



دولت جمهوري اسلامي افغانستان
اداره تعليمات تخنيكي و مسلکي
معاونيت امور اکادميک
رياست نصاب و تربيه معلم

اساسات برنامه نویسی (C++)

رشته: کمپیوتر ساینس - دیپارتمنت: شبکه
صنف ۱۳ - سمستر دوم

سال: ۱۳۹۹ هجری شمسی



شناسنامه کتاب

نام کتاب: اساسات برنامه نویسی (C++)

رشته: کمپیوتر ساینس

تدوین کننده: صدیق الله بارکزی

همکار تدوین کننده: عبدالمبین صمدزاده

- کمیته نظارت: ندیمه سحر رئیس اداره تعلیمات تخنیکي و مسلکی
- عبدالحمید اکبر معاون امور اکادمیک اداره تعلیمات تخنیکي و مسلکی
- حبیب الله فلاح رئیس نصاب و تربیه معلم
- عبدالمتمین شریفی آمر انکشاف نصاب تعلیمی، ریاست نصاب و تربیه معلم
- روح الله هوتک آمر طبع و نشر کتب درسی، ریاست نصاب و تربیه معلم
- احمد بشیر هیله من مسؤل انکشاف نصاب، پروژه انکشاف مهارت های افغانستان
- محمد زمان پویا کارشناس انکشاف نصاب، پروژه انکشاف مهارت های افغانستان
- علی خیبر یعقوبی سرپرست مدیریت عمومی تألیف کتب درسی، ریاست نصاب و تربیه معلم

- کمیته تصحیح: مهدی بهار
- دوکتور احمد فرید اسداللهی
- محمد امان هوشمند مدیر عمومی بورد تصحیح کتب درسی و آثار علمی

دیزاین: صمد صبا و سید کاظم کاظمی

سال چاپ: ۱۳۹۹ هجری شمسی

تیراژ: ۱۰۰۰

چاپ: اول

وبسایت: www.tveta.gov.af

ایمیل: info@tveta.gov.af

حق چاپ برای اداره تعلیمات تخنیکي و مسلکی محفوظ است.



سرود ملی

دا وطن افغانستان دی	دا عزت د هر افغان دی
کور د سولې کور د تورې	هر بچی یې قهرمان دی
دا وطن د ټولو کور دی	د بلوڅو، د ازبکو
د پښتون او هزاره وو	د ترکمنو، د تاجکو
ورسره عرب، گوجر دي	پامیریان، نورستانیان
براهوي دي، قزلباش دي	هم ایماق، هم پشه یان
دا هیواد به تل ځلیږي	لکه لمر پر شنه آسمان
په سینه کې د آسیا به	لکه زړه وی جاویدان
نوم د حق مو دی رهبر	وایو الله اکبر وایو الله اکبر



پیام اداره تعلیمات تخنیکي و مسلکی

استادان نهایت گرامی و محصلان ارجمند!

تربیت نیروی بشری ماهر، متخصص و کارآمد از عوامل کلیدی و انکارناپذیر در توسعه اقتصادی و اجتماعی هر کشور محسوب می‌گردد و هر نوع سرمایه‌گذاری بزرگ در بخش‌های مختلف اقتصادی نیازمند به پلان‌گذاری و سرمایه‌گذاری در بخش نیروی بشری و توسعه منابع این نیرو می‌باشد. بر مبنای این اصل و بر اساس فرمان شماره ۱۱ مقام عالی ریاست جمهوری اسلامی افغانستان به تاریخ ۱۳۹۷/۲/۱ اداره تعلیمات تخنیکي و مسلکی از بدنه وزارت معارف مجزا و فصل جدیدی در بخش عرضه خدمات آموزشی در کشور گشوده شد. اداره تعلیمات تخنیکي و مسلکی به‌عنوان متولی و مجری آموزش‌های تخنیکي و مسلکی در کشور محسوب می‌شود که در چارچوب استراتژی ۵ ساله خویش دارای چهار اولویت مهم که عبارت‌اند از افزایش دسترسی عادلانه و مساویانه فراگیران آموزش‌های تخنیکي و مسلکی در سطح کشور، بهبود کیفیت در ارائه خدمات آموزشی، یادگیری مادام‌العمر و پیوسته و ارائه آموزش نظری و عملی مهارت‌ها به‌طور شفاف، کم‌هزینه و مؤثر که بتواند نیاز بازار کار و محصلان را در سطح محلی، ملی و بین‌المللی برآورده کند، می‌باشد. این اداره که فراگیرترین نظام تعلیمی کشور در بخش تعلیمات تخنیکي و مسلکی است، تلاش می‌کند تا در حیطه وظایف و صلاحیت خود زمینه دستیابی به هدف‌های تعیین‌شده را ممکن سازد و جهت رفع نیاز بازار کار، فعالیت‌های خویش را توسعه دهد.

نظام اجتماعی و طرز زندگی در افغانستان مطابق به احکام دین مقدس اسلام و رعایت تمامی قوانین مشروع و معقول انسانی عیار است. اداره تعلیمات تخنیکي و مسلکی جمهوری اسلامی افغانستان نیز با ایجاد زمینه‌های لازم برای تعلیم و تربیت جوانان و نوجوانان مستعد و علاقه‌مند به حرفه‌آموزی، ارتقای مهارت‌های شغلی در سطوح مختلف مهارتی، تربیت کادرهای مسلکی و حرفه‌ای و ظرفیت‌سازی تخصصی از طریق انکشاف و ایجاد مکاتب و انستیتوت‌های تخنیکي و مسلکی در سطح کشور با رویکرد ارزش‌های اسلامی و اخلاقی فعالیت می‌نماید.

فلذا جهت نیل به اهداف عالی این اداره که همانا تربیه افراد ماهر و توسعه نیروی بشری در کشور می‌باشد؛ داشتن نصاب تعلیمی بر وفق نیاز بازار کار امر حتمی و ضروری بوده و کتاب درسی یکی از ارکان مهم فرایند آموزش‌های تخنیکي و مسلکی محسوب می‌شود، پس باید همگام با تحولات و پیشرفت‌های علمی نوین و مطابق نیازمندی‌های جامعه و بازار کار تألیف و تدوین گردد و دارای چنان ظرافتی باشد که بتواند آموزه‌های دینی و اخلاقی را توأم با دست‌آوردهای علوم جدید با روش‌های نوین به محصلان انتقال دهد. کتابی را که اکنون در اختیاردارید، بر اساس همین ویژگی‌ها تهیه و تدوین گردیده است.

بدین‌وسیله، صمیمانه آرزو مندیم که آموزگاران خوب، متعهد و دلسوز کشور با خلوص نیت، رسالت اسلامی و ملی خویش را ادا نموده و نوجوانان و جوانان کشور را به‌سوی قله‌های رفیع دانش و مهارت‌های مسلکی رهنمائی نمایند و از محصلان گرامی نیز می‌خواهیم که از این کتاب به‌درستی استفاده نموده، در حفظ و نگهداشت آن سعی بلیغ به خرج دهند. همچنان از مؤلفان، استادان، محصلان و اولیای محترم محصلان تقاضا می‌شود نظریات و پیشنهادات خود را در مورد این کتاب از نظر محتوا، ویرایش، چاپ، اشتباهات املائی، انشایی و تاپی عنوانی اداره تعلیمات تخنیکي و مسلکی کتباً ارسال نموده، امتنان بخشد.

در پایان لازم می‌دانیم در جنب امتنان از مؤلفان، تدوین‌کنندگان، مترجمان، مصححان و تدقیق‌کنندگان نصاب تعلیمات تخنیکي و مسلکی از تمامی نهادهای ملی و بین‌المللی که در تهیه، تدوین، طبع و توزیع کتب درسی زحمت‌کشیده و همکاری نموده‌اند، قدردانی و تشکر نمایم.

ندیمه سحر

رئیس اداره تعلیمات تخنیکي و مسلکی جمهوری اسلامی افغانستان

ز.....	مقدمه.....
۱.....	فصل اول: معرفی زبان‌های برنامه نویسی Introduction to Programming Languages
۲.....	۱.۱ برنامه نویسی.....
۲.....	۱.۲ تاریخچه زبان های برنامه نویسی.....
۵.....	۱.۳ مترجمان زبان (Language Translators).....
۵.....	۱.۳.۱ اسمبلر (Assembler).....
۵.....	۱.۳.۲ مترجم (Compiler).....
۶.....	۱.۳.۳ تفسیر کننده (Interpreter).....
۹.....	فصل دوم: الگوریتم و فلوچارت Algorithm and Flowchart
۱۰.....	۱.۴ خصوصیات الگوریتم (Properties of Algorithm).....
۱۱.....	۱.۵ علامت گذاری الگوریتم (Algorithm Notations).....
۱۲.....	۱.۵.۱ فواید الگوریتم.....
۱۲.....	۱.۵.۲ نواقص الگوریتم.....
۱۲.....	۱.۶ فلوچارت (Flowchart).....
۱۴.....	۱.۶.۱ فواید فلوچارت (Advantages of Flowchart).....
۱۴.....	۱.۶.۲ محدودیت های فلوچارت (Limitation of Algorithm).....
۱۵.....	۱.۶.۳ قواعد ترسیم فلوچارت ها (Rule for writing flowcharts).....
۱۵.....	۱.۷ اشتباهات مفهومی (Semantic Errors).....
۱۶.....	۱.۸ مستند سازی (Documentation).....
۱۹.....	فصل سوم: ثابت ها، متحول ها و انواع دیتا Constants, Variables and Data Types
۲۰.....	۱.۹ تطبیق زبان سی پلس پلس (Application of C++).....
۲۱.....	۱.۱۰ ساختار ابتدایی یک برنامه C++.....
۲۹.....	۱.۱۱ اجزای C++ (C++ Token).....
۲۹.....	۱.۱۱.۱ دستورهای کلیدی (Keywords).....
۳۰.....	۱.۱۱.۲ شناسه ها (Identifiers).....
۳۱.....	۱.۱۱.۳ ثابت ها (Constants).....
۳۸.....	۱.۱۲ مفهومی Escape Sequences.....
۳۹.....	۱.۱۳ متحول ها (Variables).....
۴۳.....	۱.۱۴ معرفی متحول ها (Declaring Variables).....

۵۳.....	Operators and Expressions	فصل چهارم: عمل گر ها و عبارت ها
۵۵.....	(Unary Operators)	عمل گر های یک تایی
۵۶.....	(Arithmetic Operators)	عمل گر های حسابی
۵۸.....	(Mixed – Mode Arithmetic Operations)	عملیات مختلط حسابی
۵۸.....	(Precedence of Arithmetic Operators)	اولویت بندی عمل گر های حسابی
۶۰.....	(Relational Operators)	عمل گر های ارتباطی
۶۲.....	(Logical Operators)	عمل گر های منطقی
۶۴.....	(Assignment Operators)	عمل گر =
۷۲.....	(Expressions)	عبارت ها
۷۳.....	(Statements)	بیانیه ها
۷۴.....	(Evaluation of Expressions)	ارزیابی عبارت ها
۸۶.....	Conditional Statements	فصل پنجم: بیانیه های تصمیم گیری یا شرطی
۹۲.....	THE IF () ELSE STATEMENT	بیانیه شرطی
۱۰۲.....	NESTED IF () STATEMENT	یک شرط داخل شرط دیگر
۱۱۲.....	THE ELSE – IF ()	بیانیه
۱۱۷.....	switch	بیانیه شرطی
۱۲۴.....	(The Conditional Operator)	عمل گر شرطی
۱۲۷.....	goto	بیانیه
۱۳۳.....	Loop Statements	فصل ششم: حلقه ها
۱۳۶.....	THE FOR (; ;)	بیانیه
۱۴۴.....	THE WHILE ()	بیانیه
۱۵۱.....	THE DO – WHILE ()	بیانیه
۱۶۳.....	(Jump in Loops)	پرش در حلقه ها
۱۶۹.....	Array	فصل هفتم: صف یا
۱۷۱.....	(Array Defination)	معرفی صف
۱۷۶.....	(Initializing of one Dimensional Array)	قیمت گذاری یک صف یک بعدی
۱۷۷.....	(Processing of one Dimensional Array)	پروسس یک صف یک بعدی
۱۹۴.....	(Function)	فصل هشتم: تابع
۱۹۵.....		توابع کتابخانه ای
۱۹۶.....		توابع تعریف شده توسط کاربر
۱۹۸.....		نوشتن توابع
۲۰۳.....		روش های ارسال پارامتر ها به توابع
۲۱۰.....		مأخذ

مقدمه

طوری که همه ما می‌دانیم در این اواخر علم کامپیوتر رشد و توسعه چشم‌گیری نموده، و با گذشت هر روز در حال تغییر و پیشرفت می‌باشد. کامپیوتر و تکنالوژی‌های اطلاعاتی نقش بسیار مهم و ارزنده را در زندگی روزمره بشر ایفا می‌نماید.

هدف از تألیف این کتاب این است که بتواند نیازهای محصلان را در بخش اساسات نویسی فراهم نماید. کتاب اساسات نویسی با در نظر داشت مفردات جدید که توسط دیپارتمنت‌های متذکره اخیراً آماده شده تألیف گردیده است. کتاب با لازم لسان ++C تألیف گردیده است. مفاهیم اساسی نویسی و ++C در این کتاب به زبان ساده و عام هدف این که توانایی‌های محصلان را در بخش حل مسایل (Problem solving) با فراهم نمودن دانش و مهارت‌های فهم بیان گردیده است. در هر فصل این کتاب یک تعداد مثال‌ها ذکر گردیده تا درک مفاهیم را ساده‌تر سازد.

فصل اول در رابطه به مفاهیم اساسی برنامه نویسی، تحلیل و دیزاین حل مسأله. فصل دوم در مورد الگوریتم و فلوچارت. فصل سوم در رابطه به اهمیت و تطبیق لسان ++C، ساختار ابتدایی یک ++C. فصل چهارم در رابطه به ثابت‌ها، متحول‌ها، دستوره‌های کلیدی و انواع دیتا. فصل پنجم در رابطه به عمل‌گرها، انواع مختلف عمل‌گرها، حق اولویت بندی عمل‌گرها، عبارات و بالآخره توابع کتاب‌خانه. فصل ششم در مورد تنظیمات ورودی و خروجی، فایل صدی، عمل‌گرهای ورودی و خروجی. فصل هفتم در رابطه به بیانیه‌های تصمیم‌گیری مانند if، switch و عمل‌گر شرطی The Conditional Operator. فصل هشتم در مورد بیانیه‌های حلقوی مانند while، do while، for، Nested loop و Jump in loops. فصل نهم در رابطه به صف و انواع صف فصل دهم در مورد توابع، نیاز برای توابع، ساختار توابع و ارگومنت‌های حقیقی و رسمی.

هر فصل با مقدمه، اهداف آموزشی در ابتداء و به تعقیب آن به موضوعات اساسی فصل پرداخته شده است، به هر موضوع به اندازه نیاز شرح داده شده، خلاصه مطالب هر فصل در قسمت اخیر آن آورده شده و پرسش برای ارزیابی مطالب فصل در آخر آن گنجانیده شده است و فصل‌ها با معرفی فهرست مآخذ پایان یافته است.

امیدوارم خداوند متعال این تلاش ناچیز را قبول نموده و وسیله گسترش علم و دانش در بخش برنامه نویسی کامپیوتر در کشور عزیزمان بگرداند.



هدف کلی کتاب

آشنایی با اساسات پروگرام نویسی و زبان C++

فصل اول

معرفی زبان‌های برنامه نویسی Introduction to Programming Languages



هدف کلی: مصحلان با زبان‌های برنامه نویسی آشنا شوند.

اهداف آموزشی: در پایان این فصل مصحلان قادر خواهند بود تا:

۱. برنامه را تعریف نمایند.
۲. تاریخچه برنامه نویسی را بیان نمایند.
۳. انواع زبان‌های برنامه نویسی را توضیح دهند.

این فصل حاوی مفاهیم اساسی و ابتدایی برنامه نویسی مانند اهمیت، تعریف و تاریخچه برنامه نویسی و مترجمان برنامه نویسی می باشد، که عناوین پی یک دیگر به تفصیل مورد بحث قرار گرفته است.

کمپیوتر وسیله الکترونیکی است که غرض حل مسایل مختلف استفاده می شود. هدف از مسایل (Problems) مشکلات یا چالش های است که برای آن ها راه های حل با استفاده از کمپیوتر دریافت می نماییم. برای روشن شدن مفهوم مسأله مثال های ذیل را در نظر گیرید:

- مسأله (Problem) در بخش انجینیری می تواند طراحی یک ماشین و یا طراحی نقشه یک تعمیر باشد.

- مسأله در بخش طب می تواند تشخیص دقیق امراض، کمپیوتری نمودن سیستم مدیریت شفاخانه ها و یا نگهدارهای انبار ادویه یک شفاخانه باشد.

- مسأله در بخش تجارت می تواند کمپیوتری (Computerize) نمودن داد و گرفت امور بانکی، فهرست اموال و کنترل بازار اسعار، محاسبه مالیات و یا محافظت حساب بانکی باشد.

پس گفته می توانیم که حل مسایل با استفاده از کمپیوتر یک پروسه پرچالش ولی در عین زمان مبتکرانه است؛ به خاطر این که این پروسه شامل تحلیل و تجزیه، پلان گزاری و گرفتن تصامیم منطقی و درست می باشد. قابل ذکر است که برای حل مسایل مختلف باید اقداماتی لازم و منطقی راه اندازی گردد تا به حل درست و مطلوب مسایل نایل گردیم.

۱.۱ برنامه نویسی

برنامه نویسی در حقیقت پروسه راه یابی حل یک مسأله می باشد. زبان برنامه نویسی برای نوشتن مجموعه از کدهای کمپیوتر استفاده می شود.

زبان برنامه نویسی مجموعه دستورالعمل های است که توسط آن به کمپیوتر دستور داده می شود تا عملیات خاص را اجرا نماید. این برنامه نویس است که با استفاده از زبان های مختلف برنامه نویسی کمپیوتر را هدایت نموده تا عملیات مطلوب را انجام دهد.

۱.۲ تاریخچه زبان های برنامه نویسی

در سال 1960 یک تعدادی از زبان های برنامه نویسی به میان آمد که هر یکی آن موارد استفاده خاصی خود را داشت. مردم به فکر این شدند که به جای آموختن چندین زبان برای اهداف متعدد چرا یک زبان را نیاموزند که در تمام بخش ها مورد استفاده باشد. به همین خاطر کمیته بین المللی، زبان الگول 60 (Algol 60) را ایجاد نمود. اما این زبان خسته کن و بسیار عمومی بود و مورد پسند مردم قرار نگرفت و سبب

گردید تا زبان جدیدی به نام زبان ترکیبی کامپیوتر (CPL)¹ توسط پوهنتون کامبریج به میان آید. بعداً دیده شد که آموختن و تطبیق این زبان بسیار دشوار است. به تعقیب آن، زبان دیگر به نام زبان ترکیبی ابتدایی کامپیوتر (BCPL)² به میان آمد به امید این که مشکلات فوق را نداشته باشد. اما دیده شد که این زبان ضعیف و در عرصه‌های خاص مورد استفاده قرار می‌گرفت. در این وقت زبان دیگر تحت نام B به وجود آمد که برای سیستم عامل یونکس (Unix) توسط کین تامسن (Ken Thompson) در لابراتوار AT و Bill نوشته شد.

در سال 1972 شخصی دیگر به نام دینس ریچی (Dennis M Ritchie) با ترکیب نمودن بعضی مشخصات لسان B و BCPL و مفکوره‌های خودش یک زبان جدید را به نام زبان C به میان آورد. C یک زبان عمومی برنامه نویسی می‌باشد. این زبان به هدف خاصی که بتواند کدهای سیستم عامل یونکس را برای بار دوم نوشته کند تهیه شده بود (Aitken, Jones, 2014).

در سال 1979 برنامه نویس به نام Bjarne Stroustrup در لابراتوار Bell ایالت نیوجرسی آیالات متحده امریکا کار را روی لسان C++ را آغاز نمود. لسان C++ شکل انکشاف یافته C بوده و از جمله زبان‌های شی‌گرا (Object Oriented Programming Language) محسوب می‌گردد. این زبان دارای محیط گرافیکی و دوستانه بوده و آموختن آن نسبتاً آسان می‌باشد.

طوری که قبلاً ذکر گردید C++ شکل پیش‌رفته‌تر زبان C می‌باشد و می‌تواند تقریباً تمامی برنامه‌های که در C قابل اجرا است در این زبان هم اجرا گردد. عمده‌ترین برتری زبان C++ نسبت به C این است که C++ مفاهیم کلاس‌ها (Classes)، وراثت (Inheritance)، توابع (Functions) را دربر دارد. این ویژگی‌های خاص باعث شده که انواع دیتاهای مجمل (Abstract Data Types) را ایجاد و خصوصیات (Attributes) کلاس‌ها را به ارث برده و مفکوره چندین شکلی (Polymorphism) را حمایت نماید (Aitken & Jones, 2014).

به صورت عموم، زبان‌های برنامه نویسی به پنج دسته تقسیم گردیده که عبارت‌اند از:

۱. زبان ماشین؛
۲. زبان‌های سطح پایین؛
۳. زبان‌های سطح بالا؛
۴. زبان‌های نسل چهارم؛

¹ Combine Programming Language

² Basic Combined Programming Language

۵. زبان‌های نسل پنجم.

حال هریکی از این دسته زبان‌ها را به بحث می‌گیریم:

زبان ماشین

عبارت از زبانی است که از 0 و 1 تشکیل گردیده و توسط کامپیوتر قابل فهم می‌باشد. به این معنا که ضرورت نیست کدهای 0 و 1 توسط یک مترجم ترجمه گردد تا کامپیوتر آن را بفهمد.

زبان‌های سطح پایین

زبان‌های است که بیش‌تر مشابهت به زبان ماشین دارد. دستورهای این زبان‌ها به شکل سمبول‌های خاص است که با استفاده از آن می‌توانید برنامه‌های مختلف را تهیه نمایید. زبان اسمبلی (Assembly) مربوط به زبان‌های سطح پایین می‌باشد. قابل یادآوری است که مترجم اسمبلر (Assembler) به‌خاطر ترجمه عبارت‌های زبان اسمبلی به زبان ماشین استفاده می‌گردد.

درک عبارت‌های سطح پایین برای پرزدهات کامپیوتر نظر به انسان ساده‌تر است زیرا عبارت‌های این زبان‌ها بیش‌تر شباهت به زبان ماشین دارد.

زبان‌های سطح بالا

زبان‌های است که عبارت‌های آن به زبان‌های بشر تهیه و ترتیب می‌گردد. به‌همین خاطر، نوشتن برنامه‌ها در این زبان‌ها برای برنامه‌نویس آسان می‌باشد. قابل یادآوری است که کامپیوتر عبارت‌های این نوع زبان‌ها را نمی‌شناسد و ضرورت به ترجمه دارد. به‌همین دلیل است که این نوع زبان‌ها برای ترجمه نمودن عبارت‌های خویش از مترجم کمپایلر (Compiler) و یا انترپریتر (Interpreter) استفاده می‌نمایند. زبان‌های مانند BASIC، COBOL، C++ از جمله زبان‌های سطح بالا می‌باشد.

زبان‌های نسل چهارم

این زبان‌ها نیز از جمله زبان‌های سطح بالا می‌باشد ولی برنامه‌نویسی در این نوع زبان‌ها آسان‌تر است. این نوع زبان‌ها بیش‌تر غرض ایجاد دیتابیس‌ها استفاده می‌گردد. زبان FoxPro، MYSQL و Oracle از جمله زبان‌های نسل چهارم می‌باشد.

زبان‌های نسل پنجم

این نوع زبان‌ها نیز مربوط به زبان‌های سطح بالا می‌باشد. تفاوت خاص این نوع زبان‌ها با زبان‌های دیگر، در این است که برنامه نویسی در این زبان‌ها آسان است زیرا زبان‌های نسل پنجم از جمله زبان‌های گرافیکی است. زبان Java، Visual BASIC و ++C از جمله زبان‌های شی گرا محسوب می‌گردد.

۱.۳ مترجمان زبان (Language Translators)

طوری که همه می‌دانیم کامپیوتر تنها 0 و 1 را می‌شناسد. به عبارت دیگر، لسان کامپیوتر 0 و 1 می‌باشد. وقتی که یک کد یا بیانیه (Statement) به زبان‌های سطح بالا نوشته می‌گردد؛ کامپیوتر آن را نمی‌فهمد و نیاز است که کد متذکره به زبان 0 و 1 تبدیل گردد تا کامپیوتر آن را بداند و عمل مطلوب را انجام دهد. به همین خاطر تمام زبان‌های برنامه نویسی دارای مترجمان اند که کدهای سطح بالا را به زبان ماشین (0 و 1) تبدیل می‌نماید.

به صورت عموم، سه نوع مترجمان وجود دارد که ذیلاً به آن‌ها می‌پردازیم:

۱.۳.۱ اسمبلر (Assembler)

این مترجم کدهای زبان اسمبلی را به زبان ماشین (0 و 1) تبدیل می‌نماید. مترجم اسمبلر تنها در زبان اسمبلی مورد استفاده قرار می‌گیرد. زبان اسمبلی یکی از زبان‌های سطح پایین بوده و نوشتن برنامه در آن نیز مشکل می‌باشد. به همین خاطر از زبان اسمبلی فعلاً استفاده زیاد صورت نمی‌گیرد.

۱.۳.۲ مترجم (Compiler)

این مترجم کدهای زبان‌های سطح بالا را به زبان کامپیوتر (0 و 1) تبدیل می‌نماید. لسان ++C و یک تعداد زبان‌های دیگر برنامه نویسی از مترجم Compiler استفاده می‌نمایند. این مترجم تمام کدهای یک برنامه را در اخیر بعد از این که برنامه نویس روی دکمه Run کلیک نماید به 0 و 1 تبدیل می‌نماید. اگر در برنامه اشتباه وجود داشته باشد طور مثال سیمی کولن (؛) در اخیر یک بیانیه نوشته نشده باشد و یا کدام دستور به شکل نادرست نوشته شده باشد، در آن صورت مترجم Compiler اشتباهات برنامه را دریافت و مشخص نموده طی یک پیام روی سکرین نشان می‌دهد. برنامه نویس بعد از خواندن پیام، اشتباهات را رفع نموده و برنامه را دوباره اجرا (Run) می‌نماید. در صورت که تمام اشتباهات برنامه رفع شده باشد برنامه اجرا (Execute) گردیده و نتایج برنامه روی سکرین ظاهر می‌گردد.

۱.۳.۳ تفسیر کننده (Interpreter)

این مترجم برنامه را سطر به سطر ترجمه می نماید. به این معنا که وقتی برنامه نویس یک بیانیه را مکمل نوشته کند و دکمه Enter روی صفحه کلید را فشار دهد در آن صورت Interpreter به شکل اتومات سطر متذکره را ترجمه می نماید. اگر در بیانیه کدام مشکل وجود داشته باشد تفسیر کننده اتومات یک پیام را روی سکرین نشان می دهد و اشتباه موجود در همان سطر را نشانی می نماید که باید اصلاح گردد در غیر آن کرزر به سطر بعدی نمی رود و یا هم سطر متذکره را به رنگ سرخ نشان می دهد. بعد از رفع اشتباه کرزر به سطر بعدی رفته و رنگ نوشته نیز درست (سیاه) می شود.

همین طور وقتی برنامه نویس در سطر دوم بیانیه را نوشته می کند بعد از فشار دادن کلید Enter در صورت وجود اشتباه در سطر جاری تفسیر کننده یک پیام را نشان می دهد و اگر کدام اشتباه وجود نداشته باشد در آن صورت کرزر به سطر بعدی رفته برنامه نویس می تواند بیانیه بعدی را نوشته کند.

پس گفته می توانیم که سرعت مترجم Compiler نسبت به Interpreter بیش تر است به خاطر Compiler برنامه را در اخیر به یک باره گی اجرا می نماید.



در این فصل مسایل مختلف که با استفاده از کمپیوتر حل می‌شود مورد بحث قرار گرفت و در نتیجه مفاهیم ذیل را آموختیم:

برای این که مسایل مختلف را با استفاده از کمپیوتر حل نماییم، لازم است که مسأله به شکل واضح و روشن تعریف گردد. با استفاده از زبان‌های برنامه‌نویسی مسئله را پروگرام نموده به حل کمپیوتری آن نایل می‌شود.

مبحث دوم که در این فصل توضیح گردید تاریخچهٔ زبان‌های برنامه‌نویسی می‌باشد. از سال 1960 تا حال زبان‌های مختلف برنامه‌نویسی به میان آمده که هر یکی آن دارای خصوصیات متفاوت می‌باشد. در حال حاضر، بیش‌تر تلاش می‌گردد که غرض حل مسائل مختلف از زبان‌های شی‌گرا (Object Oriented Languages) استفاده صورت گیرد به‌خاطر که این نوع زبان‌ها دارای محیط گرافیکی و دوستانه بوده و کار کردن در آن نیز آسان می‌باشد. مبحث نهایی این فصل مترجمان لسان‌های برنامه‌نویسی می‌باشد که کدهای زبان‌های سطح بالا را به زبان ماشین (0 و 1) تبدیل می‌نماید. مترجمان انواع مختلف دارند که هر یکی آن دارای مشخصات جداگانه می‌باشد.



سوالات و فعاليت های فصل اول

۱. مسئله را تعريف نماييد؟
۲. برنامه را تعريف نماييد؟
۳. تاريخچه برنامه نويسي را توضيح دهيد؟
۴. نسل های زبان های برنامه نويسي را نام بگيريد؟
۵. قبل از وجود زبان های برنامه نويسي، مردم مسایل مختلف را چگونه حل می نمود؟

فعاليت ها

۱. اهميت برنامه نويسي در حل مسایل روزمره را بحث نماييد؟
۲. تفاوت بين زبان های سطح پايين و سطح بالا را توضيح دهيد؟

فصل دوم

الگوریتم و فلوچارت Algorithm and Flowchart



هدف کلی: با الگوریتم‌ها و فلوچارت‌ها آشنا شوند.

اهداف آموزشی: در پایان این فصل محصلان قادر خواهند شد تا:

۱. الگوریتم را تعریف نمایند.
۲. الگوریتم یک برنامه را نوشته بتوانند.
۳. فلوچارت‌ها را بیان نمایند.
۴. فلوچارت‌ها برای برنامه‌ها را ترسیم بتوانند.

