر می‌نماید.
مثال: برنامه زیر 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) آن‌ها و دیگری بوسیله کلاس حافظه آن‌ها.
نوع متغیر قبلًا اشاره شد. بعنوان مثال int ، double ، float ، ولی کلاس حافظه یک متغیر در مورد طول عمر و وسعت و دامنهٔ متغیر بحث می‌نماید.

بطور کلی کلاس حافظه متغیرها به چهار دستهٔ تقسیم می‌گردد :

automatic .۱

static .۲

external .۳

register .۴

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 می‌رود به عبارت دیگر وسعت و دامنه متغیرهای از نوع **automatic** تابعی می‌باشد که متغیر در آن اعلان گردیده است.

Static: متغیرهای **static** نیز در درون توابع تعریف می‌شوند و از نظر وسعت و دامنه شبیه متغیرهای **automatic** هستند ولی در خاتمه اجرای تابع، حافظه وابسته به این نوع متغیرها 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

آن‌ها 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** متغیرهایی هستند که در بیرون از توابع اعلان می‌شوند و وسعت و دامنه فعالیت آن‌ها 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 << endl;
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 می شود.

Register: وقتی متغیری از نوع `register` اعلان می شود از کامپیوتر عملأً درخواست می شود که به جای حافظه از یکی از رجیسترها موجود استفاده نماید.

کاربرد کلاس `register`: معمولاً از نوع رجیستر برای شاخص های دستور تکرار و یا اندیس های آرایه ها استفاده می شود. بایستی توجه داشت که متغیرهای از نوع رجیستر قابل استفاده در دستور `cin` نمی باشند.

سربارگذاری توابع (function overloading)

در C++ این امکان وجود دارد که در یک برنامه بتوانیم از چند تابع 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 معادله‌ی درجه دومی را خوانده، آن ها را به تابعی ارسال کند. تابع معادله را حل کند و جواب‌ها را در خروجی چاپ کند. (تابع جواب‌ها را به تابع اصلی برنمی‌گرداند.)
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 بازگشته صورت گیرد.
۴. برنامه ای بنویسید که حاصلضرب دو عدد صحیح را به کمک جمع کردن محاسبه کند. برای این کار از تابع بازگشته استفاده کنید.

$$\left. \begin{array}{ll} b=1 & a \\ b>1 & a \times (b-1) + a \end{array} \right\} = a \times b$$

فصل هفتم: ساختار ها

► رشته ها و توابع مربوطه

► ساختارها
Union ها

ساختارها

ساختارها شبیه آرایه ها بوده بدین صورت که یک نوع داده گروهی است که فضای پیوسته از حافظه اصلی را اشغال می نماید.

```
نام ساختار  
struct time  
{  
int hour ; // 0 - 23  
int minute ; // 0 - 59  
int second; //  
};
```

اعضا ساختار

اما عناصر ساختار الزاماً از یک نوع نمی باشند بلکه اعضای یک ساختار می توانند از نوع های مختلفه از قبیل char ، int ، float ، ... باشند.

تعريف ساختار:

struct account

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

ساختار account دارای چهار عضو می باشد:

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

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

روش دوم	روش اول
<pre>struct account { int acc_no ; char acc_type; char name[80]; float balance; }; account cust1, cust2, cust3;</pre>	<pre>struct account { int acc_no; char acc_type; char name[80]; float balance; } cust1, cust2, cust3;</pre>

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 . جزو عملگرهای یکتائی می‌باشد.

مثال:

cust .acc_no = 4236

cust .acc_type = 'r'

cust . name = "Nader Naderi"

cust . balance = 7252.5

struct date {

int month;

int day;

int year;

};

struct account {

int acc_no ;

char acc_type;

char name[80];

float balance ;

date lastpay ;

};