این فصل حاوی مفاهیم الگوریتم، فلوچارت و نقش آن در برنامه نویسی می باشد که عناوین پی یک دیگر به تفصیل مورد بحث قرار گرفته است.

دو وسیله که به خاطر دیزاین حل مسایل بیش تر مورد استفاده قرار می گیرد عبارت اند از:

۱- الگوریتم

۲- فلوچارت

الگوریتم (Algorithm)

کلمه الگوریتم از نام ریاضی دان مشهور عربی قرن نهم به نام الخوارزمی گرفته شده است. این نام در طی سال ها از الخوارزم (Alkhowarism) به الگوریزم (Algorism) و بالاخره به الگوریتم (Algorithm) تبدیل شده است.

الگوریتم مجموعه عبارت های ساده است که به زبان انگلیسی و عبارت های حسابی نوشته می شود (Klousen, 2017).

الگوریتم یک پروسیجر مرحله وار است که برای حل مسایل مختلف استفاده می گردد (Backman, 2012). به بیان دیگر، الگوریتم عبارت از حل مرحله وار یک مسأله است که به آسانی قابل درک می باشد. در حقیقت الگوریتم مجموعه مرا حل معین است که بعد از طی نمودن آن به نتیجه مطلوب می رسیم (Lippman, 2002).

۱.۴ خصوصیات الگوریتم (Properties of Algorithm)

هر الگوریتم باید خصوصیات ذیل را داشته باشد:

۱. ساده باشد؛
۲. مختص و صریح باشد.
۳. ورودی داشته باشد.
۴. حل یگانه (Unique) را برای مسأله پیش کش نماید.
۵. بعد از طی نمودن چند مرحله محدود به پایان برسد (نقطه انجام داشته باشد).
۶. حداقل یک خروجی داشته باشد.

به خاطر این که برای یک مسأله الگوریتم را طرح نماییم باید:

۱. مسأله را به شکل درست فهمیده باشیم.
۲. خروجی که به دست می آید معلوم باشد.

۳. ورودی آن مشخص باشد.
۴. پروسه که قیمت‌ها را وارد برنامه می‌سازد و در نتیجه خروجی تولید می‌گردد دیزاین شده باشد.
۵. صحت الگوریتم ارزیابی و امتحان شده باشد؛ به این معنا که دیتا به‌طوری امتحانی وارد الگوریتم گردیده باشد و نتیجه نیز بررسی شده باشد.
۶. اگر خروجی مورد نظر به‌دست نیامد در آن صورت الگوریتم باید بار دیگر مورد بررسی قرار گیرد.

۱.۵ علامت‌گذاری الگوریتم (Algorithm Notations)

در اثنای نوشتن الگوریتم علامت‌گذاری‌های ذیل باید در نظر گرفته شود:

- i. نام (Name): مسأله را مشخص می‌سازد.
- ii. شماره مرحله (Step Number): هر هدایت (عبارت) را توسط یک شماره خاص مشخص می‌نماید.
- iii. تبصره (Comments): عبارت‌ها و عملیات را توضیح می‌دهد؛ تبصره‌ها در داخل قوس‌های مربع شکل (square brackets) نوشته می‌شود.
- iv. آغاز/ خاتمه (Beginning / Termination): این بیانیه آغاز و خاتمه الگوریتم را نشان می‌دهد.

مثال ۱.۲: الگوریتمی را بنویسید که درجه حرارت را به سانتی‌گرید بگیرد و آن را به فارنهایت تبدیل نماید.

الگوریتم: تبدیل نمودن درجه حرارت از سانتی‌گرید به فارنهایت

مرحله 1: آغاز

مرحله 2: C را بخوانید.

مرحله 3: $F = 9/5 * C + 32$ را محاسبه کنید.

مرحله 4: F را نشان دهید.

مرحله 5: پایان.

مثال 2.2: الگوریتمی را بنویسید که مفاد را محاسبه کند به شرطی که سرمایه کلی، فیصدی مفاد و مدت زمان داده شده باشد.

الگوریتم: محاسبه مفاد ساده

مرحله 1: آغاز

مرحله 2: قیمت سرمایه کلی، فیصدی مفاد و مدت زمان را وارد کنید.

مرحله 3: مفاد را محاسبه کنید .
$$\frac{\text{ه یامسر ی لک} + \text{ی دصیف دافم} + \text{تدم نامز}}{100}$$

مرحله 4: اندازه مفاد را نشان دهید.

مرحله 5: پایان.

۱.۵.۱ فواید الگوریتم

1. ردیابی مشکلات آسان است زیرا تمام مراحل به شکل مرحله وار نوشته شده است.
2. تهیه نمودن الگوریتم آسان است به خاطر که نیاز به دانستن لسان های برنامه نویسی ندارد.
3. الگوریتم ما را کمک می کند تا برنامه را درست محافظت کنیم.




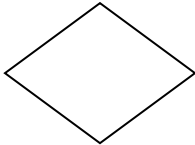


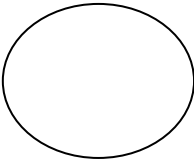

۱.۵.۲ نواقص الگوریتم

1. تهیه نمودن الگوریتم برای مسایل مغلق و پی چیده مشکل و زمان گیر است.
2. فهمیدن مسایل پی چیده (Complex logic) با استفاده از الگوریتم دشوار است.

۱.۶ فلوچارت (Flowchart)

بیان تصویری مرحله وار الگوریتم به نام فلوچارت یاد می گردد. فلوچارت شامل شیوه حل مسأله (Solution procedure)، محاسبات مرتبط (Relevant computations)، نکات تصمیم گیری (Points of decisions) و دیگر معلومات مهم می باشد. فلوچارت ها با استفاده از اشکال و سمبول های هندسی تهیه می گردد. هر شکل هندسی نشان دهنده عملیه خاص می باشد. این اشکال با استفاده از تیرها (Arrows) با هم دیگروصل گردیده که حل کامل یک مسأله بزرگ را نشان می دهد (Halterman, 2018).

جدول (۱-۲) اشکال مختلف هندسی و موارد استفاده آن را بیان می‌کند.

معنی	سمبول
آغاز / انجام (Start/End) این شکل نشان دهنده آغاز و انجام برنامه است.	
ورودی / خروجی (Input/output) این شکل به خاطر خواندن قیمت های ورودی و خروجی استفاده می‌گردد.	
پروسس کننده (Processing) تمام عملیات حسابی و فورمول ها در این شکل درج می‌گردد.	
تصمیم گیری (Decision) در این شکل مقایسه قیمت‌ها و تصمیم گیری صورت می‌گیرد.	
تکرار دستورها (Repetition) این شکل نشان دهنده این است که دستورهای داخل آن برای چندین بار تکرار می‌گردد.	
پروسه قبل تعریف شده (pre-defined process) این شکل نمایندگی از محاسبات می‌کند که قبل تعریف شده باشد. مثلاً گروپ عملیات که در کدام جای دیگر مشخص شده باشد.	
ارتباط دهنده (Connector) این شکل نشان دهنده نقطه ورود و یا خروج به بخش دیگر از فلوچارت است.	
تعیین کننده جهت (Director of flow) این اشکال جهت جریان پروسه را نشان می‌دهد.	

مثال ۳.۲: فلوچارت را رسم کنید که نمرات ۴ مضمون را وارد نموده، مجموعه و اوسط آن را دریافت نماید؟	مثال ۴.۲: فلوچارت را رسم کنید که مساحت مثلث را دریافت نماید و دارای اضلاع A، B و C باشد؟
<pre> graph TD Start([آغاز]) --> Input[/نمرات وارد شده چهار مضمون/] Input --> Sum[Sum = M₁ + M₂ + M₃ + M₄] Sum --> Avg[Avg = Sum / 4] Avg --> Print[/چاپ کنید/] Print --> End([پایان]) </pre>	<pre> graph TD Start([آغاز]) --> Input[/قیمت های وارد شده اضلاع/] Input --> S[S = A + B + C / 2] S --> Area[Area = √ S (S-A) (S-B) (S-C)] Area --> Print[/چاپ کنید/] Print --> End([پایان]) </pre>

۱.۶.۱ فواید فلوچارت (Advantages of Flowchart)

- حل مسایل مغلق و پی چیده را با استفاده از اشکال هندسی ساده تر می سازد.
- این شیوه، یک نوع مستند سازی (Documentation) برنامه است.
- با استفاده از فلوچارت برنامه به شکل موثر و آسان کد نویسی می گردد.
- شیوه خوب برای ردیابی و امتحان یک برنامه را فراهم می سازد.
- زمینه را برای برنامه نویس مساعد می سازد تا هر بخش مورد نظر برنامه را تغییر دهد.

۱.۶.۲ محدودیت های فلوچارت (Limitation of Algorithm)

- این شیوه برای نمایش مسایل بزرگ، مغلق و پی چیده موثر نیست، زیرا فلوچارت هم چنین مسایل را به شکل واضح بیان کرده نمی تواند.
- اگر کدام دستور شکل حلقوی (Loop) داشته باشد، لازم است تا اشکال هندسی مکرر ترسیم گردد.

۱.۶.۳ قواعد ترسیم فلوچارت ها (Rule for writing flowcharts)

- فلوچارت از طرف بالا به پایین و یا راست به چپ رسم می گردد.
- فلوچارت همیشه با سمبول آغاز شروع و با سمبول ختم پایان می یابد.
- تیرهای جهت دار به خاطر وصل نمودن اشکال هندسی در فلوچارت استفاده می گردد.
- در هر فلوچارت حداقل یک سمبول که خاتمه فلوچارت را نشان دهد باید وجود داشته باشد.
- شکل شرطی باید یک جهت ورودی و دو جهت خروجی داشته باشد.
- بیانیه ها و عبارت ها باید در داخل اشکال هندسی نوشته شود.

Chapter 2 انکشاف راه حل کمپیوتری (Development of Computer Solution)

a. کدنویسی (Coding)

کد نویسی در حقیقت ترجمه عبارت های الگوریتم و یا فلوچارت به زبان های برنامه نویسی (مانند C++، Pascal، Fortran، Java) و غیره می باشد.

b. امتحان و ردیابی (Testing and Debugging)

به خاطر که نتیجه مطلوب را به دست آوریم نیاز است برنامه که به شکل کد نوشته شده چک (Test)، ترجمه (Compile) و اجرا (Execute) گردد. در اثنای ترجمه کد ممکن یک تعداد اشتباهات در برنامه وجود داشته باشد که توسط مترجم (Compiler) شناسایی می گردد. این اشتباهات گرامری در اصطلاح برنامه نویسی به نام اشتباه املائی (Syntax errors) یاد می گردد. به خاطر که برنامه درست کار کند، لازم است تا این اشتباهات شناسایی و حل شود.

در اثنای اجرای برنامه ممکن یک سلسله اشتباهات در برنامه رونما گردد که به نام **اشتباهات اثنای اجرای برنامه** (Runtime errors) یاد می گردد. طور مثال، اگر یک عدد بالای صفر تقسیم گردد یک پیام مانند: "هیچ عدد صحیح بالای صفر تقسیم نمی گردد" را نشان خواهد داد. اشتباهاتی که در اثنای اجرای برنامه رونما می گردد در اصطلاح برنامه نویسی به نام Bugs یاد می شود (Halterman, 2018).

۱.۷ اشتباهات مفهومی (Semantic Errors)

عبارت ریاضی $Y = A + B$ را در نظر گیرید، دیده می شود که قیمت A و B باهم جمع شده و در متحول Y ذخیره می گردد. اگر عبارت فوق به این طور $A + B = Y$ نوشته گردد، قیمت A و B باهم جمع شده اما در متحول Y که در سمت راست علامه = قرار دارد ذخیره نخواهد شد. این نوع اشتباه را به نام اشتباه مفهومی (Semantic error) یاد می کند (Stroustrup, 2013).

Chapter 3 ردیابی اشتباهات (Debugging)

این پروسه حاوی شناسایی و پاک کاری اشتباهات (Bugs) می باشد. بعضی اوقات ممکن برنامه شاهد اشتباهات باشد که به نام اشتباهات منطقی (Logical errors) یاد می شود (Oualline, 1995).

مثال:

اگر مسأله یافتن **مساحت دایره** باشد به شرط این که قیمت شعاع معلوم باشد در آن صورت فورمول مساحت دایره عبارت است از:

$$\text{مساحت دایره} = \text{قیمت پای} * \text{شعاع} * \text{شعاع}$$

و یا:

$$\text{area} = 3.14 * r * r$$

اگر فورمول فوق به این شکل $\text{Area} = 2 * 3.14 * r$ نوشته شود واضح است قیمت خروجی که از این فورمول به دست می آید مطابق خواست ما نیست یعنی مساحت دایره را درست نمی کشد با وجود این که برنامه بیدون روبرشدن به هیچ نوع مشکل اجرا می گردد اما نتیجه فورمول درست نیست.

در هم چو حالت الگوریتم مسأله باید دوباره مرور گردد و اصلاحات لازم در آن به میان آید، بعدا در روشنی الگوریتم جدید برنامه باید دوباره طرح ریزی شود. در مرحله بعد لازم است که برنامه اجرا گردد تا دیده شود که نتیجه مطلوب به دست آمده یا خیر؟.

۱.۸ مستند سازی (Documentation)

هدف مستند سازی این است که یک مآخذ برای رهنمایی استفاده کننده (User manual) و یا نگهداری برنامه (Program maintenance) ترتیب گردد. این مستند سازی می تواند به شکل یک فایل جداگانه و یا فایل های هم کار در داخل برنامه وجود داشته باشد (Oualline, 1995).



در این فصل مسایل مختلف که با استفاده از کمپیوتر حل می‌شود، مورد بحث قرار گرفت و در نتیجه مفاهیم ذیل را آموختیم:

به‌خاطر دیزاین حل مسایل مختلف می‌توانیم از مفاهیم الگوریتم و فلوچارت استفاده نماییم. الگوریتم یک پروسیجر مرحله‌وار است که برای حل مسایل مختلف استفاده می‌گردد، در حالی که فلوچارت بیان تصویری الگوریتم را می‌گوید.



سوالات و فعالیتهای فصل دوم

۱. الگوریتم را تعریف نمایید؟
۲. مراحل که به خاطر تهیه الگوریتم ضروری اند کدامها اند؟
۳. فلوچارت را تعریف نمایید؟
۴. اشکال هندسی که در فلوچارت استفاده می گردد کدامها اند؟
۵. سمبول های که در فلوچارت استفاده می گردد کدامها اند؟
۶. اشتباه املائی (Syntax errors) چه مفهوم را ارایه می کند؟
۷. غلطی اثنای اجرای برنامه (Run time errors) چیست؟
۸. بقاء برنامه (Maintenance) چیست؟
۹. در فلوچارت چرا از اشکال هندسی استفاده می شود؟

فعالیت ها

۱. فواید و نواقص الگوریتم چیست، توضیح دهید؟
۲. وقتی الگوریتم وجود دارد چرا از فلوچارت نیز در حل مسایل مختلف استفاده می شود؟

فصل سوم

ثابت ها، متحول ها و انواع دیتا Constants, Variables and Data Types



هدف کلی: با ساختار C++، دستورهای کلیدی، ثابت ها، متحول ها و انواع دیتا در آن آشنا شوند.

اهداف آموزشی: در پایان این فصل محصلان قادر خواهند شد تا:

۱. ساختار برنامه C++ را بیان نمایند.
۲. عملگرهای ورودی و خروجی (cin / cout) C++ را شرح دهند.
۳. دستورهای کلیدی (Keywords) را شرح دهند.
۴. ثابت ها و انواع آن را تشریح نمایند.
۵. متحول (Variable) را تعریف نمایند.
۶. انواع مختلف دیتا را در برنامه های گوناگون استفاده نمایند.

در این فصل زبان برنامه‌نویسی سی پلس پلس، ساختار برنامه، دستورهای کلیدی، ثابت‌ها و انواع آن، متحول و انواع مختلف دیتا مورد بحث قرار گرفته است.

زبان برنامه‌نویسی C++ نسبت به زبان‌های دیگر برنامه‌نویسی، سطح بالا یک سلسله برتری‌های دارد که عبارت‌اند از (Rama, 2011):

- یک زبان قوی و توانمند است.
- بیش‌تر برای نوشتن نرم‌افزارهای سیستم (System software) و برنامه‌های تطبیقی (Application software) استفاده می‌گردد.
- دارای انواع متعدد نوعیت دیتا (Data types) و عمل‌گرها (Operators) می‌باشد که باعث سرعت و کارایی این زبان می‌گردد.
- دارای Platform آزاد و قابل انتقال می‌باشد. اکثریت برنامه‌های که در C++ نوشته می‌شود در تمام سیستم‌های عامل قابل اجرا می‌باشد.
- از جمله زبان‌های شی‌گرا (Object Oriented) می‌باشد.

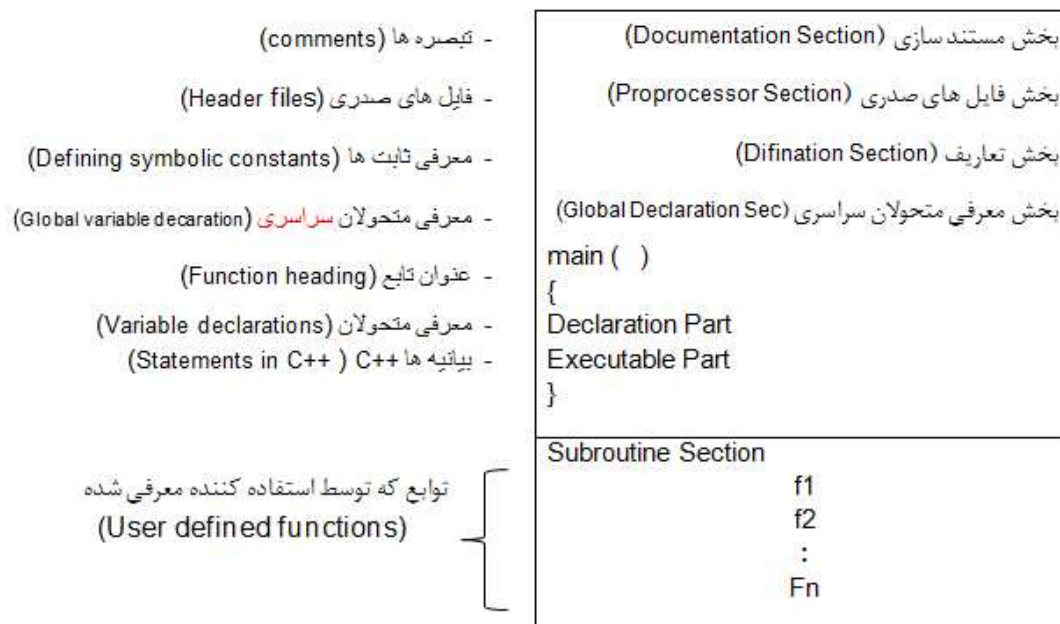
۱.۹ تطبیق زبان سی پلس پلس (Application of C++)

زبان C++ بیش‌تر برای انکشاف برنامه‌های سیستم و برنامه‌های تطبیقی مانند (Rama, 2011):

۱. مترجمین (Compilers)
۲. تفسیر کننده (Interpretors)
۳. سیستم‌های عامل (Operating Systems)
۴. سیستم‌های مدیریت دیتابیس (DBMS)
۵. برنامه ورد (MS Word)
۶. برنامه اکسپل (MS Excel)
۷. برنامه‌های گرافیک (Graphics packages)
۸. برنامه‌های ساینسی و انجینیری استفاده می‌گردد.

۱.۱۰ ساختار ابتدایی یک برنامه C++

یک برنامه C++ به چندین بخش تقسیم شده که ذیلاً توضیح می‌گردد (Rama, 2011):



شکل ۱.۳ ساختار یک برنامه C++

حال هر بخش برنامه، ذیلاً به تفصیل مورد بحث قرار می‌گیرد:

۱. بخش مستند سازی (Documentation Section)

این بخش دربرگیرنده مجموعه تبصره‌های مفید است که غرض رهنمود استفاده کننده به کار می‌رود. با وجود این که درج تبصره اجباری نیست اما استفاده کنندگان را در درک مفاهیم و اهداف برنامه‌ها و سطرهای آن کمک می‌کند.

۲. بخش پیش‌پرداز و تعریف آن (Preprocessor and Defination Section)

این بخش دربرگیرنده فایل‌های صدی است که با علامه # آغاز گردیده و در اخیر خویش **h** را نیز در بر دارد. این‌ها را به نام رهنمود پیش‌پردازنده (Preprocessor directive) یاد می‌کند که دربرگیرنده فایل صدی در C++ می‌باشد.

چند نمونه بیانیه Preprocessor عبارت اند از:

```
# include <iostream.h>
```

```
# include <math.h>
```

مثال: ثابت‌های سمبولیک (Symbolic constants) علامه # را در آغاز دستور define دارد.

```
define PI 3.14159 #
```

```
define e 2.71826 #
```

III. معرفی متحول‌های سراسری (Global Declaration Section)

در این بخش، متحول‌های (Variables) معرفی می‌گردد که در بیش‌تر از یک تابع مورد استفاده قرار می‌گیرد. این نوع متحول‌ها را به نام متحول‌ها سراسری (Global variables) یاد می‌کند که باید تحت فایل‌های صدری (Header files) معرفی گردد. یک برنامه ممکن چندین توابع را داشته باشد. فایده معرفی متحول‌ها سراسری در این است که شما می‌توانید آن‌را در هر تابع بدون این که دوباره در همان تابع معرفی گردد، استفاده کنید.

IV. تابع main (main Function)

تمام برنامه‌های C++ باید تابع (main) را داشته باشد. قابل یادآوری است که در هر برنامه فقط یک تابع (main) باید وجود داشته باشد.

V. قوس‌های بزرگ (Braces)

تمام برنامه‌های C++ حاوی تعداد قوس‌های بزرگ می‌باشد. اجرای برنامه از قوس باز تابع (main) آغاز و با قوس بسته تابع متذکره ختم می‌شود.

VI. بخش تعریف (Declaration Part)

در این بخش تمام متحول‌ها، توابع، صف یا Array و غیره معرفی می‌شود. هم‌چنان قیمت‌های اولیه (Initialization) نیز در همین قسمت معرفی می‌گردد.

VII. بخش اجرائیوی (Execution Part)

این بخش دربرگیرنده بیانیه‌های اجرائیوی مانند: بیانیه‌های ورودی و خروجی (Input/Output Statements)، بیانیه‌های حسابی (Arithmetic statements)، بیانیه‌های شرطی و حلقوی (Control statements) و غیره بیانیه‌ها می‌باشد. این بخش می‌تواند که تبصره‌ها (Comments) را نیز دربر داشته باشد. طوری که قبلاً گفته شد، مترجم از تبصره‌ها صرف نظر نموده و از جمله بیانیه‌های اجرائیوی نیز محسوب نمی‌گردد. بخش تعاریف (Declaration section) و بخش اجرائیوی باید در بین قوس‌های بزرگ قرار گیرد. هر بیانیه اجرائیوی توسط سیمی کولن (;) ختم می‌گردد (Balagtas, 2006).

VIII. بخش توابع فرعی (Subroutine Section)

این بخش اختیاری است و در برگیرنده آن‌عه توابعی می‌باشد که توسط برنامه نویس معرفی گردیده (User defined function) و از تابع اصلی () main اجرایی می‌گردد.

```
/* Program to print a message */
#include <iostream.h>

main ( )
{
cout <<" Programming in C++ is fun!! \n";
return 0;
}
```

Output

Programming in C++ is fun!!

نتیجه این برنامه عبارت است از:

Programming in C++ is fun!!

سطر اول مثال فوق مربوط به بخش مستند سازی (Documentation) می‌شود. این بخش دربرگیرنده مجموعه‌ای از تبصره‌های است که برای معرفی نام برنامه و توضیح کدهای برنامه استفاده می‌شود. سطر دوم به نام بیانیه پیش‌پرداز (<iostream.h> #include) preprocessor یاد می‌گردد که رابطه را با کتاب‌خانه سیستم به وجود آورده و دستور خروجی cout را کمک نموده تا نتیجه برنامه را روی سکرین نمایش دهد. سطر سوم مربوط به تابع اصلی (main) می‌باشد. تمام برنامه‌های C++ باید یک تابع اصلی داشته باشند. بخش باقی‌مانده تماماً مربوط به بخش اجرایی می‌شود که در آن بیانیه‌های اجرایی قرار دارد. بخش معرفی (declaration section) و بخش اجرایی (execution section) باید در بین قوس‌های جفت {} قرار گیرد. هر بیانیه بخش معرفی و اجرایی با سیمی کولن اختتام می‌یابد. یک برنامه از قوس باز { که بعد از تابع اصلی (main) قرار دارد آغاز و با قوس بسته } که در آخر تابع قرار دارد ختم می‌گردد (Nawaz, 2006).

تبصره در C++ (Comments in C++)

در C++ دو نوع تبصره (comments) وجود دارد.

- **تبصره یک سطر (Singleline Comment):** هر وقتی که می‌خواهید در برنامه خویش تبصره‌های یک سطر داشته باشید در آن صورت از singleline comments استفاده می‌توانید. این نوع تبصره همیشه با دو سلس چپه (//) آغاز می‌گردد.

\\ This is my first program in C++

مثال:

تبصره چند سطری (Multiline Comments): زمان که می‌خواهید در برنامه خویش تبصره چند سطری داشته باشید در آن صورت از multiline comments استفاده نموده می‌توانید. این نوع تبصره همیشه با استفاده از یک سلس و ستاره (/*) آغاز و به علامه ستاره و سلس (/) در اخیر تبصره بسته می‌گردد.

طور مثال:

```
/* This is
my first
program in C++ */
```

لازم به تذکر است که عبارت‌ها تبصره قابل اجرا نیست (non-executable) و توسط کمپایلر صرف نظر (Ignore) می‌شود. یک برنامه نویس می‌تواند که چندین تبصره را در برنامه خویش داشته باشد. هدف تبصره این است که مفهوم عبارات و کدهای مختلف را برای دیگران واضح و روشن سازد. کوشش شود که تبصره‌ها مختصر و واضح باشد.

عمل گر خروجی (Output Operator - cout)
بیانیه ذیل را در نظر گیرید:

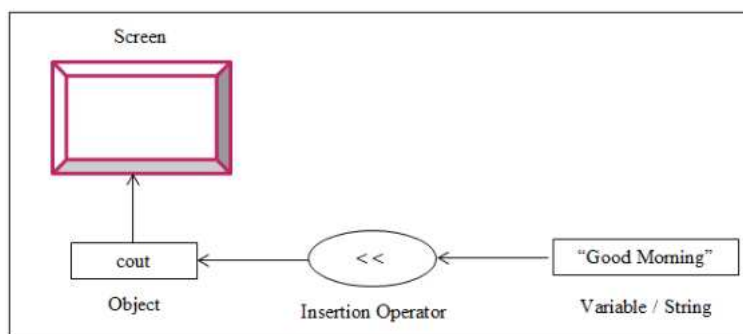
```
"cout << " Programming in C++ is fun
```

در این جا عمل گر cout برای این که قیمت خروجی را به بخش نمایش (مانند صفحه نمایش) ارسال نماید، استفاده شده است؛ cout مخفف console output می‌باشد.

cout به شکل (سی اوت (C - Out)) تلفظ می‌گردد و یکی از عمل گرهای از قبل معرفی شده (pre-defined) است که به خاطر نمایش نتایج استفاده می‌گردد (Nawaz, 2006).

به صورت عموم، قیمت که غرض نمایش به cout ارسال می‌گردد به شکل سلسله حروف تبدیل می‌شود.

عمل گر << به نام insertion یا put to یاد می‌شود که محتویات سمت راست خود را به object سمت چپ خویش ارسال می‌نماید. شکل ذیل را ملاحظه کنید:



شکل ۲.۳ نمایش خروجی با استفاده از cout

مثال ۲.۳: برنامه ذیل قیمت حروفی را با استفاده از cout روی سکرین نشان می‌دهد.

```
#include <iostream.h>
#include <string> // used for string data type
main ( )
{
    // Declaration
    string First_Name = "Habibullah ";
    string Last_Name = "Barakzai ";
    cout << "The Complete Name is : " << First_Name << Last_Name;
    return 0;
}
```

Output

The Complete Name is : Habibullah Barakzai

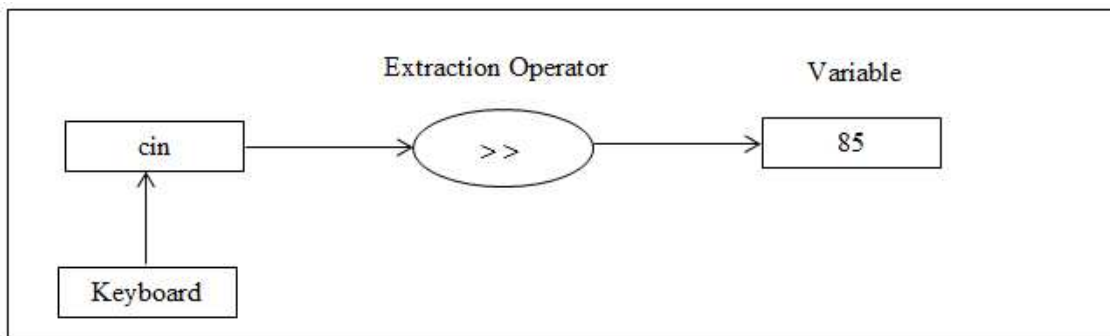
عمل‌گر دخولی (Input Operator – cin)

بیانیه ذیل را در نظر بگیرید:

```
cin >> marks ;
```

عمل‌گر cin در لسان C++ جهت وارد نمودن دیتا از طریق کیبورد استفاده می‌شود. Cin مخفف Console input می‌باشد. عمل‌گر cin سبب می‌شود که برنامه فوق منتظر بماند تا استفاده‌کننده قیمت marks را از کیبورد وارد نماید. قیمت وارد شده بعداً در متحول به نام marks ذخیره می‌شود. cin به شکل (سی‌اِن - in) تلفظ می‌شود و یکی از عمل‌گرهای از قبل معرفی شده (pre - defined) می‌باشد که به‌خاطر ورود دیتا از صفحه کلید (Keyboard) استفاده می‌شود (Rama, 2012).

عمل‌گر << به نام Extraction یا get from یاد می‌شود که دیتا را از کیبورد می‌گیرد و آن را در متحول سمت راست خود ذخیره می‌نماید. شکل ذیل را ملاحظه نمایید:



شکل ۳.۳ ورودی با استفاده از cin

مثال ۳.۳: برنامه‌ذیل نام، تخلص و عمر یک شخص را از صفحه کلید گرفته روی سکرین نشان می‌دهد.

```

#include <iostream.h>
#include <string> // used for string data type
main ( )
{
    // Declaration
    string FName;
    string LName;
    int Age;
    cout << " Enter your first and second name and your age please : ";
    cin >> FName >> LName >> Age;
    cout << "Your full name is : " << FName << " " << LName << "\n";
    cout << "You are " << Age << " years old";
    return 0;
}
  
```

Output

```

Enter your first and second name and your age please : Asma Gul 8
Your full name is : Asma Gul
You are 8 years old
  
```

تکرار عمل گرهای ورودی و خروجی (CASCADING OF I/O OPERATORS)

هرگاه عمل گر << و >> چندین بار در یک بیانیه استفاده شود این حالت به نام تکرار عمل گرها یاد می شود. عمل گر خروجی << می تواند به طور ذیل تکرار گردد (Halterman, 2018):

```
cout << " Simple Interest = " << SI << " \n";
```

به طور مشابه، عمل گر ورودی >> می تواند چنین تکرار گردد:

```
cin >> Prin >> Rate >> Year;
```

در دستور ورودی cin قیمت از سمت چپ به راست به متحول ها توزیف می گردد. طور مثال، قیمت اول در متحول Prin، قیمت دوم در متحول Rate و قیمت سوم در متحول Year ذخیره می گردد. بهتر است که بین عمل گر << و متحول ها که در مقابل آن درج می شود به اندازه یک سپیس فاصله اضافه گردد تا نتیجه را به طور واضح روی سکرین نشان دهد.

مثال ۴.۳: برنامه ذیل مساحت مثلث قائم الزاویه را محاسبه می نماید.

```
// Program finds area of triangle
#include <iostream.h>
main ( )
{
    double Base, Hight, Area;
    cout << " Enter value for Base and Hight of a triangle: " ;
    cin >>Base >> Hight;
    Area = Base * Hight / 2;
    cout << " The area of Triangle = " << Area;
}
```

Output

```
Enter value for Base and Hight of a triangle : 10    5
The area of Triangle = 25
```

مجموعه حروف (C++ Character Set)

مجموعه حروف حاوی، مجموعه سمبول ها مانند: الفبا، ارقام و سمبول های خاص (Special symbols) می باشد که به خاطر ارایه معلومات استفاده می گردد (Rama, 2011).

جدول ۱.۳ مجموعه کرکترهای C++

حروف الفبا (Alphabets)	حاوی حروف بزرگ (A تا Z) و حروف کوچک (a تا z)
ارقام (Digits)	حاوی ارقام (0 تا 9)
<p>سمبول‌های خاص (Special symbols)</p>	<p>tilde ~ back quote ` exclamation ! hash # percentage % caret ^ ampersand & double quote " single quote (apostrophe) ' left parantheses (right parantheses) left braces { right braces } left bracket [right bracket] plus sign + minus sign - - asterisk * - slash / - underscore _ - equal sign = - backslash \ - semi colon ; - colon : - comma , - dot . -</p>

less than < - greater than > - question mark ? - vertical bar -	
شامل خالی‌گاہا (spaces)، (tab، \v)، (\\t) خط جدید (\\n) و (form feed)	فاصله‌های خالی (White spaces)

۱.۱۱ اجزای C++ (C++ Token)

یک برنامه C++ مجموعه عناصر است که به نام نشانه‌ها (Token) یاد می‌گردد. نشانه در حقیقت کوچک‌ترین عنصر یک برنامه است. در C++ انواع مختلف نشانه‌ها وجود دارد که در جدول ذیل لیست شده است. یک نشانه از یک یا چند حرف C++ ترتیب شده است.

جدول ۲.۳ نشانه‌ها

نشانه‌ها	موارد استفاده
دستورهای کلیدی (Keywords)	شامل do، int، for و غیره می‌شود
شناسه (identifiers)	شامل نام‌های متحول‌ها مانند sum، area، number ... می‌شود
ثابت‌ها (Constants)	شامل اعداد مانند 250، 25.5، -200 و غیره می‌شود
حروف (String)	شامل سلسله حروف مانند "College"، "2+3" و غیره می‌شود
عملگرها (Operators)	شامل علامت‌ها مانند +، -، *، / و غیره می‌شود
سمبول‌های خاص (Special Symbols)	شامل سمبول‌ها مانند !، #، [،]، {، } و غیره می‌شود

۱.۱۱.۱ دستورهای کلیدی (Keywords)

دستورهای کلیدی عبارت از کلماتی است که برای اهداف خاص استفاده می‌گردد. کلمات ریزرف شده در C++ وجود دارد که از قبل معانی معیاری و مشخص دارند و آن‌را می‌توانیم تنها برای اهداف مشخص استفاده نماییم. تمام دستورهای کلیدی باید به حروف کوچک نوشته گردد. این دستورها را نمی‌توانیم به حیث نام‌های متحول‌ها (variable names) استفاده کرد (Buard, 2005).

دستوری‌های کلیدی معیاری C++ قرار ذیل است:

Asm	Double	Not	Switch
Auto	else	operator	Template
break	enum	private	this
case	extern	protected	throw
catch	float	public	try
char	for	register	typedef
class	friend	return	union
const	goto	short	unsigned
continue	if	signed	virtual
default	inline	sizeof	void
delete	int	static	volatile
do	long	struct	while

۱.۱۱.۲ شناسه‌ها (Identifiers)

- ⇐ شناسه به نام‌های اطلاق می‌شود که برای متحول‌ها (variables)، توابع (functions) و صف (array) در برنامه استفاده می‌شود.
- ⇐ شناسه: سلسله یا مجموعه کرکترهای است که از حروف (Z - A)، (z - a)، ارقام (0 - 9) و اندرسکور (_) تشکیل گردیده باشد.
- ⇐ C++ یک لسان Case sensitive است که در آن ALFA، Alfa، alfa از هم‌دیگر متفاوت اند.
- ⇐ سمبول اندرسکور (_) به‌طور کل در وسط نام Identifier استفاده می‌گردد. مثلاً: F_Name
- ⇐ طول نام شناسه یا idenfier به صورت متوسط می‌تواند بین 8 الی 10 کرکتر باشد. در قسمت طول نام identifier محدودیتی وجود ندارد و می‌تواند طولانی‌تر باشد.
- ⇐ نام Idnetifiers به حرف آغاز می‌گردد و به تعقیب آن می‌تواند حروف، ارقام و یا ترکیب هر دو وجود داشته باشد. نام Identifiers نمی‌تواند به عدد آغاز گردد.
- ⇐ C++ یک لست دستورهای کلیدی مشخص دارد که آن را نمی‌توانید در جای دیگر برای مقاصد متفاوت استفاده کنید. به‌طوری مثال، شما نمی‌توانید int را به حیث نام متحول استفاده کرد.

شناسه های مجاز عبارت اند از:	Sum, avg _ marks, x1, y2, PINCODE, Total
شناسه های غیر مجاز عبارت اند از:	10th , S.I, total amount, Std – no

۱.۱۱.۳ ثابت‌ها (Constants)

هر قیمت که در اثنای اجرای برنامه قابل تغییر نباشد به نام ثابت یاد می‌گردد. لسان C++ دارای چندین نوع ثابت‌ها می‌باشد.

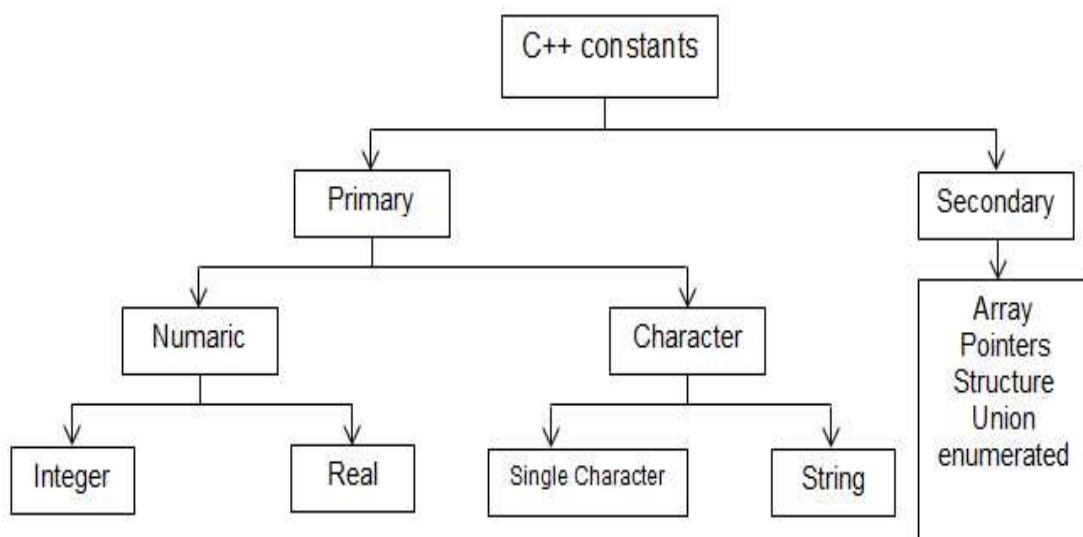
انواع ثابت‌ها (Types of Constants)

به صورت عموم ثابت‌های C++ به دو نوع تقسیم گردیده است که عبارت اند از:

۱. ثابت‌های ابتدایی

۲. ثابت‌های دومی

این ثابت‌ها به انواعی دیگر نیز تقسیم گردیده است که ذیلا بیان می‌شود:



شکل ۴.۳ انواع ثابت‌ها

ثابت‌های تام (Integer Constants)

ثابت‌های تام مربوط به ثابت‌های عددی (Numarical Constant) می‌باشد که از اعداد ثابت تام تشکیل گردیده است. در کل سه نوع ثابت‌های تام وجود دارد که عبارت‌اند از:

↔ قاعده 10 (Decimal system)

↔ قاعده 8 (Octal system)

↔ قاعده 16 (Hexa Decimal system)

جدول ۳.۳ خصوصیات ثابت‌های تام

تام به قاعده ۱۰	تام به قاعده ۸	تام به قاعده ۱۶
حاوی ارقام (0 الی 9)	حاوی ارقام (0 الی 7)	حاوی ارقام (0 الی 9) و حروف (A الی F) می‌باشد که حروف اعداد 10 الی 15 را نمایش می‌دهد.
حاوی حد اقل یک رقم باشد	اولین رقم باید 0 باشد	با 0x و یا 0X آغاز گردد
علامهٔ اعشاری نداشته باشد	علامهٔ اعشاری نداشته باشد	علامهٔ اعشاری نداشته باشد
می‌تواند حاوی اعداد مثبت و یا منفی باشد	می‌تواند حاوی اعداد مثبت و یا منفی باشد	می‌تواند حاوی اعداد مثبت و یا منفی باشد
کامه و یا خالی‌گاه باید بین ارقام وجود نداشته باشد	کامه و یا خالی‌گاه باید بین ارقام وجود نداشته باشد.	کامه و یا خالی‌گاه در بین ارقام وجود داشته نباشد.
مثال اعداد مجاز 426، 17+، 5050 است اعداد مجاز 10.0، 12.45 نیست	مثال اعداد 051، 044+، 025 مجاز است اعداد 068، 03.45، 540 مجاز نیست	مثال اعداد 0xA1، 0xa38، 0x1 مجاز است اعداد 0xEgg، 0ABC، 0x2.5 مجاز نیست

ثابت‌های اعشاری (Real Constants)

مقدار عددی که بخش اعشاری داشته باشد به نام ثابت‌های اعشاری یاد می‌گردد که خصوصیات ذیل را دارد:

↔ ثابت اعشاری حداقل باید یک رقم داشته باشد.

↔ دارای علامهٔ اعشاری باشد.

↔ می‌تواند عدد مثبت و یا منفی باشد.

↔ در بین عدد کامه و فاصله مجاز نیست.

مثال ۶.۳
325.66، 45.0، -38.2، 124.6789+ از جمله ثابت‌های مجاز اعشاری است در حالی که 10E-5، 530، 11 از جمله ثابت‌های غیر مجاز اعشاری می‌باشد.

اگر عدد اعشاری بسیار بزرگ و یا کوچک باشد در آن صورت از شیوه علمی (Scientific) که به نام شیوه نمادار (exponential) یاد می گردد، استفاده می شود. در این صورت عدد اعشاری در دو بخش نمایش داده می شود.

قیمت که قبل از e قرار دارد به نام مانتس (mantissa) و بخش که بعد از e قرار دارد به نام نما یا توان (Exponent) یاد می شود. طور مثال، عدد اعشاری 516.55 را در نظر بگیرید. این عدد را می توانیم به شکل $5.1655E^2$ به شیوه علمی یا Scientific نوشت. در E^2 قیمت E مساوی به 10 بوده و 2 نما یا توان قاعده E می باشد.

قواعد که جهت نمایش عدد اعشاری به شیوه علمی باید در نظر گرفته شود عبارت اند از:

- ↔ بخش مانتیس و بخش توان باید توسط حرف e از هم دیگر جدا شود.
- ↔ بخش مانتیس می تواند مثبت و یا منفی باشد.
- ↔ بخش توان باید حداقل یک رقم باشد که ممکن یک عدد تام مثبت و یا منفی باشد.
- ↔ ساحت قیمت اعداد اعشاری به شیوه علمی بین 3.4×10^{38} و -3.4×10^{38} می باشد.

مثال ۷.۳
$5 - 2.4E, -0.5e + 3, -4.5e8, -5.2e + 3$ از جمله ثابت های نمادار (Exponentials constant) مجاز در حالیکه $5.5 - 2e, 4.3E06$ - ثابت های نمادار غیر مجاز می باشد.

ثابت های حرفی (Character Constants)

- ↔ ثابت حرفی عبارت از یک حرف است که بین قوس ناخنک یگانه (single quote) قرار دارد.
- ↔ یک ثابت حرفی نمی تواند بیش تر از یک حرف یا کرکتر باشد.
- ↔ عملیات حسابی را می توان بالای ثابت های حرفی انجام داد به خاطر که تمام قیمت های آن به شیوه عددی ذخیره می شود.
- ↔ لسان ++C کرکتر های escape sequence مانند $\backslash t, \backslash n, \backslash b$ را می شناسد.

مثال ۸.۳
از جمله ثابت های مجاز حرفی می باشد در حالی که $'a', 'o', '=', ' '$ از جمله ثابت های غیر مجاز حرفی می باشد. $'26', 'a4', 'xy'$

توجه داشته باشید که ثابت حرفی '2' با عدد 2 یکسان نیست. هر حرف یا کرکتر یک کد ثابت که به نام ASCII code³ یاد می‌شود، دارد. جدول ذیل تعداد ثابت‌های حرفی و کدهای اسکی آنرا می‌دهد.

ثابت‌ها	کد اسکی
'A'	۶۵
'B'	۶۶
'Z'	۹۰
'a'	۹۷
'z'	۱۲۲
'\'	۴۹
'۲'	۵۰
'#'	۳۵

ثابت‌های حرفی (String Constants)

ثابت‌های حرفی عبارت از مجموعه‌ی حروف‌های است که تعداد آن بیش‌تر از یک حرف باشد و در داخل قوس ناخنک دوگانه (double quotes) درج می‌گردد. این حروف یا کرکترها که داخل قوس‌های ناخنک دوگانه قرار دارد شاید حروف، اعداد و یا سمبول‌های خاص و یا خالی باشد.

مثال ۹.۳
<p>“Read”, “Phone : 334 – 5670”, “AFN 20.25”, “”, “” باشد در حالی که</p> <p>“Line1 \n \ Line2 \n Line3”, “The Error” از جمله ثابت‌های غیر مجاز حرفی است.</p> <p>نوت: “ ” به نام ثابت حرفی خالی (null string) یاد می‌شود.</p>

متوجه باشید که “A” با “A” یکسان نیست. حروف ثابت که در داخل دو قوس ناخنک (“A”) وجود دارد در حقیقت قیمت تام عددی (ASCII code) ندارد.

Chapter 4 ثابت‌های سمبولیک یا ماکرو (Symbolic Constants)

#define از جمله رهنمودهای pre-processor است که غرض معرفی ثابت‌ها استفاده می‌شود (2007,

```
#define name replacement-
```

(Soulié).

³ American Standard Code for Information Interchange

در این جا (name) متحول است که می تواند قیمت ثابت عددی یا حروفی را در خود ذخیره کند. هر وقتی که برنامه اجرا می شود در هر جای که نام متحول (name) وجود داشته باشد قیمت که در آغاز برنامه به این متحول در نظر گرفته شده با آن تعویض می شود قابل یادآوری است که این متحول نمی تواند قیمتی متفاوت از آن که در آغاز برنامه برایش تعریف گردیده در خود ذخیره کند. در اصل #define متحول name را مقید می سازد تا قیمت که در آغاز برنامه توسط (user) برای آن تعریف گردیده در خود ذخیره نموده و هیچ قیمت متفاوت نگیرد.

چند مثال ثابت های سمبولیک:

```
#define Max_Marks 100
```

```
#define PI 3.14159
```

```
#define University "Kabul Education University"
```

مثال ۱۰.۳: برنامه ذیل شعاع یک دایره را دریافت می نماید در صورتی که قیمت مساحت موجود باشد.

```
// Program to find radius of a circle given area
#include <iostream.h>
# define PI 3.14159
main ( )
{
    float r, area ;
    cout << "\n " << " Enter the area of circle : " ;
    cin >> area ;
    r = sqrt ( area / PI ) ;
    cout << "\n" << "The radius of the given circle is : " << r ;
    return 0 ;
}
```

Output

Enter the area of the circle : 12 . 57

The radius of the given circle is : 2. 0029

Enter the area of the circle : 19 . 3

The radius of the given circle is : 2.47

قواعد که در اثنای معرفی ثابت‌های سمبولیک باید در نظر گرفته شود:

- ⇨ ثابت‌های سمبولیک معمولاً در قسمت آغاز برنامه تعریف می‌شود اما استفاده کننده می‌تواند در هر قسمت برنامه قبل از استفاده آن‌را معرفی نماید.
 - ⇨ `#define` با سیمی کولن ختم نمی‌گردد. اگر سیمی کولن در اخیر این بیانیه اضافه گردد مترجم آن‌را به حیث کرکتر ثابت محسوب خواهد نمود.
 - ⇨ گذاشتن سیمی کولن بین `#` و `define` مجاز نیست.
 - ⇨ بین `#define` و نام سمبولیک بهتر است به اندازه یک الی دو سپیس فاصله در نظر گرفته شود. به عین شکل بهتر است که بین نام سمبولیک و قیمت آن فاصله مدنظر گرفته شود.
 - ⇨ تعریف نام سمبولیک عین قواعد که برای معرفی نام یک متحول در نظر گرفته می‌شود مراعات می‌نماید. معمولاً، نام سمبولیک به حروف بزرگ نوشته می‌شود تا از نام‌های متحول‌های عادی تفکیک گردد.
 - ⇨ شما نمی‌توانید قیمت متفاوت از آن چه در ابتدای برنامه برای نام سمبولیک مدنظر گرفته شده در نظر گیرید.
 - ⇨ وقتی که برنامه اجرا می‌شود هر جای که نام سمبولیک وجود دارد، قیمت خود را به‌طور اتومات از قسمتی که تعریف شده می‌گیرد.
- طور مثال:

```
#define PI 3.14159
```

⇨

⇨

⇨

```
Curriculum = 2 * PI * Radius ;
```

.....

```
Cout << "\n PI value = " << PI ;
```

وقتی که برنامه فوق اجرا شود نام سمبولیک PI به قیمت آن تعویض می‌شود و بیانیه شکل ذیل را به خود می‌گیرد:

```
Curriculum = 2 * 3.14159 * Radius ;
```

فکر کنید که اگر ماکرو طوری ذیل معرفی شده باشد:

```
#define PI 3.14159 ;
```

یعنی در اخیر نام ماکرو سیمی کولن اضافه شده باشد در آن صورت بیانیه شکل ذیل را خواهد داشت:

Curriculum = 2 * 3.14159 ; * Radius;

این کار سبب می شود که برنامه به شکل درست اجرا نگردد.

بعضی از مثال های نادرست ماکرو قرار ذیل است:

```
# define      SUM = 0
# define      TOTAL 100 ;
# defineA     10 , B 20
# DEFINE      COUNT 10
```

⇐ اگر یک جمله یا قیمت حروف طولانی را در `#define` تعریف نماییم که چندین سطر را دربر گیرد، در آن صورت لازم است که در اخیر هر جمله کرکتر `\` (back slash) را اضافه نماییم و جمله متباقی را در سطر جدید نوشته کنیم. هدف نوشتن `\` در اخیر جمله به این معنا است که به مترجم گوش زد نماید که جمله جریان دارد.

مثال ۱۱.۳: برنامه ذیل شعاع یک دایره را در صورتی دریافت می کند که قیمت محیط دایره وجود

دارد.

```
// Program to find radius of a circle given the circumference
# include <iostream.h>
# define PI  3.14159

main ( )
{
    float r, circum ;
    cout << "\n" << " Enter the circumference of a circle : " ;
    cin >> circum ;
    r = circum / ( 2 * PI ) ;
    cout << "\n" << "The radius of the given circle is : " << r ;
    return 0 ;
}
```

Output

Enter the circumference of the circle : 8

The radius of the given circle is : 1 . 27

فواید ماکرو

- ⇨ استفاده از ماکرو، برنامه‌ها را ساده‌تر و منظم‌تر می‌سازد.
- ⇨ نام ماکرو باید پر مفهوم باشد و استفاده کننده را کمک کند که چه نوع قیمت را در خود ذخیره نموده است.
- ⇨ استفاده از ماکرو، برنامه را برای آوردن تغییرات آسان‌تر می‌سازد. این کار آسان است که قیمت ماکرو را تغییر داد به جای این که در هر موقعیت، قیمت متحول را تغییر دهد.
- ⇨ استفاده از ماکرو برنامه را خواناتر می‌سازد.

۱.۱۲ مفهوم Escape Sequences

بعد از کرکتر/ که به نام (backslash) یاد می‌گردد یک حرف خاص وارد می‌شود و یک عمل خاص را انجام می‌دهد؛ ترکیب این کرکترها به نام escape sequence یاد می‌شود (Deitel, 2012).

طور مثال:

```
cout << " \n this is easy !!" ;
```

بیانیه را در نظر بگیرید.

در این بیانیه `\n` نمونه از escape sequence می‌باشد و به خاطر استفاده شده که کرزر به سطر جدید انتقال یابد و بعد جمله `this is easy` را روی سکرین نشان دهد. اگر خواسته باشید که یک قیمت حروفی را در چند سطر چاپ کنید در آن صورت از `\n` استفاده کنید تا قیمت که بعد از کرزر وجود دارد در سطر جدید چاپ نماید. اگر قیمت حروف یا جمله در آخر خود `\n` را نداشته باشد در آن صورت جمله بعدی نیز به ادامه جمله قبلی در عین سطر ظاهر خواهد شد.

مثال ۱۲.۳: برنامه ذیل استفاده از `\n` را نمایش می‌دهد.

```
main ( )
{
    cout << "This \n string \n will \n be \n printed \n in \n 8 \n Lines. ";
    return 0 ;
}
```

Output

This
string

will
be
printed
in
8
Lines.

اگر خواسته باشید که بخشی از جمله طولانی در سطر بعدی ظاهر گردد در آن صورت n\ را در آغاز همان بخش مطلوب جمله بنویسید. به یاد داشته باشید که تمام سلسله escape sequence با \ آغاز می‌شود. دیگر escape sequence های که در C++ وجود دارد عبارت اند از:

جدول ۴.۳ توضیح کرکترهای escape sequence

وظیفه	escape sequence
برنامه در اثنای اجرای به صدا در می‌آید	\a
علامه سوالیه	?\
برگشت به اندازه یک space به عقب	\b
برگشت به آغاز پیام خروجی	\r
Form feed	\f
جمپ به اندازه یک Tab	\t
عمودی Tab	\v
کرکتر خالی اسکی	\0
رفتن به سطر جدید	\n
علامه double quotation	\"
Backslash	\\
خالی	\
قوس ناخنک یگانه (single quotation)	'\'

۱.۱۳ متحول‌ها (Variables)

متحول یک نام است که به قسمتی از حافظه RAM داده می‌شود و همان قسمتی از حافظه را تصرف (Reserve) می‌کند. وقتی که قیمت به متحول داده می‌شود در همان قسمتی ریزرف شده حافظه به شکل موقت ذخیره می‌شود (Klousen, 2017).

متحول تنها یک قیمت را در عین زمان در خود ذخیره می‌تواند. به این معنا، هر قیمت جدیدی که در یک متحول ذخیره می‌شود در حقیقت جای‌گزین قیمت قبلی می‌شود. این قیمت ممکن عددی و یا حروفی باشد. متحول می‌تواند در هر بار اجرای برنامه قیمتی مختلف را در خود ذخیره نماید.

مثال ۱۳.۳

بخش ذیل برنامه C++ را در نظر بگیرید:

```
int x, y, z;
char p;
:
:
X = 1;          /* value 1 is assigned to x */
Y = 2;          /* value 2 is assigned to Y */
Z = 3;          /* value 3 is assigned to Z */
X = Y + Z;      /* value X changes to 5 */
P = 'A'         /* value 65 is assigned to P */
Y = 6;          /* value of Y changes to 6 */
Z = 7;          /* value of Z changes to 7 */
P = 'a';        /* value of P changes to 97 */
```

از مثال فوق معلوم می‌شود که یک متحول می‌تواند قیمت‌های مختلف را در عین برنامه ذخیره کند.

قواعد نام‌گذاری متحول در C++

- ⇨ نام متحول باید به حرف آغاز گردد و به تعقیب آن حروف، اعداد و یا ترکیب هردو می‌آید.
- ⇨ در ترکیب نام متحول بدون اندر سکور (_) هیچ سمبولی دیگر اجازه نیست که استفاده شود.
- ⇨ زبان C++ در حقیقت case sensitive است. به همین خاطر counter، Counter و COUTNER سه متحول مختلف اند.
- ⇨ دستورهای کلیدی C++ نمی‌تواند به حیث نام متحول استفاده شود.
- ⇨ در بین حروف نام یک متحول اجازه گذاشتن فاصله (space) نیست.
- ⇨ بهتر است که نام متحول کوتاه و پرمفهوم باشد.

مثال ۱۴.۳

Sum_total, avg _ ht از جمله متحول‌های مجاز بوده در حالی که #No, 50th, Lotus 123 از جمله متحول‌های غیرمجاز محسوب می‌شود.

انواع دیتا (Data Types in C++)

زبان C++ چهار نوع دیتا دارد که عبارت اند از (2012, Backman):

↔ تام یا Integer: (int).

↔ حرف یا character: (char).

↔ اعشاری با دقت ساده floating (float).

↔ اعشاری با دقت مضاعف (double).

هر data type دارای خصوصیات و ظرفیت خاص می‌باشد که در جدول ذیل توضیح شده است.

جدول ۵.۳ توضیح نوعیت دیتا

نوعیت دیتا (Data Type)	توضیح	اندازه به بایت	ساحه تعریف
Int	تنها اعداد تام را در خود ذخیره می‌کند.	2	32767 الی -32768
Char	تنها یک حرف را در خود ذخیره می‌کند.	1	127 الی -128
Float	اعداد اعشاری با دقت ساده که الی 7 رقم بعد از اعشاریه را نشان می‌دهد در خود ذخیره می‌کند.	4	3.4E+38 الی 3.4E-38
Double	اعداد اعشاری با دقت مضاعف که الی 15 رقم بعد از اعشاریه را نشان می‌دهد در خود ذخیره می‌کند.	8	1.7E+308 الی 1.7E-308

نوعیت دیتاهای تام (Integer Data Types)

این نوعیت دیتا (int) تنها شامل اعداد کامل و تام می‌باشد و اعداد اعشاری را در خود ذخیره نمی‌کند.

انواع مختلف integer قرار ذیل اند:

a. عدد تام بدون علامه (unsigned int)

متحول که به شکل (unsigned int) معرفی گردد در حقیقت تمام بیت های خویش را غرض ذخیره نمودن اعداد تام مثبت آماده می سازد و کدام بیت را برای تشخیص نمودن علامه (مثبت یا منفی) عدد ریزرف نمی کند.

b. عدد تام کوچک علامه دار (short int)

این نوعیت دیتا غرض ذخیره نمودن اعداد تام کوچک (short int) استفاده می شود.

c. عدد تام کوچک بدون علامه (unsigned short int)

وقتی که متحول از نوع (unsigned short int) معرفی گردد به این معنا است که می تواند اعداد تام کوچک مثبت را در خود ذخیره نماید.

d. اعداد تام بزرگ (long int)

این نوعیت دیتا برای اعداد تام بزرگ مورد استفاده قرار می گیرد.

e. اعداد تام بزرگ بدون علامه (unsigned long int)

این حالت بیت علامه را از بین برده و حافظه را دو چند می سازد و می تواند که اعداد تام بزرگ مثبت را در خود ذخیره نماید.

جدول ذیل انواع دیتا (data types)، اندازه ظرفیت (size in bytes) و ساحت تعریف (range) اعدادی تام (integer numbers) را نشان می دهد:

جدول ۶.۳ توضیح نوعیت دیتا

نوعیت دیتا (Data Type)	اندازه به بایت (Size in bytes)	ساحت تعریف (Range)
عدد تام کوچک بدون علامه unsigned short	2	0 الی 65535
عدد تام کوچک علامه دار (signed short)	2	- 32768 الی + 32767
عدد تام (unsigned int)	4	0 الی ۴۲۹۴۹۶۷۲۹۵
عدد تام (signed int)	4	- 2147483648 الی + ۲۱۴۷۴۸۳۶۴۷
عدد تام بزرگ (long)	8	0 الی ۹.۲۲۳۳۷۲۰۴ E +۱۸
حرفی بدون علامه (unsigned char)	1	0 الی 255

نوعیت دیتای اعشاری با دقت ساده (Float Point Data Types)

ما می‌توانیم نوعیت دیتا با دقت اعشاری ساده را با استفاده از دستور کلیدی float تعریف نماییم. float چهار بایت حافظه را ریزرف می‌کند و تا شش خانه بعد از اعشاریه در اجرای عملیات دقیق است.