اگرداشته باشیم

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

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 آن‌ها مقادیر اولیه تخصیص نمود.

```
struct struct1 {  
    char name[40];  
    int pay1;  
    int pay2;  
};  
struct1 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 آن‌ها 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

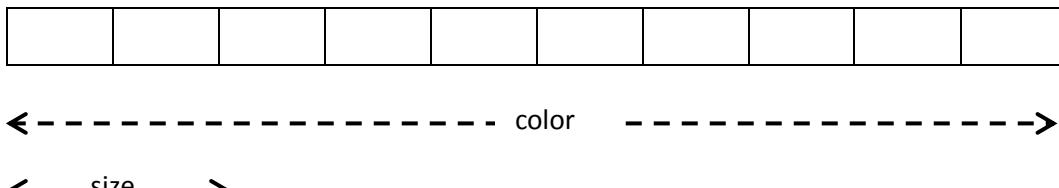
union از نظر ساختاری شبیه struct می‌باشد. با این تفاوت که عضوهایی که تشکیل union می‌دهد همگی از حافظه مشترک در کامپیوتر استفاده می‌نمایند. تمام اعضای داده‌ای در یونیون از یک محل شروع می‌شوند.

بنابراین استفاده از union باعث صرفه‌جوئی در حافظه می‌گردد.

مثال:

union id

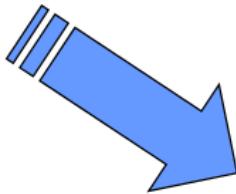
```
{  
    char color [10];  
    int size;  
} x, y;
```



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.

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

مثال:



x	
بایت اول	بایت دوم
y[1]	
	y[0]

رشته‌ها و توابع مربوطه

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

```
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 S[10];
```

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

S[0]	S[2]	S[4]	S[6]	S[8]					
c	o	m	p	u	t	e	r	\0	?
S[1]	S[3]	S[5]	S[7]	S[9]					

```
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 کاراکتر بخارط '\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 می گیرد یا اگر *space* یا *tab* را فشار دهد دوباره

مقدار می گیرد.

استفاده از متدهای get()

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

`cin.get('جداکننده', [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 حداقل مکانی است که رشته می‌تواند بگیرد.

مثلاً:

```
char str[30];
```

```
cin.get(str, 10);
```

```
cin.get(str, 10, '.'');
```

در دستور دوم حداقل می‌توانیم با طول ۱۰ به `str` از ورودی مقدار بدھیم و دستور سوم بجای `Enter` که آخر رشته

باشد با '.' مشخص می‌شود.

تفاوت `cin` و `get`

همانگونه که گفته شد انتهای ورودی دستور `cin` را یا `Enter` یا `space` می‌کند ولی در

می‌توانیم فاصله و `tab` را نیز به رشته بدھیم. 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 بربزیم.

دستور $\text{strcpy}(S1, S2)$ مقدار $S2$ را در داخل $S1$ کپی می نماید.

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

```
char S2[20];
```

```
strcpy(S2, S1);
```

مقایسه رشته ها:

برای مقایسه رشته نمی توانیم به این صورت عمل نماییم.

چون $S2, S1$ اشاره گر به رشته می باشند ولی دستوری بنام

وجود دارد که دو رشته $S2, S1$ را مقایسه می نماید.

$S_1 S_2$	$S_1 = S_2$	$S_1 > S_2$	$S_1 < S_2$
$\text{strcmp}(S_1, S_2)$	•	عدد مثبت	عدد منفی

عدد برگردانده شده از تابع strcmp طبق جدول بالا می باشد. منظور از اینکه دو رشته از هم کوچکتر باشد

یا بزرگتر به این صورت می باشد که سمت چپ ترین کارکتر را با کد اسکی مقایسه می کند اگر بزرگتر بود که آن

رشته از رشته دوم بزرگتر می باشد. اگر کوچکتر بود، آن رشته کوچکتر می باشد و اگر مساوی بود کارکتر بعدی را

نگاه می کند.

الحق دو رشته:

برای الحق دو رشته از تابع $\text{strcat}(S1, S2)$ استفاده می نماییم. این تابع $S2$ را به انتهای $S1$ وصل می نماید.

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 strcmp(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 << strcmp(s1, s2) << endl;
```

0

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 بر می‌گرداند.

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

-1

تابع strncmp(s1, s2,n)

حداکثر n کرکتر از رشته s1 را با n 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 بر می‌گرداند.

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

0

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

32

تابع strcat(s1, s2)

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

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

تابع strncat(s1, s2, n)

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

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

تابع strlen(s)

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

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

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

تابع strcpy(s1,s2)

دو رشته s1 و s2 را بعنوان آرگومان گرفته رشته s2 را در رشته s1 کپی می‌نماید و نهایتاً مقدار رشته s1 را برمی‌گرداند.

```
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 رشته s2 ، s1 و مقدار صحیح و مثبت n را بعنوان آرگومان گرفته، حداکثر n کرکتر را از رشته s2 در رشته s1 کپی نموده، نهایتاً مقدار رشته s1 را برمی‌گرداند.

```
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 ابتداء 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;
}
```

-1
-6

مثال:

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

4

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+1]=='\0' )
        return 0;
return (s[i]-t[i]);
}
```