نوعیت دیتای اعشاری با دقت مضاعف (Double Precision Data Types)

این نوعیت دیتا می‌تواند بزرگ‌ترین اعداد را با دقت بیش‌تر محاسبه نموده و نتیجه را محاسبه نماید. double هشت بایت از حافظه را ریزرف نموده و تا 14 رقم بعد از اعشاریه را با دقت عالی اجرا می‌نماید. برای تعریف این نوعیت دیتا از دستور کلیدی double استفاده صورت می‌گیرد.

نوعیت دیتای حرفی (Character Data Type)

این نوعیت دیتا می‌تواند یک حرف یا کرکتر ست ASCII را در خود ذخیره نماید. ظرفیت این نوع دیتا یک بایت بوده و با استفاده از دستور کلیدی char معرفی می‌شود. اگر بیت علامه را در نظر بگیریم این نوعیت دیتا می‌تواند عدد بین 0 الی 255 را در خود ذخیره نماید.

۱.۱۴ معرفی متحول‌ها (Declaring Variables)

معرفی متحول‌ها دربر گیرنده نوعیت دیتا و نام متحول می‌باشد که در ختم آن سیمی کولن (;) درج می‌گردد. این اعلامیه (declaration) در حقیقت مترجم یا کمپایلر را در باره نام و نوعیت متحول آگاه می‌سازد.

ساختار معرفی متحول طور ذیل است:

```
data – type a1, a2, a3 , ..... an
```

در این جا a1, a2, a3 و an نام‌های متحول‌ها است و در قسمت data-type یکی از نوعیت دیتاهای که قبلاً معرفی گردید (مانند int, float و غیره) درج می‌گردد. اگر نوعیت دیتای بیش‌تر از یک متحول معرفی گردد در آن صورت a1, a2, a3 با استفاده از کامه از هم‌دیگر جدا می‌شود.

مثال ۱۵.۳

```
int x, y, z ;  
float avg, sum ;  
char ans ;  
double amount ;
```

در این جا x, y و z از نوع int، avg و sum از نوع float، ans از نوع char و amount از نوع double معرفی شده اند. متحول های فوق را می توانیم به شکل ذیل نیز معرفی نماییم:

```
int x;
```

```
int y;
```

```
int z;
```

float avg;

```
float sum;
```

char ans;

نوعیت متحول‌های را می‌توانیم به شکل ذیل نیز معرفی نماییم:

```
short int count;
```

```
long int fact;
```

این که short و short int یک مفهوم را دارا است و long int و long نیز عین مفهوم را دارد می‌توانیم بیانیه‌های فوق را به‌طور ذیل معرفی نماییم:

short count ;

long fact;

۱.۱۴.۱.توظیف قیمت‌ها در متحول‌ها (Assigning Values to Variables)

شما می‌توانید به سه طریق قیمت‌ها را در متحول‌ها ذخیره کنید:

۱. در بخش اعلامیه (Declaration part)

۲. در بخش اجرایی (Executable part) برنامه با استفاده از علامه مساوی =

۳. با استفاده از صفحه کلید (Keyboard)

1.14.1.1 بیانیه = (Assignment Statement)

وقتی که می‌خواهیم قیمت را در یک متحول ذخیره نماییم از علامه = استفاده می‌کنیم.

شکل عمومی، آن قرار ذیل است:

Variable _name = Constants / Variable / Expression;

بیانیه ذیل را در نظر گیرید:

Total = 0;

این بیانیه نشان می‌دهد قیمت 0 که در سمت راست = قرار دارد در متحول سمت چپ = یعنی Total ذخیره می‌شود. در سمت راست = می‌تواند قیمتی ثابت، متحول و یا عبارت که حاوی هر دو (ثابت و متحول) باشد قرار گیرد. طور مثال:

Sum = marks 1 + marks 2;

در این فورمول بخش سمت راست = مورد ارزیابی قرار گرفته و بعداً مجموعه هر دو متحول marks1 و marks 2 در متحول سمت چپ = یعنی Sum ذخیره می‌شود.

Count = Count + 1

در بیانیه فوق، عدد 1 به قیمت متحول سمت راست = اضافه گردیده و بعد نتیجه در متحول سمت چپ = یعنی count ذخیره می‌شود. اگر قیمت اولی متحول cout طور مثال 10 باشد قیمت جدید آن 11 خواهد شد.

مثال ۱۶.۳: دادن قیمت به متحول‌ها در بخش اعلامیه (Decleration part)

// Assigning value to variables in the declaration part

```
.  
.   
.   
int x = 1 , y = 2, z = 3 ;  
float avg = 0.0 , pi ;  
char opt = ' y ' ;
```

Examples of assigning values to variables in the executable part are :

دادن قیمت به متحول‌ها در بخش اجرایی برنامه

```
pi = 3.14159 ;  
total = 0 ;  
opt = 'N' ;
```

لسان C++ به استفاده کننده (User) اجازه می‌دهد که عمل‌گرهای (Operators) مختلف را در یک بیانیه یا فورمول استفاده کند. عدد عبارات مجاز قرار ذیل است:

; A = B = C = 0

; X1 = X2 = Large

نوت: در سمت چپ علامه مساوی همیشه باید نام یک متحول وجود داشته باشد.

مثال ۱۷.۳: برنامه ذیل حاصل جمع دو عدد تام (int) را دریافت می نماید.

```
// program to find sum of two integer numbers
#include <iostream.h>

main ( ) {
    // Declaration and assignment
    int N1 =200, N2 = 300, Sum;
    Sum = N1 + N2;
    cout << "\n " << " Total = " << Sum;
    return 0;
}
```

Output

Total = 500

مثال ۱۸.۳: برنامه ذیل محیط دایره را دریافت می کند.

```
// program to find circumference of a cricle
#include <iostream.h>

main ( )
{
    // Declaration and assignment
    float pi = 3.14150 , radius = 10 , circum ;
    circum = 2 * pi * radius ;
    cout << "\n " << " Circumference = " << circum ;
    return 0 ; }
```

Output

Circumference = 62 . 831802

مثال ۱۹.۳: برنامه ذیل مجموعه و اوسط نمرات چهار مضمون را دریافت می کند.

```
// Program to find total and average of marks obtained in 4 subjects.
```

```
#include <iostream.h>
```

```
main ( ) {
```

```
    int m1, m2, m3, m4 ;
```

```
    float total , avg ;
```

```
    // assignment in executable part
```

```
    m1 = 50 , m2 = 70 , m3 = 75 , m4 = 67 ;
```

```
    total = m1 + m2 + m3 + m4 ;
```

```
    avg = total / 4 ;
```

```
    cout << "\n " << " The average marks = " << avg ;
```

```
    return 0 ;
```

```
}
```

Output

The average marks = 65 . 500000

مثال ۲۰.۳: برنامه ذیل مساحت دایره را محاسبه می کند.

```
// program to compute area of a circle
```

```
#include <iostream.h>
```

```
main ( )
```

```
{
```

```
    // Declarations
```

```
    float pi , r , area ;
```

```
    // Assignment
```

```
    pi = 3.14159 ;
```

```
    r = 5 ;
```

```
    // Calculation and printing
```

```
    area = pi * r * r ;
```

```
    cout << "\n " << " Area of circle = " << area ;
```

```
    return 0 ;
```

```
}
```

Output

Area of circle = 78 . 539749

شیوه دیگر که با استفاده از آن می توانیم قیمت ها را در متحول ها ذخیره نماییم استفاده از دستور ورودی cin می باشد. cin دستور ورودی C++ است که به استفاده کننده (user) اجازه می دهد تا قیمت ها را مستقیماً از keyboard وارد متحول سازد. برعکس، cout از جمله دستورهایی خروجی C++ است که نتایج را روی سکرین نشان می دهد.

مثال ۲۱.۳: برنامه ذیل حاصل جمع دو عدد تام (int) که قیمت های ن را با استفاده از دستور cin از صفحه کلید می گیرد ، دریافت می نماید.

```
// program to find sum of two integer numbers inserted via keyboard
#include <iostream.h>

main ( )
{
    // Declaration of variables
    int N1, N2, Sum;
    cout << "Enter 2 Integer numbers to Sum : ";
    cin >> N1;
    cin >> N2;
    Sum = N1 + N2;
    cout << "\n " << " The Sum is = " << Sum;
    return 0; }
```

Output

Enter 2 Integer numbers to Sum : 10 20
The Sum is = 30

مثال ۲۲.۳: برنامه ذیل ۳ عدد را صفحه کلید گرفته، حاصل جمع، تفریق، ضرب و تقسیم آن را دریافت نموده روی سکرین نشان می دهد.

```
/* program to find sum, Minus, Product and Division of three float numbers inserted
from keyboard */

# include <iostream.h>
```



```

main( )
{
    // Declaration
    float N1, N2, N3, Sum, Min, Prod, Div;
    cout << "Enter 3 Integer numbers to Sum, Minus, Product and Divide : ";
    cin >> N1 >> N2 >> N3;
    Sum = N1 + N2 + N3;
    Min  = N1 - N2 - N3;
    Prod = N1 * N2 * N3;
    Div  = N1 / N2 / N3;
    cout << "\n " << Sum;
    cout << "\n " << Min;
    cout << "\n " << Prod;
    cout << "\n " << Div;
    return 0; }

```

Output

```

Enter 3 Integer numbers to Sum, Minus, Product and Divide 20 10 5
35
5
1000
0.4

```

مثال ۲۳.۳: برنامه ذیل مفاد ساده را محاسبه می‌کند.

```

// program to calculate simple interest
#include <iostream.h>

main ( )
{
    int year ;
    float prin, rate, si ;
    cout << " Enter principle , rate and period : " ;
    cin >> prin >> rate >> year ;

```

```

si = prin * rate * year / 100 ;
cout << "\n " << " Simple interest = " << si ;
return 0 ;
}

```

Output

```

Enter principle, rate and perios : 1000  5  2
Simple interest  = 100. 0000000

```

مثال ۲۴.۳: برنامه ذیل شعاع یک دایره را در صورتی دریافت می کند که قیمت محیط دایره وجود دارد.

Program to find radius of a circle given the circumference //

```

#include <iostream.h #
main( )
{
    char x , y;
    cout << "\n " << " Enter two characters separately : \t;"
        cin >> x >> y;
    cout << "\n" << "You entered " << x << " and " << y << " characters ;"
    return 0;
}

```

Output

```

Enter two characters separately :  a    b
You entered a    and  b characters

```



در این فصل دستورهای کلیدی (keywords)، شناسه‌ها (identifiers)، ثابت‌ها (constants)، متحول‌ها (variables) و انواع دیتا (data types) مورد بحث قرار گرفت و در نتیجه مفاهیم ذیل را آموختیم:

دستورهای کلیدی عبارت از کلمات است که برای اهداف خاص در یک زبان استفاده می‌شود و نمی‌توانیم آن‌را به حیث نام‌های متحول‌ها استفاده نماییم. اما شناسه به نام اطلاق می‌شود که برای متحول‌ها، توابع و صف در برنامه استفاده می‌گردد.

ثابت عبارت از قیمت است که در اثنای اجرای برنامه قابل تغییر نباشد. ثابت‌ها در C++ به دو نوع می‌باشد که عبارت از ثابت‌های اولی و ثابت‌های دومی می‌باشد.

متحول نام است که به یک قسمتی از حافظه RAM داده می‌شود و همان قسمتی حافظه را ریزرف می‌کند. هر وقتی که قیمت به متحول داده می‌شود، قیمت در قسمت ریزرف شده حافظه به شکل موقت ذخیره می‌گردد. به یاد داشته باشید وقتی متحول معرفی می‌شود، باید نوعیت دیتا آن نیز تعیین گردد. شما می‌توانید که نوعیت متحول را عددی (تام و یا اعشاری) و یا حروفی (یک حرفی و یا چند حرفی) تعریف کنید.



سوالات و فعالیت های فصل سوم

۱. هدف بیانیه `#include <iostream.h>` چیست؟
۲. چرا متحول ها در برنامه تعریف می گردد؟
۳. شکل عمومی دستور `cin` و `cout` را بنوسید؟
۴. عمل گر `cin` را در یک مثال توضیح دهید؟
۵. عمل گر `cout` را در یک مثال ساده توضیح دهید؟
۶. تفاوت بین `cout` و `printf` چیست؟ توضیح دهید؟
۷. شناسه (Identifier) چیست؟
۸. دستورهای کلیدی (keywords) چیست؟ چند دستور کلیدی را نام بگیرید؟
۹. انواع دیتاهای ساده C++ را مختصرا توضیح دهید؟
۱۰. ثابت های عددی به چند نوع تقسیم شده است؟
۱۱. قواعد نام گذاری متحول ها چیست؟
۱۲. تفاوت بین متحول و ثابت چیست؟
۱۳. چطور می توانید یک متحول را تعریف (declare) نمایید، در یک مثال توضیح دهید؟

فعالیت ها

۱. ساختار عمومی یک برنامه C++ را در یک مثال توضیح دهید؟
۲. یک برنامه را بنوسید که در آن عمل گر ورودی و خروجی وجود داشته باشد؟
۳. تفاوت بین عمل گر `<<` و `>>` چیست؟
۴. تفاوت بین متحول ها و دستورهای کلیدی را با ذکر مثال های آن بیان دارید؟
۵. تفاوت میان ثابت تام `int` و ثابت اعشاری `float` چیست؟
۶. تفاوت بین ثابت حرفی `char` و ثابت حروفی `string` چیست؟

فصل چهارم

عمل‌گرها و عبارات ها Operators and Expressions



هدف کلی: با عمل‌گرها (Operators) و عبارات (Expressions) در برنامه‌نویسی آشنا شوند .

اهداف آموزشی: در پایان این فصل محصلان قادر خواهند شد تا:

۱. عمل‌گر (Operator) را تعریف نمایند.
۲. انواع مختلف عمل‌گرهای C++ را بیان نمایند.
۳. فواید عمل‌گرهای مختصر را توضیح دهند.
۴. عبارات ترکیبی را در مثال‌های مناسب نمایش دهند.
۵. انواع مختلف بیانیه‌ها (statements) را تشریح نمایند.

این فصل حاوی مفاهیم عمل گرها، انواع عمل گرها، عبارات و انواع عبارات می باشد یک هر یک از این مفاهیم به تفصیل مورد بحث قرار گرفته است.

زبان برنامه نویسی C++ چندین عمل گر را جهت اجرای عملیات ساده حسابی مانند: جمع، تفریق، ضرب، تقسیم، باقی مانده و دیگر عملیات منطقی دارد.

عمل گر (Operator) عبارت از سمبولی است که به خاطر اجرای عملیات خاصی حسابی و یا منطقی استفاده می شود. دیتای که روی آن عملیه اجرا می گردد به نام (Operands) یاد می شود.

عبارت (Expression) ترکیب عمل گرها و (Operands) های است که در نتیجه آن یک قیمت به دست می آید (Soulié, 2007).

مثال ذیل را در نظر گیرید:

$12 * 5$

* عمل گر ضرب است و 12، 5 عبارت از (operands) می باشد.

به صورت عموم، عمل گرها به چهار دسته تقسیم گردیده که عبارت اند از:

i. عمل گر یک تایی (Unary Operator)

ii. عمل گر دو تایی (Binary Operator)

iii. عمل گر سه تایی (Ternary Operator)

iv. عمل گر خاص (Special Operator)

۱.۱۵ عمل‌گرهای یک‌تایی (Unary Operators)

عمل‌گر یک‌تایی عبارت از عمل‌گری است که بالای یک عنصر- (operands) اجرا می‌گردد. این عمل‌گر همیشه قبل از operand ظاهر می‌شود. انواعی مختلف عمل‌گرهای unary که در C++ وجود دارد عبارت اند از (Stroustrup, 1997):

- i. یک‌تایی مثبت (+) – (Unary plus)
- ii. یک‌تایی منفی (-) – (Unary minus)
- iii. افزایش (++) – (Increment)
- iv. کاهش (--) – (Decrement)
- v. نفی منطقی (!) – (Logical NOT)
- vi. بیتی مکمل (~) – (Bitwise complement)
- vii. آدرس (&) – (Address)

علامه یک‌تایی منفی (-) غرض تفریق استفاده می‌شود. استفاده این علامه، علامه مقدار سمت راست خود را تغییر می‌دهد. طور مثال، این علامه مقدار موجود را ضرب 1- می‌سازد. هم‌چنان علامه یک‌تایی مثبت (+) مقدار موجود خود را ضرب 1+ می‌سازد.

مثال ۱.۴ قیمت a را یک عدد اضافه می‌کند.

```
/* Program to print a message */
#include <iostream.h>
main ( )
{
int a = 10;
    a++;
    cout <<"The Value of a is: "<< a << endl;
return 0;
}
```

Output

The Value of a is: 11

عملگرهای دوتایی (Binary Operators)

این عملگرها بالای دو عنصر (operands) اجرایی شود. عملگر دو تایی به پنج دسته تقسیم گردیده است (Oualline, 1995):

- i. عملگر حسابی (Arithmetic Operator)
- ii. عملگر ارتباطی (Relational Operator)
- iii. عملگر منطقی (Logical Operator)
- iv. عملگر مساوی (Assignment Operator)
- v. عملگر بیتی (Bitwise Operator)

۱.۱۶ عملگرهای حسابی (Arithmetic Operators)

زبان برنامه‌نویسی C++ پنج نوع عملگرهای حسابی دارد که در جدول ذیل لیست گردیده.

جدول ۱.۴ عملگرهای حسابی

عملگر	هدف
+	جمع
-	تفریق
*	ضرب
/	تقسیم
%	باقی‌مانده

در C++ هیچ عملگر برای محاسبه توان وجود ندارد اما تابع `pow()` است که غرض محاسبه توان در این لسان استفاده می‌شود.

تقسیم یک عدد تام بالای عدد تام دومی به نام **تقسیم تام** (Integer division) یاد می‌گردد که در آن بخش اعشاری صرف نظر می‌شود. عملگر % برای دریافت باقی‌مانده (Remainder) در لسان C++ مورد استفاده قرار می‌گیرد.

طور مثال، اگر $a = 10$ و $b = 3$ باشد نتیجه $a \% b$ یا $10 \% 3 = 1$ می‌شود. در حالی که $3 = 3 / 10$ می‌شود.

قابل یادآوری است که عملگر % تنها روی اعداد تام قابل اجرا است.

مثال ۲.۴: متحول‌ها a و b را از نوع اعداد تام یا int در نظر بگیرید

$a = 16, b = 3.$

عبارت‌ها	نتیجه
$16 +$	۱۶
$5 -$	۵-
$a + b$	۱۹
$a - b$	۱۳
$a * b$	۴۸
a / b	۵ (بخش اعشاری حذف گردیده است)
$a \% b$	۱ (باقی‌مانده عملیه تقسیم)

متحول C1 و C2 از نوع حروفی است و قیمت حروف P و T را در خود ذخیره نموده در نظر بگیرید.

$C1 = 'A'$ and $C2 = 'a'$

$C1 = 65$, and $C2 = 97.$

عبارت‌ها	نتیجه
$C1 + C2$	۱۶۲
$C1 + C2 + 5$	۱۶۷
$'C1' + '1'$	۱۱۴

به یاد داشته باشید که قیمت '1' مساوی به ۴۹ است.

در تقسیم اعداد تام، اگر یکی از اعداد منفی باشد در آن صورت نتیجه متعلق به کمپیوتر می‌شود.

طور مثال:

$$5 / 6 = 0$$

$$-5 / -6 = 0$$

اما نتیجه $6/5$ ممکن ۰ و یا -۱ شود. در Turbo C++ نتیجه ۰ می‌شود. اما در عملیه باقی‌مانده، علامه عدد باقی‌مانده متعلق به علامه عدد اول می‌باشد.

طور مثال:

$$-20 \% 6 = -2$$

$$-20 \% -6 = -2$$

$$20 \% -6 = 2$$

اگر در یک عملیه هر دو عدد اعشاری باشد، چنین عملیه را به نام Real Arithmetic () یاد می‌کند. در این نوع عملیه اعداد می‌توانند به شکل اعشاری و یا توان دار باشند. نتیجهٔ محصول نیز به شکل اعشاریه‌دار ظاهر می‌شود.

۱.۱۶.۱ عملیات مختلط حسابی (Mixed – Mode Arathmatic Operations)

در عملیات مختلط حسابی یک عدد از نوع تام (int) و عدد دومی از نوع اعشاری (real) می‌باشد. وقتی که یکی از این دو عدد اعشاری باشد نتیجه نیز شکل اعشاری را می‌گیرد (Prinz, 2002).

طور مثال:

$$15/2 = 7 \text{ و } 15/2.5 = 7.5 \text{ می‌شود.}$$

۱.۱۶.۲ اولویت بندی عمل‌گرهای حسابی (Precedence of Arathematic Operators)

هر عبارت مطابق قواعد اولویت مورد ارزیابی قرار می‌گیرد.

طور مثال:

$$4 * 5 / 2$$

این عبارت از سمت چپ به راست ارزیابی می‌شود. در قدم اول $4*5$ گردیده و بعد حاصل ضرب (20) تقسیم 2 می‌شود. ترتیب اجرای عبارت بیش‌تر متکی به اولویت عمل‌گرها است که در جدول ذیل بیان گردیده است.

جدول ۲.۴ توضیح حق اولویت بندی عمل‌گرهای مختلف

جهت فعالیت	عمل‌گر	حق اولویت بندی
چپ به راست	()	۱
راست به چپ	Unary	۲
چپ به راست	%, /, *	۳
چپ به راست	-, +	۴

- در یک عملیهٔ حسابی عبارت که در بین قوس قرار دارد اول اجرا می‌گردد.
- در صورتی که در یک عملیه چندین جوهر قوس‌ها وجود داشته باشد، عبارتی که در وسط قوس وجود دارد اول اجرا می‌شود و به همین قسم به طرف بیرون یکی پی دیگری اجرا می‌شود.

- اگر در یک عملیه دو یا بیش تر عمل گرها که دارای حق اولویت مساوی باشد وجود داشته باشد در آن صورت عملیه از سمت چپ به راست اجرا می شود. به این معنا، آن عمل گر که نزدیک مساوی قرار دارد اول اجرا می شود.
- قوس ها می تواند حق اولویت را تغییر دهند.

مثال ۳.۴: معادله ذیل را در نظر بگیرید.

$$X1 = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

اگر $a = 1$ ، $b = 4$ و $c = 4$

معادل معادله فوق در زبان C++ قرار ذیل است:

$$\begin{aligned} x &= (-b + \text{sqrt}(b * b - 4 * a * c)) / (2 * a) \\ &= ((-4) + \text{sqrt}(4 * 4 - 4 * 1 * 4)) / (2 * 1) \\ &= (-4 + \text{sqrt}(4 * 4 - 4 * 1 * 4)) / (2 * 1) // \text{Evaluation inner most prantheses} \\ &= (-4 + \text{sqrt}(16 - 4 * 4)) / (2 * 1) \\ &= (-4 + \text{sqrt}(16 - 16)) / (2 + 1) \\ &= (-4 + \text{sqrt}(0)) / (2 * 1) \\ &= (-4 + 0) / (2 * 1) \\ &= -4 / 2 \\ x1 &= -2 \end{aligned}$$

مثال ۴.۴: عبارت ذیل را در نظر بگیرید.

$$(-5 * 2 / (6 + 4 - 2) + (2 / 3 + 5))$$

ترتیب ارزیابی آن قرار ذیل می باشد:

$$\begin{aligned} &= (-5 * 2 / (6 + 4 - 2) + (2 / 3 + 5)) \\ &= (-5 * 2 / (10 - 2) + (2 / 3 + 5)) \\ &= (-5 * 2 / 8 + (2 / 3 + 5)) \\ &= (-5 * 2 / 8 + (0 + 5)) \\ &= (-5 * 2 / 8 + 5) \\ &= (-10 / 8 + 5) \\ &= -1 + 5 \end{aligned}$$

= 4

برنامه که عبارت ذیل را مورد ارزیابی قرار می دهد:

```
// Program to Evaluate a given Expression
#include <iostream.h>
main (
{
    int result ;
    cout << "\n " << "The given expression is : (-5 *2/(6+4-2) + ( 2/3 + 5)) " ;
    result = (-5 * 2 / ( 6 + 4 -2 ) + ( 2/3+5));
    cout << "\n" << "Result of evaluation is : " << result ;
    return 0;
}
```

Output

The given expression is : (-5 *2/(6+4-2) + (2/3 + 5))

Result of Evaluation : 4

اگر بیش تر از یک جوهره قوس ها در کنار یک دیگر ظاهر گردد در آن صورت ارزیابی آن از سمت چپ به راست در نظر گرفته می شود. طور مثال، چپ ترین ست قوس ها اول اجرا گردیده و جوهره قوس های که در سمت راست موقعیت دارد آخر اجرا می شود. از قوس ها وقتی استفاده می شود که در قسمت اولویت عمل گر ها شک و شبه وجود داشته باشد. هم چنان استفاده از قوس ها برنامه را خوانا تر می سازد.

مثال ۵.۴: عبارت ذیل را در نظر گیرید.

++x - y ++	where x = 6 , y = 2
++x - 2	unary operators (from right to left)
7 - 2	unary operators
	First x = 6 , x = 7 (pre-increment)
5	Substraction

۱.۱۷ عمل گر های ارتباطی (Relational Operators)

این عمل گر ها بیش تر در عملیات شرطی که نتیجه آن درست (True) و یا نادرست (False) می باشد استفاده می گردد. این عمل گر ها نتیجه هر عبارت را به شکل تام نشان می دهد. اگر نتیجه عبارت

نا درست باشد آن را به 0 نمایش می‌دهد ولی اگر نتیجه درست باشد آن را به قیمت تام که خلاف 0 باشد، نشان می‌دهد.

طور مثال:

$$50 < 200$$

$$X > Y$$

عبارت‌های فوق را به نام عبارت‌های ارتباطی (relation expressions) یاد می‌کنند که نتیجه آن ممکن درست و یا نادرست باشد. اگر قیمت $X = 10$ و $Y = 5$ باشد در آن صورت $X > Y = \text{True}$ می‌شود.

به صورت عموم، شش نوع عمل‌گرهای ارتباطی وجود دارد.

معنی	عمل‌گر
بزرگ‌تر از	$<$
کوچک‌تر از	$>$
بزرگ‌تر و یا مساوی	$= <$
کوچک‌تر و یا مساوی	$= >$
مساوی	$= =$
نا مساوی	$= !$

دو عمل‌گر اخیر را به نام عمل‌گرهای مساوی (equality operators) نیز یاد می‌کند.

مثال ۶.۴		
$i = 1, j = 2, k = 7$		3 متحول تام مقابل را در نظر بگیرید:
قیمت تام	قیمت منطقی	عبارت‌ها
۱	درست	i. $10.5 < = 12$
۰	نادرست	ii. $i = = 20$
۰	نادرست	iii. $5 > 20$
۰	نادرست	iv. $(I + j) > k$
۱	درست	v. $(j + k + 1) > (i + 5)$
۰	نادرست	vi. $k ! = 7$

در مثال دوم و سوم فوق، عبارت‌های حاوی عبارت‌های حسابی می‌باشد. به همین خاطر در عبارات ارتباطی $k > (j +)$ اول عبارت داخل قوس اجرا می‌گردد و بعد نتیجه با k مقایسه می‌شود. این

عملگرهای حسابی نسبت به عملگرهای ارتباطی دارای اولویت بیش‌تر می‌باشد. عملگرهای ارتباطی برای مقایسه دو مقدار در بیانیه‌های تصمیم‌گیری استفاده می‌گردد.

مثال ۷.۴: برنامه ذیل طریقه استفاده عملگرهای ارتباطی (Relational operators) را نشان می‌دهد.

```
// Program to show usage of relational operators
#include <iostream.h>
main ( )
{
    int i = 1, j = 2, k = 7 ;
    cout << "\n" << " Value of " << 10 << " <= " << 12 ;
    cout << "\n" << " Value of i" << " == " << 2 ;
    cout << "\n" << " Value of " << 4 << " < " << 5;
    cout << "\n" << " Value of " << i + j << " != " << k ;
    cout << "\n" << "Value of " << j + k + 1 << " > " << i + 5 ;
    cout << "\n" << "Value of " << k << " == " << 7 ;
    return 0 ;
}
```

Output

```
Value of 10 <= 12
Value of i == 2
Value of 4 < 5
Value of 3 != 7
Value of 10 > 6
Value of 10 == 7
```

۱.۱۲.۱ عملگرهای منطقی (Logical Operators)

این عملگرها برای یک‌جا نمودن دو یا بیش‌تر از دو عبارت منطقی استفاده می‌گردد. به‌صورت عموم، سه عملگر منطقی در ++C وجود دارد که عبارت از AND منطقی، OR منطقی و Not منطقی می‌باشد.

جدول ۳.۴ عمل گر های منطقی

عمل گر	معنی
!	Not منطقی
&&	AND منطقی
	OR منطقی

⇐ نتیجه (AND) منطقی وقتی درست می برآید که هر دو قیمت درست باشد اما اگر یکی از این دو قیمت ها نادرست باشد نتیجه (AND) منطقی نیز نادرست می شود.

⇐ نتیجه (OR) منطقی در صورتی نادرست می برآید که هر دو قیمت نادرست باشد اما اگر یکی از این دو قیمت ها درست باشد نتیجه (OR) منطقی نیز درست می شود.

⇐ ++C یک عمل گر دیگری به نام عمل گر یکانی (!) که معنای NOT را ارایه می کند دارد که نتیجه یک عبارت منطقی را همیشه برعکس آن نشان می دهد.

جدول ۴.۴ جدول حقیقت (truth table) برای عمل گرهای منطقی

<p>مثال 8.4: اگر $i = 8$ ، $x = 5.1$ ، $c = 'a'$ باشد در صورت که i دارای دیتا تایپ تام یا <code>int</code>، متحول x اعشاری و c دارای دیتا تایپ حرفی یا <code>character</code> باشد.</p> <p>استفاده عمل گرهای منطقی را ذیلا مشاهده نمایید.</p>		
عبارت (Expression)	قیمت منطقی (Logical value)	قیمت تام (Integer value)
$(i > 6) \&\& (c == 'a')$	True	۱ (خلاف صفر)
$i < 6 \ \ (c == 'b')$	False	۰
$i >= 6 \ \&\& \ (c == 97)$	True	۱
$(x < 10) \ \ (I > 10)$	True	۱
$!(i <= 4)$	True	۱
$(c != 'p')$		۱

جدول ۵.۴ توضیح جدول حقیقت (truth table) برای عمل‌گرهای منطقی

نتیجه شرط اول a !	نتیجه b a	نتیجه b && a	شرط دوم B	شرط اول A
نادرست (False)	درست (True)	درست (True)	درست (True)	درست (True)
نادرست (False)	درست (True)	نادرست (False)	نادرست (False)	درست (True)
درست (True)	درست (True)	نادرست (False)	درست (True)	نادرست (False)
درست (True)	نادرست (False)	نادرست (False)	نادرست (False)	نادرست (False)
		نادرست (False)		

۱.۱۷.۲ عمل‌گر (=) (Assignment Operators)

این عمل‌گر (=) وقتی استفاده می‌شود که به یک متحول، ثابت و یا عبارت قیمت داده شود.

ساختار این عمل‌گر قرار ذیل است:

```
variable = variable / constant / expression;
```

طور مثال:

$X = 10$; در این عبارت قیمت X عبارت از 10 می‌باشد.

$B = X$; در این عبارت قیمت X که عبارت از 10 است توظیف می‌شود.

$Sum = X + B$; در این عبارت مجموعه X و B که عبارت از 20 می‌شود در متحول Sum ذخیره می‌شود.

مثال ۹.۴: برنامه ذیل مجموعه ۴ عدد را دریافت نموده روی سکرین نشان می‌دهد.

```
// Program to print sum of 4 numbers
#include <iostream.h >

main ( )
{
    int a, b, c, d, sum ;
    cout << "\n " << " Enter 4 numbers : " ;
    cin >> a >> b >> c >> d;
    sum = a + b + c + d ;    // assignment statement
```


<pre>cout << "\n" << sum ; return 0 ; }</pre>
Output Enter 4 numbers : 10 15 22 40 Sum = 87

Variable operator = expression;

طور مثال:

$N = N + 5 ;$

بیانیه فوق را می توان به شکل مختصر چنین نوشت:

$N += 5;$

در این جا $+=$ شکل اختصار یافته بیانیه فوق ($N = N + 5$) است که به نام **عمل گر ترکیبی** (compound operator) یاد می شود. $+=$ به معنای این است که 5 را با قیمت N (سمت راست =) اضافه می کند و در نهایت نتیجه در N (سمت چپ =) ذخیره می شود. تعدادی از عمل گرهای اختصار یافته = قرار ذیل است.

جدول ۶.۴ توضیح عمل گرهای اختصار یافته

معنا	عمل گر	= به شکل ترکیبی	= به شکل عادی
$= +$ و	$= +$	$n += 1$	$n = n + 1$
$= -$ و	$= -$	$n -= 50$	$n = n - 50$
$= *$ و	$= *$	$n *= 10$	$n = n * 10$
$= /$ و	$= /$	$n /= 6$	$n = n / 6$
$= \%$ و	$= \%$	$n \% = 5$	$n = n \% 5$

فواید عمل گرهای مختصر (Advantages of shortened Assignment Operators)

۱. متحول سمت چپ لازم نیست که بار دوم در سمت راست = نوشته شود.
۲. عبارت کوتاه و مختصر می گردد و خواندن آن آسان تر می شود.
۳. دارای موثریت (efficient) عالی می باشد.

تفاوت بین عمل گر == و =

۱. == از جمله عمل گرهای مقایسوی است و به نام علامه مساوی (equalityoperator) یاد می شود. این عمل گر برای مقایسه نمودن دو قیمت استفاده می شود. طور مثال:

If (x == 10)

در حالی که = عمل گر توظیف (assignment operator) است که یک قیمت را در متحول توظیف می نماید. عمل گر توظیف (=) قیمت که در سمت راست آن قرار دارد در متحول سمت چپ = ذخیره می کند. طور مثال:

Y = 20 ;

۲. عمل گر (==) قیمت متحول را که در سمت چپ قرار دارد تغییر نمی دهد. در حالی که = قیمت سمت چپ علامه = را تغییر می دهد.

چندین عمل گر = (Multiple Assignment)

C++ به ما اجازه می دهد که چندین = را در یک بیانیه استفاده نماییم مانند:

Variable = Variable1 = Variable2 = = Expression;

در این جا علامه = پی یک دیگر از سمت راست به چپ اجرا می گردد. طور مثال $X = Y = Z = 10$ به معنای $(X) = (Y) = (Z) = 10$ می باشد. در این جا 10 در قدم نخست در Z ذخیره می شود؛ در قدم دوم قیمت Z که عبارت از 10 می باشد در Y ذخیره شده و به همین ترتیب قیمت Y در X ذخیره می شود. در اخیر قیمت X، Y، Z مساوی به 10 خواهد بود.

مثال ۱۰.۴: برنامه ذیل عدد بزرگ تر بین ۲ عدد وارد شده را دریافت می نماید

```
// Program to find Largest of 2 numbers
#include <iostream.h>

main ( )
{
    int n, m, big ;
    cout << "\n" << "Enter two integer numbers : " ;
    cin >> n >> m ;
```

```
big = ( n > m ) ? n : m ;
cout << "\n" << "The Largest of " << n << " and " << m << " is : " << big ;
}
```

Output

Enter two integer number : 10 56
The Largest of 10 and 56 is : 56

مثال ۱۱.۴: برنامه ذیل عدد ورودی را چک نموده که تاق است و یا جفت .

```
// Program to find whether a number is even or odd
#include <iostream.h>
main( )
{
    int x ;

    cout << "\n" << "Enter a number; " :
    cin >> x;
    ( x % 2 == 0 ) ? cout << x << " is even " : cout << x << " is odd ;"
}
```

Output

Enter a number : 17
17 is odd
Enter a number : 20
20 is even

مثال ۱۲.۴: برنامه ذیل بلندترین نمره بین نمرات ۴ مضمون وارد شده را دریافت می نماید.

```
//Program to find the highest marks of a student in 4 Exams
#include <iostream.h>
main( )
{
    int m1, m2, m3, m4, highest;
```

```

cout << "\n " << "Enter the marks in 4 papers : " << "\n ";
cin >> m1 >> m2 >> m3 >> m4;
highest = m1 > m2 ? m1 : m2;
highest = highest > m3 ? highest : m3;
highest = highest > m4 ? highest : m4 ;
cout << "\n " << "Highest marks in 4 papers = " << highest;
}

```

Output

Enter the marks in 4 papers :

56 74 66 60

Highest marks in 4 papers = 74

عمل گر بیتی (Bitwise Operators)

این عمل گر ها به خاطر کار کردن روی بیت ها استفاده می شوند. در C++ این عمل گر ها قرار ذیل اند:

- i. عمل گر بیتی AND(&)
- ii. عمل گر بیتی OR(|)
- iii. عمل گر بیتی XOR(^)
- iv. عمل گر بیتی NOT(~)
- v. عمل گر بیتی تغییر مکان به سمت چپ (<<) Shift left
- vi. عمل گر بیتی تغییر مکان به سمت راست (>>) Shift right

عمل گر بیتی AND (&)

عمل کرد عمل گر بیتی AND شبیه عمل گر منطقی AND است. با این تفاوت که عمل گر بیتی کارش روی بیت انجام می دهد. اگر قیمت هردو طرف (1) باشد پس نتیجه (1) در غیر آن قیمت (0) می شود.

عمل گر بیتی OR (|)

این عمل گر نیز شبیه عمل گر منطقی (OR) است. اگر تمام قیمت ها یک سان باشد قیمت (0) در آن صورت قیمت (0) در غیر آن قیمت (1) می شود.

عمل گر بیتی (NOT (~))

این عمل گر مقدار بیت هارا معکوس می کند. (قیمت 1 را 0 و قیمت 0 را به یک تبدیل می کند).

عمل گر بیتی (XOR (^))

در صورتی که مقادیرهای هردو طرف این عمل گر هردو (0) ویا (1) باشد قیمت (0) در غیر آن قیمت (1) می شود.

عمل گر بیتی تغییر مکان به سمت چپ (<<) Shift Left

این عمل گر بیت های Operand سمت چپ را به تعداد n مکان مشخص شده توسط Operand سمت راست، به سمت چپ منتقل می کند.

$$00001010 = 10$$

$$00101000 = 2 \gg 10$$

عمل گر بیتی تغییر مکان به سمت راست (>>) Shift Right

این عمل گر شبیه به عمل گر تغییر مکان به سمت چپ است با این تفاوت که بیت ها را به سمت راست انتقال می دهد.

$$0110100 = 100$$

$$00000110 = 3 \ll 100$$

جدول درستی عمل گر های AND, OR, NOT, XOR

X	Y	X~	Y~	AND	OR	XOR
۱	۱	۰	۰	۱	۱	۰
۱	۰	۰	۱	۰	۱	۱
۰	۱	۱	۰	۰	۱	۱
۰	۰	۱	۱	۰	۰	۰

مثال ۱۳.۴: استفاده عمل گرهای بیتی در C++.

```
#include <iostream.h>

main ( )
{
    int And, Or, Not_a, Not_b, Xor;
    int a = 8 , b=5;
    And = a & b;
    Or = a | b;
    Not_a = ~a;
    Not_b = ~b;
    Xor = a ^ b;
    cout <<"The Value of a&b is: "<< And << endl;
    cout <<"The Value of a|b is: "<< Or << endl;
    cout <<"The Value of a NOT is: "<< Not_a << endl;
    cout <<"The Value of b NOT is: "<< Not_b << endl;
    cout <<"The Value of a^b is: "<< Xor << endl; return 0;
}
```

Output

The Value of a&b is: 0
The Value of a|b is: 13
The Value of a NOT is: -9
The Value of b NOT is: -6
The Value of a^b is: 13

عمل گرهای سه تایی (Ternary Operators)

این عمل گرها بالای سه عنصر دیتا اجرا می شود. عمل گر سه تایی (? :) به نام عمل گر شرطی نیز یاد می گردد. (این عمل گرها در فصل پنجم به بحث گرفته شده است).

عملگرهای خاص (Special Operators)

این نوع عملگرها شامل کامه (،) و sizeof می باشد.

Chapter 19

Chapter 20 عملگر (sizeof)

i. این عملگر اندازه قیمت وارد شده در سمت راست = را به شکل بایت نشان می دهد.

ii. ساختار این عملگر قرار ذیل است:

sizeof (n)

در این جا n می تواند متحول، ثابت و یا دیتا تایپ باشد.

iii. این یک عملگر یک تایی است که از راست به چپ در نظر گرفته می شود.

iv. این عملگر جهت تعیین و یا تشخیص نمودن اندازه صف (array) و یا structure استفاده می گردد. این عملگر را می توانیم جهت جا به جا نمودن متحول در حافظه متحرک (dynamic memory) نیز استفاده کرد.

مثال ۱۴.۴: برنامه ذیل شیوه استفاده عملگر sizeof را نشان می دهد.

```
// Program to show usage of sizeof operator
#define STRING "Usage of sizeof Operator"
#include <iostream.h>

main ( )
{
    char a = 'p' ;
    cout << "\n" << " The size of char is          : " << sizeof (char) ;
    cout << "\n" << " Therefore the size of char a is  : " << sizeof (a) ;
    cout << "\n" << " However the size of ' p ' is    : " << sizeof ( 'p');
    cout << "\n\n" << "STRING  Usage of sizeof Operator\n";
    cout << " \n" << " The number of bytes in STRING IS : " << sizeof (STRING)
;
    cout << "\n" << "The size of short is          : " << sizeof (short);
    cout << "\n " << "The size of int is          : " << sizeof (int) ;
```

```
cout << "\n " << "The size of long is      : " << sizeof (long) ;
cout << "\n " << "The size of float is      : " << sizeof (float) ;
cout << "\n " << "The size of double is      : " << sizeof (double) ;
}
```

Output

```
The size of char is      : 1
Therefore the size of char a is      : 1
However the size of ' p ' is      : 1
STRING    " Usage of size Operator "
The number of bytes in STRING IS : 25
The size of short is      : 2
The size of int is      : 4
The size of long is      : 4
The size of float is      : 4
The size of double is      : 8
```

۱.۱۸ عبارت‌ها (Expressions)

عبارت‌های مجاز شامل عمل‌گرها (operators)، ثابت‌ها (constants) و متحول‌ها (variables) می‌باشد. این‌ها ممکن حاوی تابع صداکننده (function calls) نیز باشد که قیمت‌ها را بازگشت (return values) می‌دهد (Rama, 2011).

به صورت عموم چهار نوع عبارت در لسان C++ وجود دارد که عبارت‌اند از:

- عبارت ثابت (Constant Expression)
- عبارت حسابی (Arithmetic Expression)
- عبارت ارتباطی (Relational Expression)
- عبارت منطقی (Logical Expression)

عبارت که از ترکیب عبارت‌های فوق به میان می‌آید به نام عبارت ترکیبی (Compound expression) یاد می‌شود.

عبارت ثابت (Constant Expression): عبارت ثابت آن است که در تشکیل خویش تنها قیمت‌های ثابت را داشته باشد.

عبارت حسابی (Arithmetic Expression): آن است که در ترکیب خویش متحول‌ها، ثابت‌ها و عمل‌گرهای حسابی را داشته باشد.

عبارت ارتباطی (Relational Expression): آن است که در ترکیب خود متحول‌ها، ثابت‌ها، عمل‌گرهای حسابی و مقایسوی را داشته باشد. نتیجه این عبارت‌ها به شکل true یا false ظاهر می‌گردد.

عبارت منطقی (Logical Expression): آن است که در آن دو یا بیش‌تر از دو عمل‌گر ارتباطی به کمک عمل‌گرهای منطقی وصل شده باشد. نتیجه این عبارت به شکل (true) و (false) ظاهر می‌شود.

۱.۱۹ بیانیه‌ها (Statements)

بیانیه بخش مهمی یک برنامه است که قابلیت اجرا شدن (execution) را داشته باشد (Overland, 2013).

یک بیانیه مشخصات ذیل را داراست:

- بیانیه یک بخش کوچکی از برنامه است که قابلیت اجرا شدن را دارد.
- تمام بیانیه‌ها باید به سیمی کولن (;) ختم گردد.
- مجموعه دستورهایی که در یک بلاک قرار دارد و یک‌جا باید اجرا گردد نیاز است که در بین یک جفت قوس‌بزرگ { } نوشته شود.

بیانیه‌های انتخابی شامل () if و switch می‌باشد.

بیانیه‌های تکراری شامل حلقه while ، for و do-while می‌باشد. این‌ها را به نام بیانیه‌های حلقوی نیز یاد می‌کند.

بیانیه‌های پرش (Jump) شامل break، continue، goto و return می‌باشد.

بیانیه‌های label شامل case، default و goto می‌باشد.

بیانیه عبارت (Expression statement) از عبارت‌های مجاز ترکیب شده‌اند.

بیانیه‌های بلاک که به نام بیانیه‌های ترکیبی نیز یاد می‌شود از مجموعه بیانیه‌های است که در بین جوره قوس‌های بزرگ (Braces) نوشته می‌شود.

۱.۱۹.۱ ارزیابی عبارت ها (Evaluation of Expressions)

- اگر خواسته باشیم یک بیانیه را ارزیابی نماییم لازم است تا افاده الجبری را به افاده C++ تبدیل نماییم.
- در قدم دوم باید آنرا به فارمت = یعنی (assignment statement) نوشته نماییم.
- تمام متحول های سمت راست قبل از این که ارزیابی گردد باید قیمت داده شود.
- در هر جای که نیاز به عمل گر است باید ذکر گردد.
- در زبان C++ عمل گر خاص برای توان وجود ندارد و به همین خاطر از تابع pow استفاده صورت می شود.

جدول ۸.۴ نشان دهنده چند مثال حسابی و تبدیل آن به عبارات C++

عبارت های حسابی	عبارت های C++
1. $ab + cd$	1. $a * b + c * d$
2. $(a + b)(a - b)$	2. $(a + b) * (a - b)$
3. $\frac{a}{b} + d$	3. $(a / b) + d$
4. $\frac{2x^2 + 3x - 1}{10}$	4. $(2 * x * x + 3 * x - 1) / 10$
5. $\text{root} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$	5. $\text{root} = (-b + \text{sqrt}(b * b - 4 * a * c)) / (2 * a)$
6. $a^2 - \frac{b}{2} + c^2$	6. $a * a - b / 2 + c * c$

حق اولویت عمل گر ها (Precedence of Operators)

حق اولویت عمل گر ها وقتی مطرح می شود که در یک بیانیه چندین عمل گر وجود داشته باشد.

- عمل گری که حق اولویت بیش تر دارد اول اجرا می گردد (Oualline, 1995).
 - اگر عمل گر ها دارای حق اولویت مساوی باشند در آن صورت، اول عمل گری اجرایی گردد که در سمت چپ بیانیه موقعیت دارد.
- طور مثال: $(a < 10 \ \&\& \ a + b > 15)$

اگر قیمت $a = 20$ و $b = 1$ باشد.

در آن صورت اولویت بندی قرار ذیل می باشد:

i. طوری که + از بالاترین اولویت در این عبارت برخوردار است به همین خاطر عملیه جمع اول آن جام می شود.

$$a < 10 \ \&\& \ 21 > 15$$

ii. هر دو عمل گر ارتباطی (> و <) در قدم دوم حق اولویت بیش تر نظر به عمل گر منطقی (&&) دارد. پس، عمل گر ارتباطی که در سمت چپ وجود دارد دوم و به تعقیب آن عمل گر سمت راست سوم اجرا می شود.

$20 < 10 \&\& 21 > 15$

false && true

iii. نتیجه عمل گر منطقی (&&) نادرست یا (false) یا 0 می شود.

جدول ۹.۴ توضیح حق اولویت بندی تمام عمل گرهای C++

عمل گر	کتگوری عمل گر	جهت فعالیت	حق اولویت بندی
() [] . , →	صدا کردن تابع، قوسهای کوچک عناصر صنف انتخاب عناصر	از چپ به راست	۱
+ - ++ -- ! ~ * Sizeof (type)	جمع یک تایی منفی یک تایی افزودن کاهش Not منطقی One's complement Pointer reference آدرس یی اندازه متحول عمل گر cast	از راست به چپ	۲
*, /, %	ضرب، تقسیم و باقیمانده	از چپ به راست	۳
-, +	جمع و تفریق	از چپ به راست	۴
<<, >>	شفت راست و شفت چپ	از چپ به راست	۵
=<, >=, <, >	عمل گر های ارتباطی	از چپ به راست	۶
!=, ==	عمل گر مساوی	از چپ به راست	۷
&	AND بیت وایز	از چپ به راست	۸
^	XOR بیت وایز	از چپ به راست	۹

۱۰	از چپ به راست	OR بیت وایز	
۱۱	از چپ به راست	AND منطقی	&&
۱۲	از چپ به راست	OR منطقی	
۱۳	از راست به چپ	عمل گر شرطی	: ?
۱۴	از راست به چپ	مساوی عمل گر های AND به شکل سمبول ها مساوی	= =* =/ =% =+ =- =& ^= =! ==>> ==<<
۱۵	از چپ به راست	عمل گر کامه	,

مثال 15.4: در مثال ذیل چرا اوسط نمرات به شکل درست محاسبه نگردیده است؟

```
#include <iostream.h>

main ( )
{
    int Computer = 90;
    int English = 60;

    float Average = Computer + English / 2 ; // Mistake of precendence order
    cout << " \n The Average = "<< Average;
    return 0 ;
}
```

Output

The Average = 120

در مثال فوق دیده می شود که اوسط نمرات به شکل درست محاسبه نگردیده. به خاطری که در فورمول فوق علامه + و / وجود دارد و تقسیم اول اجرا گردید. نیاز است که قسمت صورت فورمول باید بین قوس های کوچک درج گردد تا در قدم اول قسمت صورت اجرا و بعد تقسیم مخرج گردد.

مثال ۱۶.۴: در مثال ذیل اوسط نمرات به شکل درست محاسبه گردیده است؟

```
#include <iostream.h>

main ( )
{
    int Computer = 90;
    int English = 60;

    float Average =( Computer + English ) / 2 ;
    cout << " \n The Average = "<< Average;
    return 0 ;
}
```

Output

The Average = 75

بعضی مسایل محاسباتی (Some Computational Problems)

در اثنای محاسبه عبارت ها مختلط (mixed mode expressions) بعضی مسایل رخ می دهد که سبب بروز اشتباه در محاسبه می شود.

دو مشکل عمده محاسباتی که معمولاً در اثنای محاسبه رخ می دهد قرار ذیل اند:

❖ اگر یک عدد بالای 0 تقسیم گردد یک مشکلی دیگر محاسباتی بروز می نماید. تقسیم یک عدد بالای 0 سبب می گردد که برنامه به شکل غیر عادی (abnormal) اختتام یابد. به همین خاطر لازم است که هیچ عدد تقسیم 0 نشود.

❖ سرریز (overflow) و پاریز (underflow) مشکل دوم است که در محاسبات رخ می دهد. به این خاطر لازم است که متحول ها به دیتاتایپ درست تعریف شود و قیمتی که به همین متحول ها وارد می شود نیز در رنج درست قرار داشته باشد.

مثال ۱۷.۴: برنامه ذیل مجموعه سلسله ذیل را دریافت می نماید.

```
/* Program to find the summation of the series
   S = 1 + 1 / 2 + 1 + ..... + 1 / n using cast operator */
#include <iostream.h>

main ( )
{
    float s = 0 ;
    int n, i ;
    cout << "\n " << " Enter the value of n : " ;
    cin >> n ;
    for ( i = 1 ; i <= n ; i ++ )
        s = s + 1 / ( float ) i ;
    cout << "\n " << " Sum of series : " << s ;
}
```

Output

Enter the value of n : 3
Sum of series : 1 . 833

توابع کتابخانه (Library Functions)

زبان ++C دارای توابع مختلف کتابخانه‌یی است که غرض اجرای عملیات مختلف مورد استفاده قرار می‌گیرد. این توابع به شکل گروپی مانند object در کتابخانه‌های جداگانه قرار دارد (Halterman, 2018).

- یک تابع کتابخانه را وقتی می‌توانیم از تابع دیگر به دسترس قرار دهیم که نام تابع و لست argument های که معلومات را به تابع بازگشت می‌دهد در تابع ذکر شود.
- آرگومنت‌ها باید در داخل قوس درج گردد اما توسط کامه از هم‌دیگر جدا گردد.
- آرگومنت می‌تواند ثابت، متحول و یا عبارت باشد.

تابع حسابی (math.h)

این تابع یکی از توابع مهم است که در فایل صدری (header file) به نام math.h ظاهر می‌شود (Meyers, 2005).

مثال ۱۸.۴: برنامه ذیل جذر مربع هر عدد وارد شده از صفحه کلید را دریافت می نماید.

```
// program to find Square Root of any number inserted from keyboard
#include <iostream.h>
#include <math.h> // required for sqrt function

main ( )
{
    // Declaration
    double Number, SQR;

    cout << "Enter a number to find its Square Root : ";
    cin >> Number;

    SQR = sqrt (Number);

    cout << " The square root of "<< Number << " = " << SQR;

    return 0;
}
```

Output

Enter a number to find its Square Root : 36

The square root of 36 = 6

مثال ۱۹.۴: برنامه ذیل هر عدد را به هر توان وارد شده دریافت می نماید.

```
#include <iostream.h>
#include <math.h> // required for pow function

main ( )
{
    // Declaration
    double Base, Exponent, Result;

    cout << "Enter value for Base and Power : ";
    cin >> Base >> Exponent;

    Result = pow(Base, Exponent);

    cout << Base << " ^ " << Exponent << " = " << Result;

    return 0;
}
```

Output

Enter value for Base and Power : 2 3

$$2^3 = 8$$

مثال ۲۰.۴: برنامه ذیل هر جذر هر عدد وارد شده را دریافت می‌نماید.

```
#include <iostream.h>
#include <math.h> // required for pow function
main ( )
{
    // Declaration
    double Base, Root, Result;
    cout << "Enter value for Base and Root : ";
    cin >> Base >> Root;
    Result = pow(Base, 1/Root);
    cout << Base << " Jazar " << Root << " = " << Result;
    return 0;
}
```

Output

Enter value for Base and Root : 32 5

$$32 \text{ Jazar } 5 = 2$$

مثال ۲۱.۴: برنامه ذیل لوگاریتم هر عدد وارد شده را دریافت می‌نماید.

```
#include <iostream.h>
#include <math.h> // required for log function
main ( )
{
    // Declaration
    double Number, Log1, Log2;
    cout << "Enter a number to find its logarithm : ";
```



```

cin >> Number;

Log1 = log10(Number); //Finds log of any number on base 10

Log2 = log(Number);    //Finds log of any number on base e

cout << " Log10 " << Number << " = " << Log1;

cout << " LogE " << Number << " = " << Log2;

return 0;

}

```

Output

Enter a number to find its logarithm : 100

$\text{Log}_{10} 100 = 2$

$\text{Log}_E 100 = 4.605$

مثال ۲۲.۴: برنامه ذیل ساین هر زاویه که از نوع درجه باشد دریافت می‌نماید.

```

#include <iostream.h>

#include <math.h> // required for sin function

main ( )

{

// Declaration

double Number, SN;

cout << "Enter an angle size to find its Sin : ";

cin >> Number;

SN = sin(Number * 3.14159 / 180);

cout << " Sin " << Number << " = " << SN;

return 0;

}

```

Output

Enter an angle size to find its Sin : 30

$\text{Sin } 30 = 0.5$

قابل یاد آوری است که کوساین (cos) و تانجانت (tan) هر زاویه را می‌توانید به شیوه فوق دریافت نمایید.

مثال ۲۳.۴: برنامه ذیل کوتانجانت هر زاویه که از نوع درجه باشد دریافت می نماید.

```
#include <iostream.h>
#include <math.h> // required for Cotangent function
main ( )
{
    // Declaration
    double Number, cot;
    cout << "Enter an angle size to find its Cotangent : ";
    cin >> Number;
    cot = 1/tan(Number * 3.14159 / 180);
    cout << " Cotangent " << Number << " = " << cot;
    return 0;
}
```

Output

Enter an angle size to find its Cotangent : 45
Cotangent 45 = 1

شما می توانید که سکینت $(1/\cos)$ و کوسکینت $(1/\sin)$ را به شیوه فوق دریافت نمایید.

مثال 24.4: برنامه ذیل قیمت $e = 2.7182$ را به توان عدد وارد شده از کیبورد بالامی برد.

```
#include <iostream.h>
#include <math.h>
main ( )
{
    // Declaration
    double Number, R;
    cout << "Enter a number: ";
    cin >> Number;
    R = exp (Number);
    cout << " Result = " << R;
    return 0;
}
```

Output

Enter a number : 2

Result = 7.389

مثال ۲۵.۴: برنامه ذیل قیمت مطلقه هر عدد وارد شده از صفحه کلید را دریافت می‌نماید.

```
#include <iostream.h>
#include <math.h>

main ( )
{
    // Declaration
    int N, R;
    cout << "Enter a number: ";
    cin >> N;
    R = abs (N);
    cout << " Result = " << R;
    return 0;
}
```

Output

Enter a number : -200

Result = 200



در این فصل عمل‌گرها (Operators)، عبارت‌ها (Expressions) و بیانیه‌های (Statements) C++ مورد بحث قرار گرفت و در نتیجه مفاهیم ذیل را آموختیم:

عمل‌گر عبارت از سمبول است که به‌خاطر اجرای عملیات خاص حسابی و یا منطقی استفاده می‌شود. در لسان C++ چندین نوع عمل‌گر وجود دارد که عبارت‌اند:

عمل‌گر یک‌تایی عمل‌گر است که بالای یک عنصر اجرا می‌شود؛ عمل‌گر دوتایی عمل‌گر است که بالای دو عنصر اجرا می‌شود.

عمل‌گر حسابی عمل‌گر است که غرض اجرای عملیات حسابی مانند جمع، منفی، ضرب، تقسیم و غیره عملیات استفاده می‌شود.

در حالی که عمل‌گرهای ارتباطی بیش‌تر در عملیات شرطی که نتیجه آن درست و یا نادرست می‌باشد، استفاده می‌گردد. این عمل‌گرها نتیجه هر عبارت را به شکل تام نشان می‌دهد. اگر نتیجه نادرست باشد آن را به 0 نمایش می‌دهد ولی اگر نتیجه درست باشد آن را به 1 نمایش می‌دهد.

عمل‌گرهای منطقی غرض یک‌جا نمودن دو یا بیش‌تر از دو عبارت منطقی استفاده می‌گردد. به طور عموم، سه عمل‌گر منطقی وجود دارد که عبارت‌اند از AND، OR و NOT.

هم‌چنان باید گفت که عبارت شامل: عمل‌گرها، ثابت‌ها و متحول‌ها می‌باشد که به چند نوع تقسیم گردیده است: عبارت‌های ثابت، عبارت‌های حسابی، عبارت‌های ارتباطی، عبارت‌های منطقی، عبارت‌های پایین‌ترها و عبارت‌های بیتی.

عبارت ثابت آن است که در ترکیب خود تنها قیمت‌های ثابت را داشته باشد. عبارت حسابی آن است که در ترکیب خود متحول‌ها، ثابت‌ها و عمل‌گرهای حسابی را داشته باشد. عبارت ارتباطی آن است که در ترکیب خود متحول‌ها، ثابت‌ها، عمل‌گرهای حسابی و مقایسوی را داشته باشد. عبارت منطقی آن است که در آن دویا بیش‌تر از دو عمل‌گر ارتباطی به کمک عمل‌گرهای منطقی وصل شده باشد. عبارت پایین‌تر قیمت آدرس حافظه را نشان می‌دهد. و بالاخره، عبارت بیتی نتیجه را به شکل بیت‌ها با استفاده از AND و OR منطقی نشان می‌دهد.



سوالات فصل چهارم

۱. عبارت (Expression) چیست؟
۲. یک مثال عمل گر یک تایی را ذکر کنید؟
۳. مفهوم اولویت بندی عمل گرها (Operator precedence) چیست؟
۴. عبارت (Expression) چیست و اجزای آن را نام بگیرید؟
۵. عبارت که چندین جفت قوس داشته باشد سلسله اجرای عملیات چطور اجرا می گردد؟
۶. عمل گرهای مختلف ارتباطی (Relational operators) را توضیح دهید؟
۷. عمل گرهای منطقی (Logical operators) را توضیح نموده، سلسله حق اولویت اجرای آن را ذکر نمایید؟
۸. سه تابع حسابی که در فصل فوق ذکر نشده دریافت و در لسان سی پلس پلس برنامه ریزی کنید؟

فعالیت ها

۱. تفاوت بین عمل گر / و % چیست؟
 ۲. تفاوت میان = و == چیست؟
 ۳. عبارات ذیل ریاضیکی را به عبارت C++ تبدیل کنید؟
- $$Y = \left| (a + b) / (c - d) \right|^{1/a}$$
- $$Y = \sqrt{\tan a} / a^b$$

فصل پنجم

بیانیه های تصمیم گیری یا شرطی Conditional Statements



هدف کلی: با دستوهای شرطی آشنا شوند.

اهداف آموزشی: در پایان این فصل محصلان قادر خواهند شد تا:

۱. دستوهای شرطی را تعریف نمایند.
۲. موارد استفاده دستوهای شرطی را شرح دهند.
۳. دستور شرطی IF را تعریف نمایند.
۴. تفاوت بین if و switch را تشریح دهند.

در این فصل دستورهای شرطی، انواع دستورهای شرطی و موارد استفاده آنها مورد بحث قرار گرفته که یکی پی دیگر به تفصیل توضیح می‌گردد.

در لسان C++ دستورها به‌طور عادی پی یک‌دیگر طور که نوشته شده اجرا (execute) می‌شود؛ این نوع برنامه‌ها دارای ساختار مسلسل (Sequential Structure) می‌باشد.

اما در بعضی موارد نیاز است که اجرای برنامه‌ها شکل مسلسل را نداشته باشد و مطابق به شرایط خاص اجرا گردد. برنامه‌های که تصمیم‌گیری در آن مربوط به شرط باشد به نام برنامه‌های انتخابی (Selection Structure) یاد می‌شود. پس می‌توان گفت دستورات شرطی برای انجام کارها و تصمیم‌گیری‌های مختلف بر اساس شرایط مختلف به کار می‌روند.

لسان C++ بیانیه‌های شرطی دارد که در مواقع نیاز می‌توانیم از آن استفاده نماییم.

این بیانیه‌ها عبارت اند از (Rama, 2011) :

- i. بیانیه If
- ii. بیانیه switch
- iii. عمل‌گر سه تایی
- iv. بیانیه goto

بیانیه (goto) وقتی استفاده می‌گردد که یک قسمت خاص از برنامه نظر به لزوم دید باید اجرا گردد.

Chapter 21 بیانیه شرطی (if)

بیانیه شرطی (If) یکی از قوی‌ترین بیانیه‌های زبان‌های برنامه‌نویسی کمپیوتر می‌باشد. این بیانیه شرط طرح شده را به شکل منطقی مورد ارزیابی قرار داده، در صورت صحت بودن آن عمل اول که توسط برنامه‌نویس طرح‌گرفته اجرا و در صورت نادرست بودن شرط عمل دوم اجرا می‌گردد.

شکل ساده بیانیه (if) قرار ذیل است:

If (conditional expression)

در (conditional expression) شرط خویش را مطرح می‌نماییم. S1 و S2 دو بیانیه است که یکی از آنها بعد از ارزیابی شرط توسط کمپیوتر اجرا می‌گردد.

- به صورت عموم، هر (conditional expression) دارای یک قیمت است.
- (conditional expression) باید در داخل قوس‌های خورد () نوشته شود.
- بیانیه S1 وقتی اجرا می‌شود که شرط درست باشد.

• اگر شرط نادرست باشد در آن صورت بیانیه S1 صرف نظر می‌شود و بیانیه دومی (S2) اجرا می‌گردد.

بیانیه S1 می‌تواند که یک بیانیه ساده و یا ترکیبی باشد. بیانیه ساده حاوی یک سطر بوده در حالی که بیانیه ترکیبی شامل چندین سطر می‌باشد.

اگر شرط درست باشد و قرار باشد که چندین سطر اجرا گردد در آن صورت تمام سطرها باید داخل قوس‌های بزرگ (Braces) نوشته گردد.

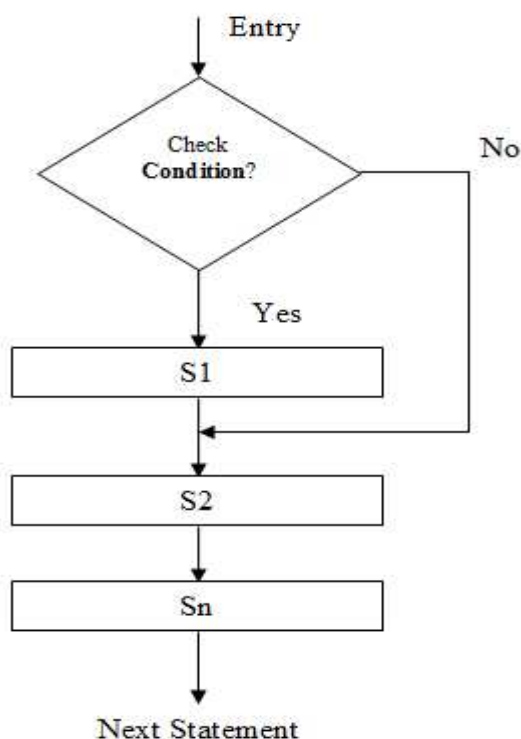
طورمثال:

```
If (Condition)
{
    S1;
    S2;
    S3;
}
```

در صورتی که شرط درست باشد تمام بیانیه‌های داخل قوس بزرگ اجرا می‌شوند.

شکل ساده دستور شرطی If را می‌توانیم در فلوچارت ذیل

نمایش داد.



شکل ۱.۵ فلوچارت بیانیه شرطی (if)

مثال ۱.۵: برنامه ذیل شیوه استفاده دستور شرطی (if) را نشان می‌دهد.

```
/* Program to show usage of simple if () */
#include <iostream.h>
main ( )
{
    int x , y = 10 ;
    cout << " \n Enter value of x  : " ;
    cin >> x ;
    if ( x > 0 )
        cout << " \n Sum = " << x + y;
    return 0 ;
}
```

Output

Enter value of x : 5
Sum = 15

مثال ۲.۵: برنامه ذیل یک عدد را از صفحه کلید می‌گیرد ؛ اگر عدد وارد شده مثبت باشد آن را ضرب ۱۰ می‌سازد.

```
/* Program multiply only positive number * 10 inserted from keyboard */
#include <iostream.h>
main ( )
{
    int x ;
    cout << " \n Enter a number  : " ;
    cin >> x ;
    if ( x > 0 )
        cout << " \n Result = " << x * 10;
    return 0 ;
}
```

Output

Enter a number : 20
Result = 200

یک تعداد مثال‌های ساده دیگر دستور شرطی if

- a) if (balance == 0)
- ```
 cout << "\n No more shopping !" ;
```
- b) if ( a > b && a > c )
- ```
    cout << "\n Largest number is " << a ;
```
- c) int rain = 1;
- ```
if (rain == 1)
{
 cout << "\n Wear a raincoat!" ;
 cout << "\n or hold an umbrella " ;
}
```
- d) if ( a > 0 && b < 10 )
- ```
{
    Sum = a + b ;
    Diff = a - b ;
    cout << "\n Sum = " << Sum ;
    cout << "\n Diff = " << Diff ;
}
```
- e) if (dues > 0)
- ```
{
 cout << "\n Account number is overdue " << accno ;
 credit = 0 ;
}
```
- f) if ( a == b || a == c || b == c )
- ```
{
    cout << "\n It is an isosceles triangle " ;
}
```

مثال ۳.۵: برنامه ذیل عدد بزرگتر بین دو عدد وارد شده را دریافت می نماید.

```
/* Program to find largest of two numbers */
#include <iostream.h>
main ( )
{
    int a , b, large ;
    cout << "\n Type 2 numbers : ";
    cin >> a >> b ;
    // Assuming largest number is a
    large = a ;
    if ( b > large )
        large = b ;
    cout << "\n Largest of " << a << " and " << b << " is " << large ;
    return 0;
}
```

Output

Type 2 numbers : 15 48

Largest of 15 and 48 is 48

مثال ۴.۵: برنامه ذیل تفاوت بین = و == را در صورت که دستور شرطی استفاده گردیده باشد نشان می دهد.

```
/*Program to show what happens when = is given instead of == in if statement/*
#include <iostream.h>
main( )
{
    int x = 0 ;
    if ( x = 0) // condition becomes false since x = 0
        cout << " x is 0, x==0 is true, but this statement will not be output /n ;"
        if (x!=0)
            cout << " x is 0, x!=0 is false, so this statement will not be output /n;"
    if (x=15) // condition becomes true since value of x is not zero
```

<pre>cout << " can you believe it, x is actually! " <<x; return 0 ; }</pre>
Output Can you believe it, x is actually! 15

در مثال فوق () if اول قیمت 0 را به x انتقال می‌دهد و نتیجه به false می‌انجامد و به همین دلیل بیانیۀ که در داخل cout قرار دارد نشان نمی‌دهد.

در () if دومی قیمت x خلاف 0 تعریف گردیده است و باز هم نتیجه به نادرستی می‌انجامد و بیانیۀ که در داخل cout نوشته شده نشان نمی‌دهد. در if سومی قیمت x مساوی به 15 در نظر گرفته شده که خلاف 0 می‌باشد و به همین خاطر نتیجۀ داخل cout را روی سکرین نشان می‌دهد.

If (Condition)
S1;
else
S2;
S3;

۱.۲۰ بیانیۀ شرطی THE IF () ELSE STATEMENT

این دستور دارای پاسخ دو گزینه است (Eckel, 2000).

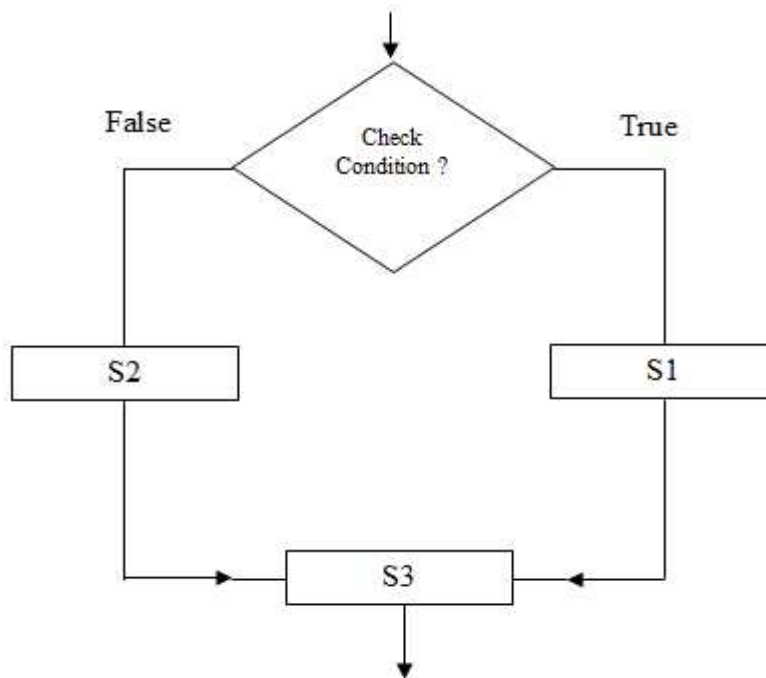
شکل عمومی آن قرار ذیل است:

• اگر شرط درست باشد S1 اجرا می‌گردد در غیر آن S2 اجرا خواهد شد (وقتی که شرط نادرست باشد).

• در هر حالت تنها یک بیانیۀ (S1 و یا S2) اجرا می‌شود. امکان ندارد که هر دو بیانیۀ در عین زمان اجرا گردد.

• اگر شرط درست و یا نادرست باشد یکی از بیانیۀ های S1 و یا S2 اجرا می‌شود و نوبت به S3 می‌رسد.

○ شکل دستور شرطی if () – else در فلوچارت ذیل نمایش داده شده است.



شکل ۲.۵ فلوجارت بیانیه شرطی - if else statement

مثال ۵.۵: برنامه ذیل یک عدد را از صفحه کلید می‌گیرد ، اگر عدد وارد شده مثبت باشد در آن صورت کلمه "Positive" و اگر منفی باشد کلمه "Negative" را روی سکرین نشان می‌دهد.

```

/* Program finds whether the entered number is positive or negative */
#include <iostream.h>
main ( )
{
    int N;
    cout << " \n Enter a number to find whether it is positive or negative " ;
    cin >> N ;
    if ( N > 0 )
        cout << " \n The Number is Positive ";
    else
        cout << " \n The Number is Negative ";
    return 0 ;
}
  
```

Output

Enter a number to find whether it is positive or negative: 5

The Number is Positive

Enter a number to find whether it is positive or negative: - 5

The Number is Negative

مثال ۶.۵: برنامه ذیل یک حرف را از صفحه کلید می گیرد ، اگر حرف وارد شده A باشد در آن صورت پیام " You Entered A " و اگر خلاف آن باشد کلمه " Wrong Character " را نشان می دهد.

```
/* Program multiply only positive number * 10 inserted from keyboard */
#include <iostream.h>
main ( )
{
    char chr ;
    cout << " \n Enter an alphabit character : " ;
    cin >> chr ;
    if ( chr == 'A')
        cout << " \n You Entered A ";
    else
        cout << " \n You Entered a Wrong Character ";
    return 0 ;
}
```

Output

Enter an alphabit character : A

You Entered A

Enter an alphabit character : X

Wrong Character

مثال ۷.۵: برنامه ذیل شیوه استفاده else - () if را نشان می‌دهد.

```
// Program to illustrate the usage of if ( ) - else
#include <iostream.h>
#include <stdlib.h>    //srand , rand
#include <time.h>      // time
main ( )
{
    int magic, guess ;
    srand (time (NULL)); //initialize random seed
    magic =rand() % 5;    // generate random number between 1 to 5
    cout << " \n Guess the magic number : " ;
    cin >> guess ;
    if (guess == magic )
        cout << "\n YOU got it ***** Right ***** " << magic;
    else
        cout << "\n Sorry ! Better luck next time. " << magic;
    return 0;
}
```

Output

Guess the magic number: 2

Sorry ! Better luck next time. 1

مثال ۸.۵: برنامه ذیل عدد واردشده به برنامه را ارزیابی نموده که آیا جفت است و یا تاق.

```
// Program to determine if a given number is even or odd
#include <iostream.h>
main ( )
{
    int num, remain ;
    cout << " \n Enter the number to be tested : " ;
    cin >> num ;
```

```

    remain = num % 2 ;
    if (remain == 0 )
    cout << num << " is Even" ;
    else
    cout << num << " is Odd" ;
    return 0;
}

```

Output

```

Enter the number to be tested : 1279
1279 is Odd

Enter the number to be tested : 38
38 is Even

Enter the number to be tested : 0
0 is Even      ( Why ? )

```

مثال ۹.۵: برنامه ذیل سال وارد شده را ارزیابی نموده که سال کبیسه است یا خیر؟.

```

/* Program to check whether a given year is a leap year */
#include <iostream.h>
main ( )
{
    int yr, rem_4, rem_100, rem_400 ;
    cout << " \n Enter the 4 digit year to be checked : " ;
    cin >> yr ;
    // to find the remainders of division by 4, 100 & 400
    rem_4 = yr % 4 ;
    rem_100 = yr % 100 ;
    rem_400 = yr % 400 ;
    if ((rem_4 == 0 && rem_100 != 0) || rem_400 == 0)
    cout << "\n is a leap year " << yr ;
    else
    cout << "\n is not a leap year ok !" << yr ;
    return 0;
}

```



```
}
```

Output

```
Enter the 4 digit year to be checked : 2004
2004 is a leap year
Enter the 4 digit year to be checked : 1998
1998 is not a leap year ok !
Enter the 4 digit year to be checked : 3000
3000 is not a leap year ok !
```

مثال ۱۰.۵: برنامه ذیل قیمت مطلقه عدد وارد شده را دریافت می نماید.

```
// Program to calculate absolute value of an integer
#include <iostream.h>
main ( )
{
    int num ;
    cout << " \n Type a number : " ;
    cin >> num;
    if (num < 0)
    {
        num = - num ;
        cout << " \n The absolute value is " << num ;
    }
    else
        cout << " \n The absolute value is " << num ;
    return 0 ; }
```

Output

```
Type a number : -456
The absolute value is 456
Type a number : 360
The absolute value is 360
Type a number : 0
```

The absolute value is 0 (how ?)

مثال ۱۱.۵: برنامه ذیل دو قیمت وارد شده را به شکل نزولی ترتیب می‌کند.

```
// Program to display 2 numbers in descending order
#include <iostream.h>
main ( )
{
    int num1, num2 ;
    cout << " \n Enter 2 numbers : " ;
    cin >> num1 >> num2;
    cout << " Number in Descending order are : " ;
    if (num1 >= num2)
        cout << "\n " << num1 << num2 ;
    else
        cout << "\n " << num2 << " " << num1 ;
    return 0 ; }
```

Output

```
Enter 2 numbers : 25 31
31 25
Enter 2 numbers : -40 10
10 -40
```

مثال ۱۲.۵: برنامه ذیل بزرگ‌ترین و عدد بزرگ‌تر بین ۴ عدد وارد شده را دریافت می‌کند.

```
/* Program to find the largest & second largest of 4 numbers and their average */
#include <iostream.h>
main ( )
{
    int m1, m2, m3, m4, first, second ;
    float avg ;
    cout << " \n Enter the marks obtained in 4 papers : " ;
    cin >> m1 >> m2 >> m3 >> m4;
    if (m1 > m2)
```

```

        first = m1, second = m2 ;
    else
        first = m2, second = m1 ;
    if (m3 > first)
        second = first, first = m3 ;
    else if ( m3 > second )
        second = m3 ;
    if (m4 > first)
        second = first, first = m4 ;
    else if ( m4 > second )
        second = m4 ;
    cout << "\n " << "Largest marks " << first ;
    cout << "\n " << "Second Largest marks " << second ;
    avg = ( first + second ) / 2.0 ;
    cout << "\n " << " Average of best two marks = " << avg ;
    return 0;
}

```

Output

```

Enter the marks obtained in 4 papers : 56 78 40 48
Largest marks = 78
Second Largest marks = 56
Average of best two marks = 67.00

```

مثال ۱۳.۵: برنامه ذیل نتایج هر مضمون را در چهار امتحان نشان می‌دهد.

```

/* Program to declare result in each subject in 4 papers */
#include <iostream.h>
main ( )
{
    int m1, m2, m3, m4;
    cout << " \n Enter the marks in 4 papers \n " ;
    cin >> m1 >> m2 >> m3 >> m4;
    if (m1 > 59)

```

```

        cout << " FIRST CLASS in paper 1 \n";
    else if ( m1 > 39 )
        cout << " SECOND CLASS in paper 1 \n";
    else
        cout << " Fail in paper 1 \n" ;

    if (m2 > 59)
        cout << " FIRST CLASS in paper 2 \n";
    else if ( m2 > 39 )
        cout << " SECOND CLASS in paper 2 \n";
    else
        cout << " Fail in paper 2 \n" ;

    if (m3 > 59)
        cout << " FIRST CLASS in paper 3 \n";
    else if ( m3 > 39 )
        cout << " SECOND CLASS in paper 3 \n";
    else
        cout << " Fail in paper 3 \n" ;

    if (m4 > 59)
        cout << " FIRST CLASS in paper 4 \n";
    else if ( m4 > 39 )
        cout << " SECOND CLASS in paper 4 \n";
    else
        cout << " Fail in paper 4 \n" ;

    return 0;
}

```

Output

Enter the marks in 4 papers :

38 40 24 60

Fail in paper 1
SECOND CLASS in Paper 2
FAIL in paper 3
FIRST CLASS in paper 4

مثال ۱۴.۵: برنامه ذیل شیوه استفاده (char) را در دستور شرطی if () - else نشان می دهد.

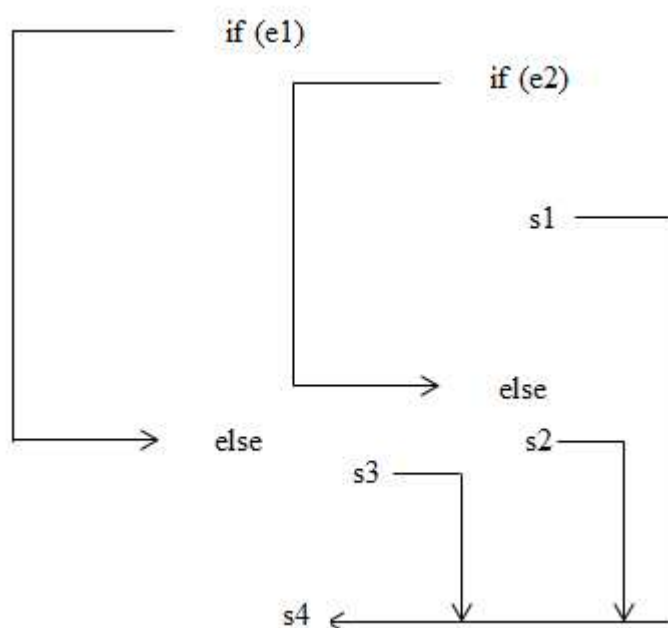
```
// Program to show usage of char in if ( ) statement
#include <iostream.h>
#include <stdio.h>
main ( )
{
    char wish;
    cout << " \n Want to got to movie Y/ N ?:" ;
    wish = getchar( );
    if (wish == 'Y' || wish == 'N')
        cout << "\n " << " Go and get ready quickly ! " ;
    else
        cout << "\n " << " Go and start studying ! " ;
    return 0;
}
```

Output

Want to go to movie Y / N ? Y
Go and get ready quickly !
Want to go to movie Y / N ? n
Go and start studying !
Want to go to movie Y / N ? y
Go and start studying !
Want to go to movie Y / N ? A
Go and start studying !

۱.۲۱ یک شرط داخل شرط دیگر NESTED IF () STATEMENT

وقتی که یک If داخل بلاک If دیگر قرار گیرد چنین بیانیه شرطی را به نام (Nested if) یاد می‌کند. شکل عمومی (NESTED IF) قرار ذیل است (Soulié, 2007):



شکل ۳.۵ فلوجارت بیانیه شرطی Nested if statement

در این جا $e1$ و $e2$ شروط و $S1$ ، $S2$ ، $S3$ و $S4$ بیانیه‌ها هستند.

- هرگاه شرط $e1$ درست باشد مترجم به تعقیب آن شرط $e2$ را چک می‌کند. اگر شرط $e2$ هم درست باشد در آن صورت بیانیه $s1$ اجرا می‌شود و کنترل به بیانیه $s4$ منتقل می‌گردد. اگر شرط $e2$ نادرست باشد در آن صورت بیانیه $s2$ و به تعقیب آن $S4$ اجرای گردد.
- اگر شرط $e1$ نادرست باشد در آن صورت بیانیه $S3$ و به تعقیب آن بیانیه $S4$ اجرای گردد.
- امکان دارد که $S1$ ، $S2$ ، $S3$ حاوی بیانیه‌های دیگر $\text{if} - \text{else}$ باشند که این حالت را به نام Multi - Layer Nesting یاد می‌کند.

مثال ۱۵.۵: برنامه ذیل شیوه استفاده () nested if را نشان می‌دهد.

```
// Program to show usage of nested if ( )
#include <iostream.h>

main ( )
{
    float hrs;
```

```

cout << " \n Type the line in hours :";
cin >> hrs ;
if (hrs >= 0.0 && hrs < 12.0)
    cout << "\n " << " Good Morning " ;
else
    if (hrs >= 12.0 && hrs < 17.0 )
        cout << "\n " << " Good Afternoon" ;
    else
        if (hrs >= 17.0 && hrs<21.0)
            cout << "\n Good Evening ";
        else
            if (hrs >= 21.0 && hrs < 24.0)
                cout << "\n Good Night ";
            else
                cout << "\n Wrong Input ";
                return 0;
}

```

Output

```

Type the line in hours : 20.6
Good Evening
Type the line in hours : 5.0
Good Morning
Type the line in hours : 22
Good Night
Type the line in hours : 15
Good Afternoon
Type the line in hours : 50
Wrong Input

```

مثال ۱۶.۵: برنامه ذیل بزرگ‌ترین عدد را بین ۳ عدد وارد شده با کمک () nested if دریافت می‌نماید.

```
// Program to find the largest of three numbers using nested if ( ) – else
#include <iostream.h>

main ( )
{
    int a, b, c, large ;
    cout << " \n Enter 3 numbers :";
    cin >> a >> b >> c ;

    {
        if ( a > b )
        {
            if (a > c)
                large = a;
            else
                large = c;
        }
        else
        {
            if (b > c)
                large = b;
            else
                large = c;
        }
    }

    cout << " \n Largest Number is " << large; }
```

Output

```
Enter 3 numbers : 22 6 11
Largest Number is 22

Enter 3 numbers : 10 50 42
Largest Number is 50

Enter 3 numbers : 13 6 80
Largest Number is 80
```

مثال ۱۷.۵: برنامه ذیل کرکتر وارد شده به برنامه را می‌خواند و دریافت می‌نماید که کرکتر وارد شده حرف الفبا، رقم و یا از جمله سمبول‌های خاص (special characters) است.

```
/* Program to read a character and determine whether it is an alphabet, digit or a
special character */

#include <iostream.h>
```



```

main ( )
{
    char ch;
    cout << " \n Enter a single character please : " ;
    cin >>ch;
    if ((ch>='a' && ch <='z') || (ch >= 'A' && ch <= 'Z'))
        cout << "\n " << " It is an Alphabet" ;
    else
        if ( ch >='0' && ch <= '9' )
            cout << "\n " << " It is a digit" ;
    else
        cout << "\n It is a special character ";
    return 0; }

```

Output

```

Enter a single character please : u
It is an Alphabet
Enter a single character please : !
It is a special character
Enter a single character please : 7
It is a digit

```

برنامه فوق برای حروف بزرگ و کوچک مورد استفاده قرار می گیرد. وقتی که ch قیمت را در خود ذخیره کند که آن نه حرف و نه هم عدد باشد در آن صورت آن را به حیث سمبول خاص قبول می کند.

مثال ۱۸.۵: برنامه ذیل جذرهای معادلات یک مجهوله درجه ۲ را دریافت می کند.

```

/* Program to find the roots of a quadratic equaiton */
#include <iostream.h>
#include <math.h>
main ( )
{
    float a, b, c,d;

```

```

float x, x1, x2;
float rpart, ipart;
cout << "\n Enter 3 numbers :";
cin >> a >> b >> c ;
// solution to equation of the form bx + c = 0
if (a==0)
{
    x = - b / c ;
    cout << "\n Only root " << x ;
}
// Calculate discriminant d
d = b * b - 4 * a * c;
// Solution with real and distant roots
if (d > 0)
{
    cout << "\n Real and distant roots are : ";
    x1 = (-b + sqrt (d)) / (2*a) ;
    x2 = (-b - sqrt (d)) / (2*a) ;
    cout << "\n x1 = " << x1 << " \n " << " x2 = " << x2;
}
else
if (d ==0)
{
    // Solution with repeated roots
    cout << "\n Repeated Roots are : ";
    x1 = -b / (2*a) ;
    x2 = x1 ;
    cout << "\n x1 & x2 = " << x1 ;
}
else
{

```

```

// Solution with complex roots
d = sqrt(abs(d));
rpart = -b / (2 * a);
ipart = d / (2 * a);
cout << "\n Complex roots are : ";
cout << "\n x1 = " << rpart << " + i " << ipart; // Real part of a complex root
        cout << "\n x2 = " << rpart << " + i " << ipart; // Imaginary part of a
complex root
    }
    x = - b / c;
    cout << "\n Only root " << x;
}

```

Output

Enter 3 numbers : 1 4 1

Real and distinct roots are :

x1 = -0.267

x2 = -3.732

Enter 3 numbers : 4 4 1

Repeated roots are :

x1 & x2 = -0.5

Enter 3 numbers : 1 2 3

Complex roots are :

x1 = -1 + i 1.41421

x2 = -1 + i 1.41421

مثال 19.5: برنامه ذیل 2 عدد را از کیبورد گرفته مطابق به علامه وارد شده (+, -, *, /) اجراء می کند.

```
// Program to simulate the 4 arithmetic operations +, -, *, /
// for 2 real number depending on the operators
#include <iostream.h>
main ( )
{

    float num1, num2 ;
    char op;
    cout << " \n Enter an operator :";
    cin >> op;
    cout << " \n Enter 2 numbers :";
    cin >> num1 >> num2 ;

    { if (op=='+')
        cout << " \n Sum = " << num1 + num2 ;
      else
        { if (op == '-')
            cout << " \n Differnce = " << num1 - num2 ;
          else
            { if (op == '*')
                cout << " \n Product == " << num1 * num2 ;
              else
                { if ((op == '/') && num1 != 0)
                    cout << " \n Quotient = " << num1 / num2 ;
                  else
                    cout << " \n Error in Input ";
                }
            }
          }
    }

    return 0;
}
```

Output

```
Enter an operator : +  
Enter 2 nummbers : 5 4  
Sum = 9.00  
Enter an operator : -  
Enter 2 nummbers : 5 8  
Difference = -3.00  
Enter an operator : *  
Enter 2 nummbers : 12.4 6  
Product = 74.40  
Enter an operator : /  
Enter 2 nummbers : 12 4  
Quotient = 3.00  
Enter an operator : &  
Enter 2 nummbers : 2 3  
Error in Input
```

مثال ۲۰.۵: برنامه ذیل بزرگترین و یا عدد بزرگتر دوم را بین ۳ عدد وارد شده دریافت می‌نماید.

```
/* Program to find the largest & second largest of 3 numbers */
```

```
#include <iostream.h>
```

```
main ( )
```

```
{
```

```
    int a, b, c, first, second ;
```

```
    char op;
```

```
    cout << " \n Enter 3 numbers : " ;
```

```
    cin >> a >> b >> c ;
```

```
    {
        if (a > b)
        {
            if (a > c)
            {
                first = a ;
            }
            {
                if (b > c)
                {
                    second = b ;
                }
                else
                {
                    second = c;
                }
            }
            else
            {
                first = c;
                second = a;
            }
        }
        else
        {
            if (b > c)
            {
                first = b;
            }
            {
                if (a > c)
                {
                    second = a;
                }
                else
                {
                    second = c;
                }
            }
        }
    }
```

```
    }  
    else  
    {  
        first = c;  
        second = b;  
    }  
    cout << "\n Largest is = " << first << "\n Second largest is = " << second;  
}
```

Output

Enter 3 numbers 15 2 10

Largest is 15

Second Largest is 10

Enter 3 numbers 28 65 5

Largest is 65

Second Largest is 28

Enter 3 numbers 79 12 6

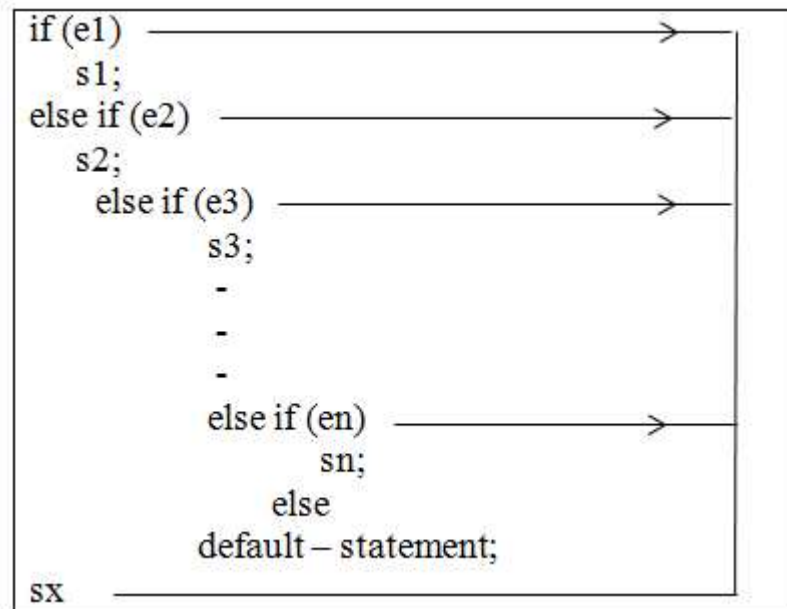
Largest is 79

Second Largest is 12

۱.۲۲ بیانیه () THE ELSE – IF

شیوه دیگر که با استفاده از آن می‌توانیم else – if () nested را توضیح دهیم عبارت از else – if () می‌باشد که در آن هر else بیانیه if را نیز با خود دارد (Nawaz, 2006).

شکل عمومی () else – if قرار ذیل است:



شکل ۴.۵ فلوچارت بیانیه شرطی if – else statement

- $e1, e2, e3, \dots, en$ شروط و $s1, s2, sn$ بیانیه‌های default و sx بیانیه عادی است.
- در این فلوچارت شرایط از بالا به پایین چک می‌گردد.
- در مرحله اول $e1$ امتحان می‌شود. اگر درست باشد $s1$ اجرا می‌شود و کنترل به sx انتقال می‌یابد در غیر آن $e2$ امتحان می‌گردد. اگر درست باشد $s2$ اجرا می‌گردد و کنترل به sx انتقال می‌یابد و تمام بیانیه‌های متباقی را صرف نظر می‌نماید.
- اگر تمام شرایط نادرست باشد $e1$ و $e2, \dots, en$ در آن صورت بیانیه‌های عادی (Default Statement) اجرا گردیده و به تعقیب آن بیانیه sx اجرا می‌گردد.

مثال ۲۱.۵: برنامه ذیل علامه عدد وارد شده را دریافت می نماید.

```
/* Program to find sign of a number */
#include <iostream.h>

main ( )
{
    int num , sign;
    cout << " \n Enter a number please : " ;
    cin >> num;
    // check if number is negative
    if (num< 0)
        sign = -1 ;
    else if (num==0)
        sign = 0;
    else // number must be positive
        sign = 1;
    cout <<" sign = " << sign;
    return 0;
}
```

Output

```
Enter a number please :-259
sign = -1
Enter a number please : 0
sign = 0
Enter a number please :5
sign = 1
```

یک استاد، سه امتحان (امتحان 1، امتحان 2 و امتحان نهایی) را از محصلان خود اخذ می نماید که هر یک از این امتحانات از 50 نمره محاسبه می شود. بلندترین نمره از بین امتحان 1 و امتحان 2 هر محصل انتخاب گردیده و به نمرات امتحان نهایی جمع می شود. قابل ذکر است که مجموعه از 100 نمره محاسبه می شود.

نمرات متباقی طور ذیل محاسبه می گردد:

Marks	Grade
0 - 39	F
40 - 49	D
50 - 59	C
60 - 74	B
75 - 100	A

با استفاده از لسان برنامه نویسی C++ پروگرامی را بنویسید که نمرات هر سه امتحان را از کیبورد بگیرد و بعد از طی مراحل نمره نهایی را روی سکرین نشان دهد؟

مثال ۲۲.۵: برنامه ذیل با در نظر داشت قاعده فوق نمره نهایی یک محصل را دریافت نموده روی سکرین نشان می دهد.

```
//Program to print the grade of a student
#include <iostream.h>
main ( )
{
    int test1, test2, final, marks;
    char grade;
    cout << " \n Enter marks in Test1, Test2, final : " ;
    cin >> test1 >> test2 >> final;
    // calculate total marks out of 100
    if (test1 > test2)
        marks = final + test1 ;
    else
        marks = final + test2 ;
    // assigning grades
    if (marks >= 0 && marks < 40)
        grade = 'F' ;
    else if (marks < 50 )
        grade = 'D';
    else if (marks < 60 )
        grade = 'C';
```

```

else if (marks <75 )
    grade = 'B';
else if (marks <= 100 )
    grade = 'A';
else
    grade = 'X' ;
if (grade != 'X')
    cout <<" Grade is = " << grade ;
else
    cout << "Error in input !!";
return 0;
}

```

Output

```

Enter marks in Test1, Test2, final : 50  50  25
Grade is  A
Enter marks in Test1, Test2, final : 20  10  30
Grade is  C
Enter marks in Test1, Test2, final : 20  10  15
Grade is  F
Enter marks in Test1, Test2, final : 40  39  80
Error in input !!

```

یک کمپنی 4 کتگوری کارمندان دارد. پرداخت کرایه به آن ها مطابق قاعده ذیل صورت می گیرد:

Catagory1 → 30% of salary
 Catoagory2 → 20% of salary
 Catoagory3 → 15% of salary
 Catoagory4 → 10% of salary

مثال ۲۳.۵: بادر نظر داشت قاعده فوق برنامه را بنوسید که معاش و کتگوری کارمند را در نظر گرفته مطابق آن اندازه کرایه آن را معلوم نماید؟

```
// Program to calculate rent allowance based on employee category
#include <iostream.h>
main ( )
{
    int cat ;
    float sal, rent ;
    cout << " \n Enter the category and salary of employee :";
    cin >> cat >> sal ;
    if (cat == 1)
        rent = sal * 0.3 ;
    else if (cat == 2)
        rent = sal * 0.2 ;
    else if (cat == 3)
        rent = sal * 0.15 ;
    else if (cat == 4)
        rent = sal * 0.1 ;
    else
    {
        cout << "\n Wrong Input " ;
    }
    cout << "\n The rent allowance = " << rent ;
    return 0;
}
```

Output

```
Enter the citatory and salary of Employee : 1 35000
The rent allowance = 10500.00
Enter the citatory and salary of Employee :4 10000
The rent allowance = 1000.00
Enter the citatory and salary of Employee : 23 3000
Wrong Input
```

۱.۲۳ بیانیه شرطی switch

استفاده از switch وقتی موثر است که در برنامه، چندین بیانیه وجود داشته باشد و خواسته باشیم که یک عده بیانیه‌های خاص نظر به شرط طرح شده اجـرا گـردد. برای چنین مسأله می‌توانیم از بیانیه if () – else نیز استفاده نماییم. اما وقتی شرایط زیاد باشد در آن صورت موارد استفاده if () – else پی‌چیده‌تر می‌گردد به خاطر که شما نیاز دارید تعداد زیاد از بیانیه‌ها را به منظور حل چنین مسأله نوشته کنید. در هم‌چو حالت بهتر است که از بیانیه شرطی switch استفاده صورت گیرد (Backman, 2012).

شکل عمومی بیانیه switch قرار ذیل است:

```
switch ( variable or expression )
```

```
{
```

```
    case Val1      :      s1;  
                                break;
```

```
    case Val2      :      s2;  
                                break;
```

```
    .....        :
```

```
    .....        :
```

```
    case Valn      :      sn  
                                break;
```

```
    default        :      sd  
                                break;
```

```
}
```

```
statement – x ;
```

شکل ۵.۵ شکل بیانیه switch

- switch یک دستور کلیدی است؛ قیمت متحول یا عبارت که در داخل قوس‌ها (Parenthesis) نوشته شده را چـک می‌کند. switch به ادامه خود لست قیمت‌های case دارد (case3, case2, case1) که به نام لیبل‌های کیس (case labels) یاد می‌گردد.
- قیمت متحول یا عبارت (variable or expression) باید تام (int) و یا یک حرف (character) باشد. قیمت case label مانند (case1, case 2 , case3 و غیره) باید ثابت باشد و نتیجه نیز به عدد تام انجامد.
- قیمت هر case باید یگانه (unique) باشد. قیمت می‌تواند به هر order باشد.

- بیانیه S1 و S2 می‌تواند یک سطر و یا بیش‌تر از آن باشد.
 - در آخر هر case label باید دو نقطه سر به سر یا کالن (:) گذاشته شود و تمام case ها در داخل یک جوره قوس‌ها (curely braces) نوشته .
 - وقتی که بیانیه switch اجرا گردد در آن صورت تمام بیانیه های مربوط به case آن نیز اجرا می‌گردد. اگر قیمت switch با هیچ قیمت case صدق نکند در آن صورت تمام بیانیه‌های مربوط به Default اجرامی گردد.
 - نوشتن بیانیه default اختیاری است. اگر وجود نداشته باشد در آن صورت بیانیه x اجرا می‌گردد.
 - نوشتن break در اخیر بلاک هر case لازمی است به‌خاطر این که ختم case موجود را نشان می‌دهد. اگر نوشته نگردد در آن صورت تمام case ها یکی پی دیگر اجرا می‌گردد.
 - نوشتن قیمت اعشاری در case label مجاز نیست.
 - شما می‌توانید switch را به شکل nested نیز استفاده کنید.
- چند برنامه switch قرار ذیل است:

مثال ۲۴.۵: برنامه ذیل تاریخ Julian 4 را نشان می‌دهد.

```
/* Program to find the Julian Date */
#include <iostream.h>
main ( )
{
    int dd, mm, yy;
    int leap, julian = 0;
    cout << " \n Enter a valid date : " ;
    cin >> dd >> mm >> yy;
    // To check for leap year
    leap = (yy % 4 == 0 && yy % 100 != 0) || yy % 400 == 0;
    julian + dd;
    switch ( mm - 1 ) {
        case 11: julian = julian + 30 ;
        case 10: julian = julian + 31 ;
```

⁴ a 5 digit number, consisting of a 2 digit year and a 3 digit day-of-year number

```

case 9: julian = julian + 30 ;
case 8: julian = julian + 31 ;
case 7 : julian = julian + 31 ;
case 6 : julian = julian + 30 ;
case 5 : julian = julian + 31 ;
case 4 : julian = julian + 30 ;
case 3 : julian = julian + 31 ;
case 2 : if ( leap );
julian += 29;
julian += 28;
case 1 : julian = julian + 31;
}
cout << "\n Julian day is : " << julian ;
return 0;
}

```

Output

```

Enter a valid date : 23  4  2005
Julian day is : 119

```

مثال ۲۵.۵: برنامه ذیل ۲ عدد را از کیبورد گرفته مطابق به علامه وارد شده (+, -, *, /) اجرات می‌نماید.

```

/* Program to stimulate the 4 arithmetic operations +, -, *, / for 2 real numbers
depending on the operator */
#include <iostream.h>
main ( )
{
    float num1, num2;
    char op;
    cout << " \n Enter an operator : " ;
    cin >> op;
    cout << " \n Enter 2 numbers : ";
    cin >> num1 >> num2 ;
}

```

```

switch ( op)
{
    case '+':
        cout << " \n Sum = " << num1 + num2 ;
        break;
    case '-':
        cout << " \n Difference = " << num1 - num2 ;
        break;
    case '*':
        cout << " \n Product = " << num1 * num2 ;
        break;
    case '/':
        if (num1 != 0)
            cout << " \n Quotient = " << num1 / num2 ;
        else
            cout << " \n Division by zero ? ";
        break;
    default :
        cout << " \n Error in Input " ;
        return 0;
}
}

```

Output

```

Enter an operator :  +
Enter 2 numbers : 6 9
Sum = 15.00

```


مثال ۲۶.۵: با در نظر داشت قاعده فوق برنامه را بنویسید که معاش و کتگوری کارمند را در نظر گرفته مطابق آن اندازه کرایه آن را معلوم نماید؟

```
/* Program to calculate rent allowance based on employee atagory */
#include <iostream.h>

main ( )
{
    int cat ;
    float sal, rent;
    cout << " \n Enter the category and salary of Employee : " ;
    cin >> cat >> sal;
    switch ( cat )
    {
        case 1:
            rent = sal * 0.3;
            break;
        case 2:
            rent = sal * 0.2;
            break;
        case 3:
            rent = sal * 0.15 ;
            break;
        case 4:
            rent = sal * 0.1;
            break;
        default :
            cout << " \n Wrong Input ";
            break;
    }

    cout << " \n The rent allowance = " << rent;
    return 0;
}
```

Output

Enter the category and salary of Employee : 2 36000

The allowance = 7200.00

مثال ۲۷.۵: برنامه ذیل جذرهای معادله یک مجهوله درجه ۲ را دریافت نموده روی سکرین نشان می دهد.

```
/* Program to find the roots of a quadratic equation using switch statement */
#include <iostream.h>

main ( )
{
    int flag;
    float a, b, c, d;
    float x, x1, x2;
    float rpart, ipart;
    cout << " \n Enter 3 numbers :";
    cin >> a >> b >> c;
    // solution to equation of the form bx + c = 0
    if (a==0)
    {
        x = -b / c;
        cout << "Only root = " << x;
    }
    d = b * b - 4 * a * c;
    if ( d > 0 )
        flag = 1;
    else if ( d == 0)
        flag = 2;
    else
        flag = 3;

    // Solution using switch-case default statement
    switch ( flag )
```

```

{
    case 1:
        cout << "\n Real and distinct roots are : " ;
        x1 = (-b + sqrt (d)) / (2 * a) ;
        x2 = (-b - sqrt (d)) / (2 * a) ;
        cout << "\n x1 = " << x1 << "\n x2 = " << x2 ;
        break;

    case 2:
        cout << "\n Repeated roots are : " ;
        x1 = -b / (2 * a) ;
        x2 = x1 ;
        cout << "\n x1 & x2 = " << x1 ;
        break;

    case 3:
        d = sqrt (abs (d)) ;
        rpart = -b / (2 * a) ; // real part of a complex root
        ipart = -b / (2 * a) ; // imaginary part of a complex root
        cout << "\n complex root are : " ;
        cout << "\n x1 = " << rpart;
        cout << "\n x2 = " << ipart;

    }
}

```

Output

```

Enter 3 numbers : 2 6 4
Real and distinct roots are :
x1 = -1.000
x2 = -2.000

```

۱.۲۴ عمل گر شرطی (The Conditional Operator)

در اکثریت موارد این عمل گر برای تصمیم گیری دوطرفه استفاده می شود. این عمل گر ترکیب علامه سوالیه؟ و کالن: می باشد و نیاز به سه قیمت (Operands) دارد (Overland, 2013).

شکل عمومی آن قرار ذیل است:

Expression1 ? Expression2 ; Expression3 ;

- شرط اول (Expression1) در قدم اول امتحان می گردد. اگر درست بود Expression2 اجرا می گردد و اگر شرط نادرست بود در آن صورت Expression3 اجرا می گردد.
- همیشه یکی از این دو شرط اجرا می گردد. امکان ندارد که هر دو شرط در عین زمان اجرا گردد.
- شما می توانید دستور if () – else را که معادل conditional operator است قرار ذیل استفاده نمایید:

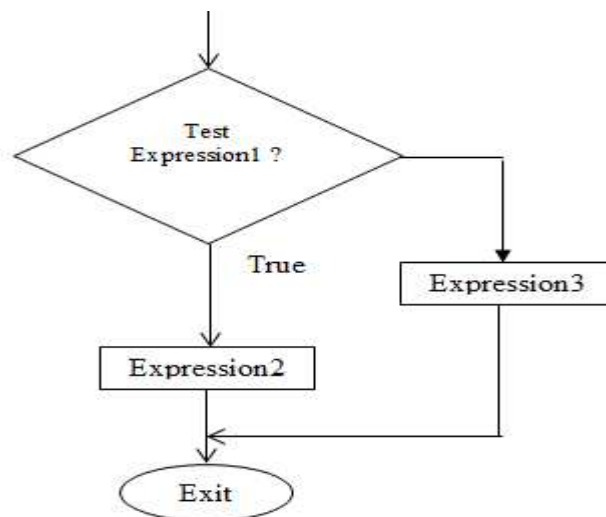
if (Expression1)

Expression 2;

else

Expression 3;

فلوچارت عمل گر شرطی (Operator Conditional) در شکل ۶.۵ نمایش داده شده است:



شکل ۶.۵ فلوچارت عمل گر Conditional

مثال 28.5: برنامه ذیل قیمت x را از کیبورد می‌گیرد و معادلات ذیل را ارزیابی می‌کند.

/* Write a program to read the value of x and evaluate the following function

$y = x + 15$ for $x > 0$

$y = 1.92x + 5$ for $x = 0$

and $y = 2x + 10$ for $x < 0$ */

// Program to show usage of $?:$ operator

#include <iostream.h>

main ()

{

float x, y ;

cout << "\n Enter a value of x :";

cin >> x ;

$y = (x > 0) ? x + 15 : ((x < 0) ? 2 * x + 10 : 1.92 * x + 5);$

cout << "\n When $x =$ " << x << " $y =$ " << y ;

return 0;

}

Output

Enter a value of x : 3

When $x = 3, y = 18$

Enter a value of x : -5

When $x = -5, y = 0$

Enter a value of x : 0

When $x = 0, y = 5$

یک مغازه رخت فروشی برای مشتریان خویش به طور ذیل تخفیف را در نظر گرفته است:

تخفیف	مقدار خرید
Discount	Purchase Amount
5%	0-100
7.5%	101-200
10%	201-300
15%	بیشتر از ۳۰۰

برنامه را بنویسید که قیمت خالص را بعد از وضع تخفیف نشان دهد؟

مثال 29.5: مسأله فوق را طور ذیل برنامه ریزی می نماییم:

```
// A cloth showroom offers discount on purchase of items as follow:
#include <iostream.h>
main ( )
{
    float amount, net;
    cout << " \n Enter the purchase amount please :";
    cin >> amount ;
    if (amount <= 0)
    {
        cout << "\n Invalid amount ";
    }
    // computation of net amount
    net = (amount <= 100 ) ? amount - amount * 0.05 :
        (amount <= 200) ? amount - amount * 0.075 :
        (amount <= 300 ) ? amount - amount * 0.10 :
        amount - amount * 0.15;
    cout << "\n The amount after discount = " << net;
    return 0;
}
```

Output

Enter the purchase amount please : 110

The amount after discount = 101.75

Enter the purchase amount please : 542.9

The amount after discount = 461.47

۱.۲۵ بیانیه goto

با استفاده از این بیانیه می‌توانید هر قسمت دل‌خواه برنامه را اجرا کنید. قابل ذکر است که goto بیانیه شرطی نیست. شکل عمومی آن قرار ذیل است (عدلیار، 1381):

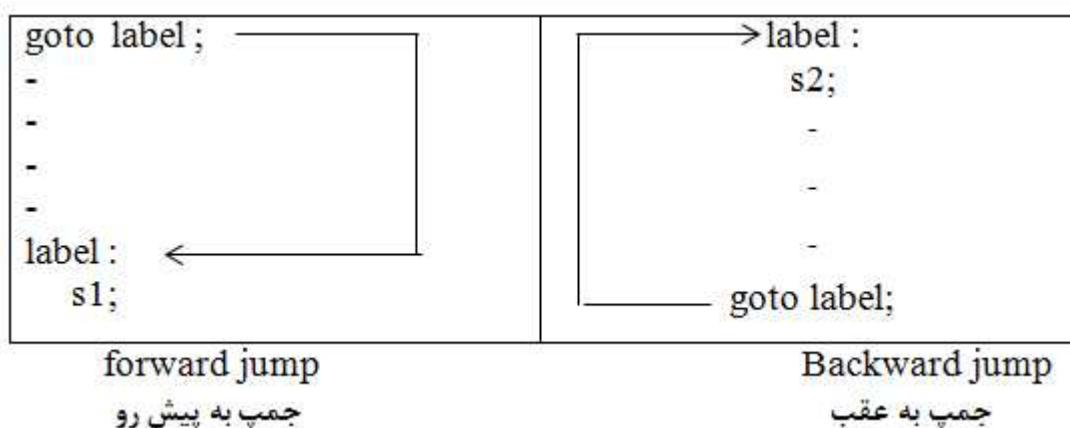
```
goto label;
```

در این جا label نام است که در آغاز سطر که کرسر باید به آن جا برود نوشته می‌گردد.

• نام لیبل باید تمام قواعد که برای تعریف نام یک متحول در نظر گرفته می‌شود مراعات کند. نام لیبل می‌تواند که حاوی حروف، ارقام و اندر سکور باشد اما اولین کرکتر آن همیشه باید حرف باشد.

• نام لیبل و به تعقیب آن دو نقطه (:) را در آغاز سطر که می‌خواهید برنامه از آن جا اجرا گردد تحریر نمایید.

• شکل عمومی goto و لیبل آن قرار ذیل است:



شکل ۷.۵ بیانیه goto

• وقتی که لیبل بعد از بیانیه goto نوشته شود آن را به نام **جمپ به پیش رو** (Jump forward) و اگر لیبل قبل از بیانیه goto معرفی شود آن را به نام **جمپ به عقب** (backward Jump) یاد می‌کند طوری که در شکل 7.5 نمایش داده شده است.

- هر لیبل در یک برنامه باید یگانه (unique) باشد. به این معنا که دو بیانیه مختلف در یک برنامه نمی‌تواند عین نام را برای لیبل داشته باشد.
- بیانیه goto را می‌توانید همراه با دستور شرطی (if) غرض تکرار برنامه (loop) استفاده کرد.
- بعضی اوقات goto سبب تکرار برنامه به شکل لایتناهی نیز می‌گردد که این حالت را به نام infinite looping یاد می‌کند.

مثال ۳۰.۵: برنامه ذیل شیوه استفاده بیانیه goto را نشان می‌دهد.

```
//Program to show usage of goto function
#include <iostream.h>
main( )
{
    float r, circum ;
    first : cout << "\n Enter Radius ( enter 0 to exit; " : (
    cin >> r;
    if ( r != 0(
    {
        circum = 2 * 3.14159 * r;
        cout << "\n Circumference = " << circum;
        goto first ;
    }
    return 0;
}
```

Output

```
Enter radius (enter 0 for exit ) : 5
Circumference = 31.416
Enter radius (enter 0 for exit ) : 0
```


مثال ۳۱.۵: برنامه ذیل باز هم شیوه استفاده بیانیه goto را نشان می‌دهد.

```
//Program to show usage of goto statement
#include <iostream.h>

main( )
{
    int a , b , big;
    cout << "\n Enter a and b; "
    cin >> a >> b;
    if ( a > b)
    {
        big = a;
        goto finish ;
    }
    if ( a <= b)
    {
        big = b;
        goto finish;
    }

    finish:
    cout << "\n Biggest of " << a << " and " << b << " is : " << big;
}
```

Output

Enter a and b : 14 38

Biggest of 14 and 38 is 38

مثال ۳۲.۵: برنامه ذیل با استفاده از `if ()` و `goto` عدد وارد شده را برعکس می‌سازد.

```
// Program to reverse a number using if ( ) and goto
#include <iostream.h>

main( )
{
    int n, rem, rev = 0;
    cout << "\n Enter a number; " :
    cin >> n;

    start:

    if ( n == 0)
    // Terminate the loop when n = 0
    goto finish;
    rem = n % 10;
    rev = rev * 10 + rem;
    n = n / 10;
    goto start ;

    finish:
    cout << "\n The reverse number is : " << rev; }
```

Output

```
Enter a number : 1234
The reverse number is 4321
Enter a number : -8329
The reverse number is -9238
```



در این فصل بیانیه‌های تصمیم‌گیری و یا شرطی مورد بحث قرار گرفت و در نتیجه مفاهیم ذیل را آموختیم:

لسان C++ بیانیه‌های شرطی دارد که در صورت نیاز می‌توانیم از آن استفاده نماییم. این بیانیه‌ها عبارت اند از: بیانیه if، بیانیه switch، بیانیه سه تایی و بیانیه goto.

بیانیه شرطی If یکی از قوی‌ترین بیانیه‌های C++ می‌باشد. این بیانیه شرط طرح شده را به شکل منطقی مورد ارزیابی قرار داده، در صورت صحت بودن آن، عمل اول که توسط برنامه‌نویس طرح گردیده اجرا و در صورت نادرست بودن، شرط عمل دوم اجرایی گردد. اگر در یک برنامه یک شرط داشته باشیم در آن صورت از () if استفاده می‌نماییم ولی اگر دو شرط داشته باشیم می‌توانیم از if - else استفاده نماییم اما اگر تعداد شرط‌ها بیش‌تر از دو باشد در آن صورت از بیانیه () if - else if - else استفاده می‌نماییم.

اگر تعداد شرط‌ها بیش‌تر باشد در آن صورت می‌توانیم از بیانیه شرطی switch نیز استفاده نماییم. به یاد داشته باشید تفاوت عمده بین if و switch در این است که switch تنها با نوعیت دیتای عددی تام (int) و یک حرفی char کار می‌کند در حالی که if می‌تواند هر نوعیت دیتا را قبول کند.

عمل‌گر سه‌تایی در اکثریت موارد برای تصمیم‌گیری دوطرفه استفاده می‌گردد. این عمل‌گر متشکل از علامه سوالیه؟ و کالن : می‌باشد و نیاز به سه قیمت (Operands) دارد.

و بالاخره، با استفاده از بیانیه goto می‌توانید هر قیمت دل‌خواهی برنامه را اجرا کنید. قابل ذکر است که goto بیانیه شرطی نیست ولی با استفاده از آن می‌توانید یک قسمت خاص برنامه را نظر به یک شرط اجرا نمایید.



سوالات و فعالیت های فصل پنجم

۱. هدف دستور if چیست؟
۲. هدف عمل گر سه گانه (ternary operator) چیست؟
۳. بیانیه nested if را توضیح دهید؟
۴. سلسله بیانیه if – else را بیان دارید؟
۵. فلوچارت if – else را بیان دارید؟
۶. تفاوت بین if و goto چیست؟
۷. تفاوت بین if – else و switch چیست؟

فعالیت ها

۱. برنامه را بنویسید که یک رقم وارد شده از کیبورد را بخواند و مترادف آن را به شکل حروفی روی سکرین نشان دهد (طور مثال، 2 وارد گردد و آن را به شیوه حروفی two نشان دهد)؟
۲. برنامه را بنویسید که دو عدد را وارد نموده و این را مشخص سازید که:
 - a. عدد اول مثبت است، منفی و یا 0
 - b. عدد دوم مثبت است، منفی و یا 0
 - c. عدد اول جفت است یا تاق
 - d. عدد دوم جفت است یا تاق
۳. برنامه را بنویسید که باقی مانده M تقسیم N را دریافت نماید (سمبول % را استفاده نکنید)؟
۴. برنامه را بنویسید که قیمت فارنهایت را بگیرد و آن را به سانتی گرید تبدیل نماید. اگر عدد 999 وارد شود برنامه ختم گردد؟

فصل ششم

حلقه‌ها

Loop Statements



هدف کلی: با بیانیه‌های حلقوی (Loop Statements) آشنا شوند.

اهداف آموزشی: در پایان این فصل محصلان قادر خواهند شد تا:

۱. مفهوم حلقه (loop) را توضیح دهند.
۲. انواع حلقه‌های زبان C++ را بیان دارند.
۳. تفاوت بین دستورهای حلقوی را شرح دهند.
۴. حلقه لایتناهی را توضیح نمایند.
۵. مفهوم break و continue را شرح دهند.

این فصل حاوی مفاهیم حلقه، انواع حلقه‌ها و اهمیت آن در برنامه نویسی مورد بحث قرار گرفته که یکی پی دیگر به تفصیل توضیح می‌گردد.

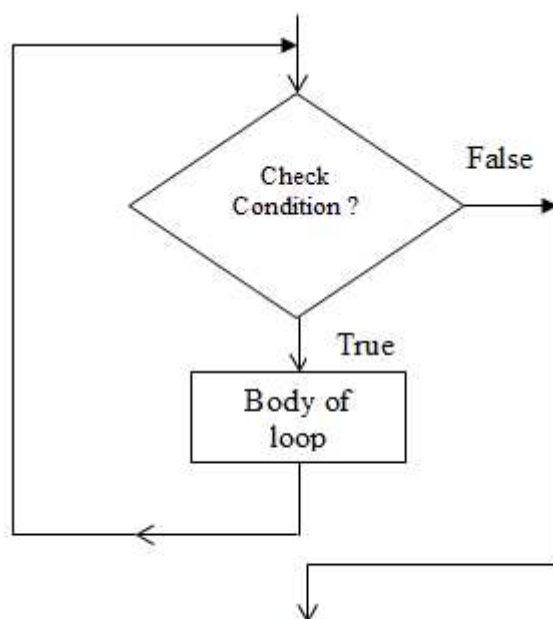
وقتی که خواسته باشیم بیانیه‌های یک برنامه را چندین بار تکرار نماییم، نیاز است تا از دستورهایی حلقوی استفاده نماییم. طوری که می‌دانیم کامپیوتر توانایی اجرای عبارت‌های مختلف یک برنامه را به شکل مکرر دارد. این قابلیت به ما اجازه می‌دهد که عبارت‌های مختلف یک برنامه را نظر به لزوم دید چندین بار تکرار نماییم. اگر کامپیوتر این خاصیت را نمی‌داشت در آن صورت مجبور بودیم که برای اجرای بعضی مسایل به هزارها سطر کد را نوشته می‌کردیم.

ساختار حلقوی (Looping Structure) عبارت از ساختار است که در آن تعداد از بیانیه‌های یک برنامه تا وقتی مکرر اجرا می‌گردد که شرط مطرح شده به دست آید. حلقه (Loop) از دو بخش تشکیل گردیده است که عبارت از بدنه حلقه (Body of the loop) و بیانیه‌های کنترل کننده (Control statements) می‌باشد.

بیانیه کنترل کننده (Control Statement) شرط را چک می‌کند که آیا برآورده شده یا خیر و مطابق به آن مجموعه از بیانیه‌ها که در بدنه همان حلقه وجود دارد اجرا می‌نماید (Horton, 2012).

ساختار کنترل کننده را می‌توان به دو بخش تقسیم کرد:

- a) Entry-controlled loops
- b) Exit-controlled loops

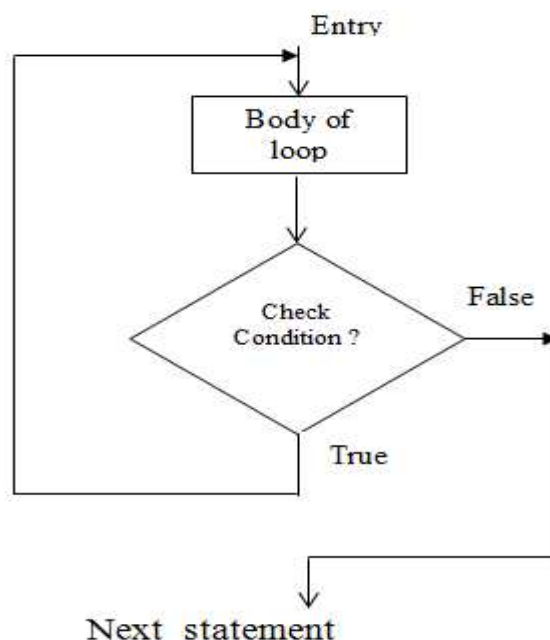


شکل ۱.۶ فلوجارت Entry- controlled loop

در حلقه‌ای که شرط اول ارزیابی گردد و بعداً مجموعه‌ای از بیانی‌ها به شکل مکرر تکرار می‌شود (Pre-test)، این نوع ساختار به نام **کنترل دخولی حلقه** (Entry – controlled Loop) یاد می‌شود. در چنین صورت بدنه حلقه وقتی اجرا می‌گردد که شرط صدق نماید در غیر آن بیانی‌های که در داخل حلقه وجود دارد، اجرا نمی‌گردد و اختیار به بیانی‌های خارج از بدنه واگذار می‌گردد. شکل فوق این موضوع را توضیح داده است. پس در حالت فوق، بدنه برنامه تا وقتی اجرا می‌گردد که شرط صدق می‌کند و به محض که شرط نادرست گردد حلقه ختم می‌گردد.

در حلقه‌ای که شرط در اخیر چک گردد این ساختار به نام **کنترل خروجی حلقه** (Exit-controlled loops) یاد می‌شود. در این حالت بدنه حلقه حداقل یک بار اجرا می‌گردد به خاطر که شرط در آخر قرار دارد و در آخر مورد ارزیابی قرار می‌گیرد (Post-test). در این حالت باز هم بدنه حلقه تا وقتی مکرراً اجرا می‌گردد که شرط صدق می‌کند و به محض که خلاف شرط عمل گردد بدنه حلقه دیگر تکرار نمی‌شود.

شکل ۲.۶ طرز کار **کنترل خروجی حلقه** را نشان می‌دهد.



فلوچارت Entry – controlled Loop

شکل ۲.۶ فلوچارت Exit – controlled loop

اگر برنامه طوری عیار شده باشد که شرط هیچ‌وقت نادرست نگردد در آن صورت برنامه به طور لایتناهی اجرا می‌گردد که این نوع حلقه را به نام حلقه لایتنهای (Infinite loop) یاد می‌کند. به این خاطر لازم است تا در وقتی تعریف یک حلقه احتیاط گردد که حلقه شکل لایتنهای را به خود نگیرد (Nakov & Koley, 2013).

به صورت عموم ساختار حلقوی حاوی مراحل ذیل می باشد:

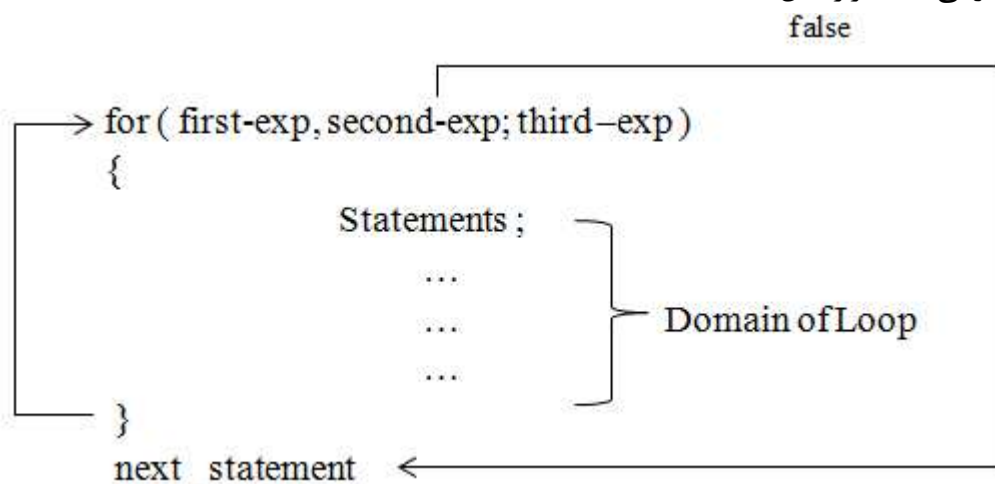
- (a) درج قیمت اولی برای محاسب (initializing a counter)
 - (b) اجرای بیانیه ها در بدنه حلقه (Executing the statements in the domain of the loop)
 - (c) ارزیابی نمودن شرط برنامه؛ به این معنا که آیا قیمت وارد شده با شرط صدق می کند یا خیر (Testing the condition in the control statement)
 - (d) اضافه شدن قیمت محاسب (counter) بعد از هر بار اجرای حلقه (Incrementing the counter)
- Chapter 22 انواع حلقه (Types of Looping Constructs)
- در لسان ++C سه نوع ساختار حلقوی وجود دارد که عبارت اند از (Glassborow, 2004):

- a) The for (; ;) Loop
- b) The while () Loop
- c) The do – while () Loop

۱.۲۵.۱ بیانیه THE FOR (; ;)

for یکی از مشهورترین حلقه های زبان ++C است که در برنامه ها بیش تر مورد استفاده قرار می گیرد. در این حلقه باز هم شرط در ابتدا مورد ارزیابی (pre-test) قرار گرفته و در صورت درست بودن آن بدنه حلقه (body of the loop) اجرا می گردد.

شکل عمومی for قرار ذیل است:



شکل ۳.۶ شکل عمومی حلقه for

۱. متحول first_exp اولین قیمت که حلقه از آن آغاز می گردد در خود ذخیره می نماید.
۲. متحول second_exp قیمت شرط حلقه است که در صورت صدق بودن آن بدنه حلقه اجرا می گردد.

۳. متحول third_exp قیمت افزودن (increment) و یا کاهش (decrement) را در خود ذخیره می‌نماید.

مثال ذیل را در نظر بگیرید:

```
for ( k = 1 ; k <= 10 ; k ++)
```

```
cout << "\n " << k;
```

در این مثال k از 1 قیمت می‌گیرد. بخش دوم یعنی $k \leq 10$ می‌باشد و در آن یک شرط مطرح گردیده که k می‌تواند الی 10 قیمت بگیرد. در بخش سوم دیده می‌شود که قیمت k بعد از هر بار اجرا به اندازه 1 اضافه می‌شود. پس گفته می‌توانیم که برنامه از 1 الی 10 قیمت می‌گیرد و آن را روی سکرین نشان می‌دهد و بعداً حلقه بسته می‌گردد.

1
2
3
10

اگر مثال فوق را طوری ذیل نوشت:

```
for ( k = 10 ; k > 0 ; k --)
```

```
cout << "\n " << k;
```

نتیجه طور ذیل خواهد بود:

10
9
8
.
.
1

۴. گذاشتن سیمی کولن (;) در بین بخش اول و دوم لازمی است.

۵. اگر قیمت کدام بخش در خود ساختار for موجود نباشد باز هم لازم است که سیمی کولن همان بخش در ساختار وجود داشته باشد.

مثال ذیل را در نظر بگیرید:

for (; I > 10 ; ++)

دیده می‌شود که بخش اول در ساختار for وجود ندارد اما سیمی کولن (;) در این بخش نوشته شده است.

۶. بدنه for می‌تواند یک یا چند بیانیه داشته باشد. اگر بیش‌تر از یک بیانیه باشد بیانیه‌ها باید داخل قوس‌های بزرگ { } نوشته شود.

۷. خوبی حلقه for نسبت به while و do-while در این است که تمام بخش‌های حلقه for در یک سطر ذکر می‌شود (بیانیه for نسبت به دیگر بیانیه‌ها ساده‌تر است).

<pre> for (i = 1; i <= 10; i ++) { Statements ; } (a) </pre>	<pre> i = 1; while (i <= 10) { Statements ; i++; } (b) </pre>	<pre> i = 1 ; do { Statements ; i++; } while (i <10) ; (c) </pre>
---	--	---

شکل ۴.۶ مقایسه ساختار هر سه حلقه

۹. عمل‌گر سیمی کالن در بیانیه (; ;) Comma Operator in for

در بیانیه (; ;) for می‌توانید چندین متحول را در عین زمان قیمت اولیه دهید مشروط بر این که بین شان عمل‌گر سیمی کالن را اضافه کنید. وقتی که کالن قیمت‌ها را جدا نماید در آن صورت اجرای آن از سمت چپ به راست مورد ارزیابی قرار می‌گیرد. بخش ذیل برنامه را مشاهده کنید:

```

x = 2 ;
for ( i = 1 ; i <= 10 ; i++)
{
    ...
    ...
    ...
    x ++ ;
}

```

شما می‌توانید مسأله فوق را با استفاده از کالن به صورت ذیل نوشت:

```

for ( x =2; i = 1; i <= 10 ; x++, i++)
{
    ...
}

```

در مثال فوق بخش اولی (Initilatzation Section) و بخش افزودی (Incrementation Section) دارای دو متحول اند.

۱۰. بخش ارزیابی حلقه for یا (test condition) نیز می‌تواند دارای چندین شرط به شکل ترکیبی باشد که یکی آن متحول داخلی حلقه و دومی آن ممکن خارجی باشد. در بخش ذیل I متحول داخل حلقه for و Sum متحول خارجی است.

```

sum = 0;

for ( i = 1; i <= 10 && sum <= 100 ; i++ )

{

    sum = sum + i;

    cout << i << sum;
}

```

۱۱. حلقه (; ;) for را می‌توانید به حیث حلقه تاخیر (delay loop) مورد استفاده قرار داد.

```

for ( x = 1; x <= 1500 ; x + + ) ;

```

این حلقه سبب تاخیر زمان می‌گردد و برای 1500 بار بدون این که هیچ بیانیه را اجرا نماید تکرار می‌شود. سیمی کولن (;) در اخیر بیانیه حلقه for به نام بیانیه خالی (Null Statement) نیز یاد می‌گردد. پس در هم‌چو حالت کمپایلر C++ این بیانیه را اشتباه نمی‌گیرد (یعنی وجود سیمی کولن در اخیر حلقه for).

مثال ۱.۶: برنامه ذیل عدد وارد شده را چک نموده که عدد اولیه است یا خیر؟

```
/* program to check whether a number is prime or not */
#include <iostream.h>
main ( )
{
    int n, i ;
    cout << " \n Enter a positive number please : " ;
    cin >>n;
    if ( n <= 1)
    {
        cout << "\n The number is not prime : " << n;
        exit(0);
    }
    for ( i = 2; i <= n/2; i++ )
    {
        if ( n % i == 0 )
        {
            cout << "\n The number is not prime : " << n;
            exit (0);
        }
    }
    // end of for ( )
    cout << "\n The number is prime : " << n;
    return 0;
} // end of main ( )
```

Output

```
Enter a positive number please : 11
The number is prime : 11
Enter a positive number please : 10
```

The number is not prime : 10

مثال 2.6: برنامه ذیل مجموعه سلسله ذیل را دریافت نموده روی سکرین نشان می دهد.

```
// Program to find the sum of the series  $1 + x + x^2 + \dots + x^n$ 
#include <iostream.h>
#include <math.h>
main ( )
{
    float i, x , sum = 1 ;
    int n ;
    cout << " \n Enter the number of terms : " ;
    cin >> n;
    cout << " Enter the value of x " ;
    cin >> x;
    for ( i = 1; i <= n ; i++ )
        sum = sum + pow (x, i) ;
    cout << "\n Summation of series = " << sum ;
    return 0;
}
```

Output

Enter the number of terms : 10

Enter the value of x : 2

Summation of series : 2047

مثال 3.6: برنامه ذیل مجموعه سلسله ذیل را دریافت می نماید.

```
// Program to print the sum of the series  $S = 1 + 1/2 + 1/3 + \dots + 1/n$ 
#include <iostream.h>
main ( )
{
    int n, i ;
    float sum = 0 ;
```

```

    cout << " \n Enter value of n : " ;
    cin >>n;
    for ( i = 1; i <= n ; i++ )
        sum = sum + 1.0 / i ;
    cout <<"\n Summation = " << sum ;
    return 0;
}

```

Output

Enter value of n : 10

Summation = 2.929

مثال ۴.۶: برنامه ذیل عدد وارد شده به برنامه را بررسی نموده که یک عدد کامل است یا خیر؟.

```

// Program to find whether a given number is a perfect number or not
#include <iostream.h>
main ( )
{
    int i , n, sum = 1;
    cout << " \n Enter a number : " ;
    cin >>n;
    for ( i = 2; i <= n / 2 ; i++ )
    {
        if ( n % i ==0 )
            sum = sum + i ;
    }
    if ( n == sum )
        cout <<"\n This is a perfect number " << n ;
    else
        cout <<"\n This is not a perfect number " << n;
    return 0;
}

```

Output

Enter a number : 6

This is a perfect number : 6

Enter a number : 5

This is not a perfect number : 5

نکات که در اثنای استفاده بیانیه (; ;) for باید در نظر گرفته شود

i. اگر قیمت آغازین (Initial value) یک حلقه نظر به قیمت آخرین آن بزرگ تر باشد در آن صورت حلقه اجرا نمی گردد.

ii. اگر در یک برنامه دو یا چند حلقه وجود داشته باشد متحول های هر دو حلقه باید نام های متفاوت داشته باشد.

iv. تعداد اجرای یک حلقه نظر به فورمول ذیل محاسبه می گردد:

$$n = \text{int} \left[\frac{\text{Upper limit} - \text{lower limit}}{\text{stepsize}} \right] + 1$$

طور مثال در حلقه ذیل:

for example in the loop

for (i = 0 ; i < 10 ; i = i + 2)

{

.....

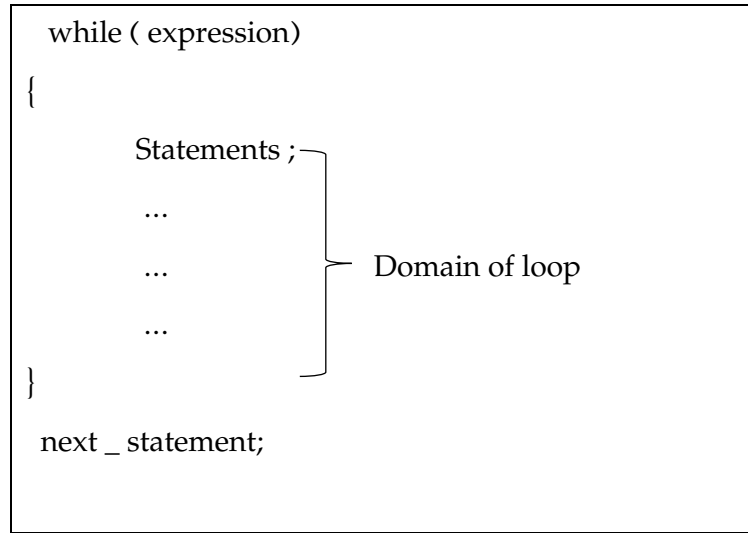
}

$$n = \left[\text{int} \frac{9 - 0}{2} \right] + 1 = 5$$

۱.۲۵.۲ بیانیه () THE WHILE

ساختار حلقه while در ابتدا شرط را مورد ارزیابی قرار می‌دهد و از جمله حلقه های Pre-test محسوب می‌گردد.

شکل عمومی آن قرار ذیل است:



شکل ۵.۶ حلقه while

بیانیه while () طور ذیل عمل می‌کند:

❖ در قدم اول شرط ارزیابی می‌شود، اگر صدق نمود بدنه حلقه اجرا می‌شود و دوباره شرط ارزیابی می‌گردد.

❖ اگر شرط باز هم صدق نمود بدنه حلقه مکرراً اجرا می‌گردد.

❖ اگر شرط صدق ننمود در آن صورت بدنه حلقه اجرا نمی‌شود و بیانیه‌های بیرون از حلقه (next statement) اجرا می‌گردد.

❖ حلقه می‌تواند در داخل خود یک یا چندین بیانیه را داشته باشد. اگر یک بیانیه باشد، نیاز نیست که داخل قوس‌بزرگ (Braces) نوشته شود اما اگر تعداد بیانیه‌ها بیش‌تر از یک باشد در آن صورت باید تمام این بیانیه‌ها در داخل یک جوهره قوس‌های بزرگ (Braces) نوشته شود.

❖ تا وقتی که شرط صدق می‌کند تمام بیانیه‌های داخل قوس که به نام بدنه حلقه (loop body) یاد می‌شود به شکل مکرر اجرا می‌گردد.

مثال ۵.۶: برنامه ذیل اعداد طبیعی ۱ الی n را با استفاده از () while روی سکرین نشان می دهد.

```
// Program to generate first N natural number
```

```
#include <iostream.h>
```

```
main ( ) {
```

```
    int i, n ;
```

```
    // initialize i to 1
```

```
    i = 1;
```

```
    cout << "\n Enter the value of n : " ;
```

```
    cin >> n;
```

```
    while ( i <= n )
```

```
{
```

```
    cout << "\n " << i ;
```

```
    i++;
```

```
}
```

```
    return 0;
```

```
}
```

Output

```
Enter the value of n: 5
```

```
1
```

```
2
```

```
3
```

```
4
```

```
5
```

در مثال فوق، اولین قیمت $i = 1$ می باشد و به تعقیب آن حلقه 5 بار اجرا می گردد. هر بار که حلقه اجرا می شود قیمت i به اندازه 1 واحد اضافه می شود. هر وقتی که قیمت i مساوی به 6 گردد شرط حالت نادرست را به خود گرفته و حلقه ختم می گردد.

مثال ۶.۶: برنامه ذیل اعداد طبیعی ۱ الی n را جمع و حاصل جمع آنرا روی سکرین نشان می دهد.

```
// program to find the sum of first n natural numbers 1 + 2 + 3 + ..... + n
#include <iostream.h>
main ( ) {
    int sum = 0, i = 1, n;
    cout << " \n Enter the value of n : " ;
    cin >> n;
    while ( i <= n )
    {
        sum = sum + i;
        i ++ ;
    }
    cout << " \n Sum = " << sum;
    return 0;
}
```

Output

Enter the value of n: 5

Sum = 15

Enter the value of n : 0

Sum = 0

مثال ۷.۶: برنامه ذیل کوچک ترین و بزرگ ترین قاسم مشترک دو عدد را با استفاده از الگوریتم Euclid's دریافت می نماید.

```
*/program to calculate the LCM and GCD of two positive integers using Euclid's
algorithm/*
#include <iostream.h>
main( )
{
    int m, n, temp, remain, lcm, gcd;
    cout << " \n Enter two integer m & n; " :
    cin >> m >> n;

    // Assign m * n to temp to retain the original value of m and n
```

```

temp = m * n;
remain = m % n;
while ( remain != 0)
{
    m = n;
    n = remain;
    remain = m % n;
}

gcd = n;
lcm = temp/ gcd;
cout << "\n LCD = " <<lcm << " GCD = " << gcd;
return 0;
}

```

Output

```

Enter two integers m & n : 8 13
LCD = 104, GCD = 1
Enter two integers m & n : 12 8
LCD = 24, GCD = 4

```

مثال ۸.۶: برنامه ذیل کوچکترین مضرب و بزرگترین قاسم مشترک بین دو عدد را نشان می‌دهد.

```

/* program to calculate the LCM and GCD of two positive integers */
#include <iostream.h>
main ( )
{
    int m, n, temp, lcm, gcd;
    cout << " \n Enter two integer m & n : " ;
    cin >> m >> n;
    // Assign m * n to tem to retain the original value of m and n
    temp = m * n ;
    // Calculation of gcd
    while ( m != n )

```

```

{
    if ( m > n)
        m = m - n ;
    else
        n = n - m ;
}

gcd = n ;
lcm = temp / gcd ;
cout << "\n LCM = " << lcm << " GCD = " << gcd ;
}

```

Output

Enter two integers m & n : 12 16

LCD = 48, GCD = 4

مثال ۹.۶: برنامه ذیل عدد وارد شده را برعکس می‌سازد.


```

/* program to reserve an integer number */
#include <iostream.h>
main ( )
{
    int num, rev, remain;
    cout << " \n Enter a number : " ;
    cin >> num;
    // initialize rev to 0
    rev = 0;
    while ( num != 0 )
    {
        remain = num % 10;
        rev = rev * 10 + remain ;
        num = num / 10 ;
    }
    cout << "\n Reserved number is : " << rev ;
}

```

Dry Run

rev = 0, num = 678



remain	rev	num
8	8	67
7	87	6
6	876	0

<pre>return 0; }</pre>	
Output Enter a number : 590 Reversed number is : 95 Enter a number : 678 Reversed number is : 876	

مثال ۱۰.۶: برنامه ذیل سلسه اعداد Fiabanocci را تهیه می نماید.

```
// program to generate the first n Fiabanocci Number
#include <iostream.h>
main (
{
    int i = 3, fib1 = 1, fib2 =1, fib, n;
    cout << " \n How many Terms ? " ;
    cin >> n;
    cout<< fib1 << "\n" << fib2 << "\n";
    while ( i < n) {
        fib = fib1 + fib2 ;
        cout << fib << "\n";
        fib1 = fib2 ;
        fib2 = fib ;
        i++;    }
    return 0; }
```

Output

How many Term ? 5

1

1

2

3

مثال ۱۱.۶: برنامه ذیل تعداد ارقام جفت و تاق را در عدد وارد شده به برنامه حساب می‌کند.

```
/* program to count the number of even and odd digits in a number */
#include <iostream.h>

main ( )
{
    int num, r, odd, even ;
    cout << " \n Enter an integer number : " ;
    cin >> num;
    odd = 0, even = 0 ;
    while ( num != 0)
    {
        r = num % 10;
        num = num / 10 ;
        if (r % 2 == 0)
            even ++;
        else
            odd++;
    }
    cout << "\n Number of Odd digits : " << odd ;
    cout << "\n Number of Even digits : " << even;
    return 0 ;
}
```

Output

Enter an integer number : 1213

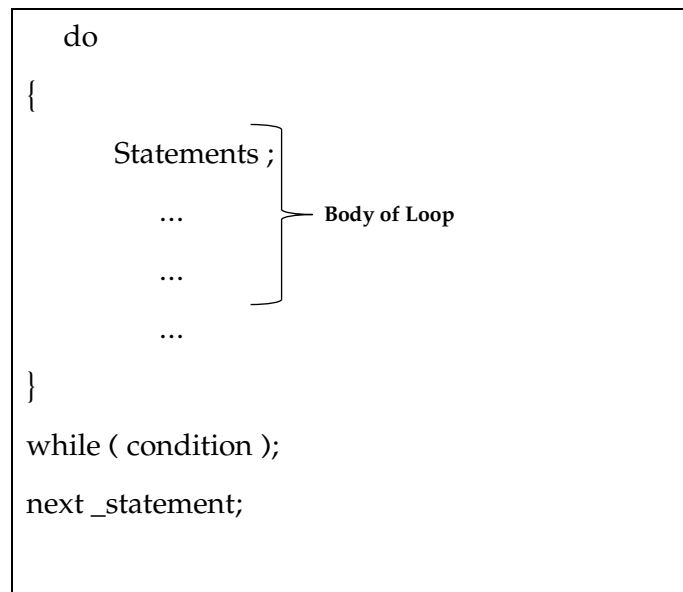
Number of Odd digits : 3

Number of Even digits : 1

۱.۲۵.۳ بیانیه () THE DO – WHILE

بیانیه do-while بعد از این که بدنه حلقه اجرا گردید شرط را ارزیابی می کند و از جمله حلقه های Post-test محسوب می گردد.

شکل عمومی آن قرار ذیل است:



شکل ۶.۶ حلقه do-while

❖ در ساختار do – while، بدنه حلقه اول اجرا می گردد.

❖ شرط در مرحله دوم ارزیابی می شود. اگر شرط صدق نماید باز هم بدنه حلقه به شکل مکرر اجرا می شود.

❖ اگر شرط صدق نکند در آن صورت بدنه حلقه به شکل مکرر اجرا نمی گردد و بیانیه های بیرون از حلقه اجرا می گردد.

مثال ۱۲.۶: برنامه ذیل قیمت وارد شده را مورد ارزیابی قرار می دهد.

```
// program to validate your input
#include <iostream.h>

main ( )
{
    int num;

    do
    {
```

```

    cout << " \n Input a positive number : " ;
    cin >> num;
}

while ( num <= 0);
return 0;
}

```

Output

```

Input a positive number : -10
Input a positive number : 0
Input a positive number : 5

```

برنامه فوق تا وقتی تکرار می شود که عدد وارد شده 0 و یا عدد منفی باشد. هر وقتی که عدد مثبت وارد گردد حلقه از بین رفته برنامه دیگر قیمت نمی گیرد.

مثال ۱۳.۶: برنامه ذیل جدول ضرب زبانی عدد وارد شده را روی سکرین نشان می دهد.

```

// program to print multiplication table of a given number
#include <iostream.h>

main ( )
{
    int i, n, prod;
    cout << "\n Enter a number : ";
    cin >> n ;
    i = 1;
    cout << "\n Multiplication Table \n\n";
    do
    {
        prod = i * n ;
        cout << "\n " << i << " * " << n << " = " << prod;
        i = i++;
    }

    while ( i <= 10);
    return 0;
}

```


}

Output

Enter a number : 4

Multiplication Table

$$1 * 4 = 4$$

$$2 * 4 = 8$$

$$3 * 4 = 12$$

$$4 * 4 = 16$$

$$5 * 4 = 20$$

$$6 * 4 = 24$$

$$7 * 4 = 28$$

$$8 * 4 = 32$$

$$9 * 4 = 36$$

$$10 * 4 = 40$$

مثال ۱۴.۶: برنامه ذیل فکتوریل عدد وارد شده را دریافت می‌نماید.

```
// program to find the factorial of a number
#include <iostream.h>
main ( )
{
    int i = 1, n;
    long fact = 1 ;
    cout << "\n Input a number please :";
    cin >> n;
    do
    {
        fact = fact * i ;
        i ++;
    }
    while ( i <= n );
    cout << "\n Factorial of " << n << " = " << fact ;
```

```
return 0;
}
```

Output

```
Input a number please : 6
Factorial of 6 = 720
Input a number please : 1
Factorial of 6 = 1
Input a number please : 0
Factorial of 6 = 1
```

مثال ۱۵.۶: برنامه ذیل مجموعه ارقام عدد وارد شده را دریافت می نماید.

```
// program to find the sum of digits of a number
#include <iostream.h>
main ( )
{
    int num, sum = 0, rem ;
    cout << "\n Enter a positive number : ";
    cin >> num;
    do
    {
        rem = num % 10 ;
        sum = sum + rem ;
        num = num / 10 ;
    }
    while ( num > 0 );

    cout << "\n Sum of the digits = " << sum ;
    return 0;
}
```

Output

```
Enter a positive number : 246
```

Sum of the digits = 12

Enter a positive number: 0

Sum of the digits = 0

مثال ۱۶.۶: برنامه ذیل بین ۱ الی n هر عدد که به ۲ و ۳ پوره قابل تقسیم اند روی سکرین نشان می دهد.

```
// program to print all the numbers divisable by 2 and 3 between 1 and n
#include <iostream.h>
main ( )
{
    int num= 1, n, rem2, rem3;
    cout <<"\n Enter value of n : ";
    cin >> n;
    do
    {
        rem2 = num % 2 ;
        rem3 = num % 3;
        if (rem2 == 0 && rem3 == 0)
            cout <<num << "\n";
        num ++ ;
    }
    while ( num <= n );
    return 0;
}
```

Output

Enter value of n : 20

6

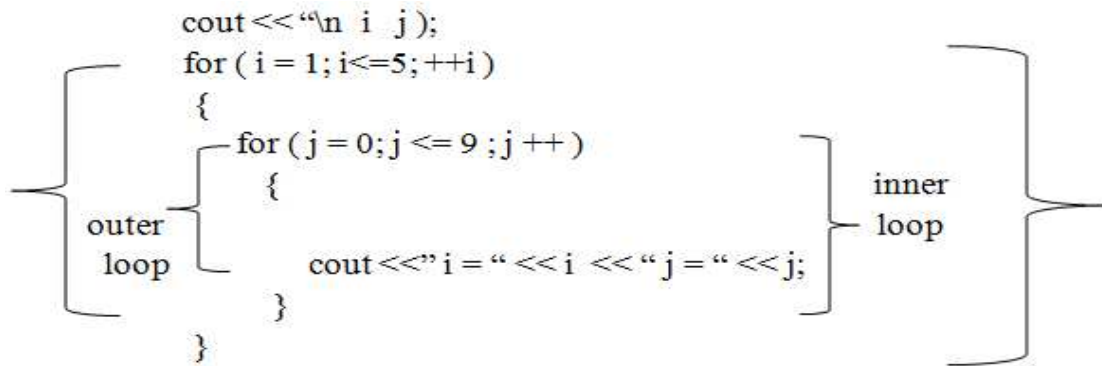
12

18

حلقه در حلقه (Nested FOR Loops)

وقتی که حلقه for در داخل یک حلقه for دیگر قرار داشته باشد این حالت را به نام Nested FOR Loop یاد می‌کند. حلقه داخلی for برای هر index حلقه خارجی باید اجرا گردد (Nawaz, 2006).

شکل 7.6 طرز کار Nested FOR Loop را نشان می‌دهد:



شکل ۷.۶ Nested for loop

❖ در قدم اول قیمت $i = 1$ در نظر گرفته شده و شرط $i \leq 5$ مورد ارزیابی قرار می‌گیرد. این که در نخست شرط صحت دارد حلقه اجرا می‌شود.

❖ در قدم دوم قیمت $j = 0$ در نظر گرفته شده و شرط $j \leq 9$ مورد ارزیابی قرار می‌گیرد. طوری که دیده می‌شود شرط صحت دارد پس بدنه حلقه داخلی اجرا می‌گردد و نتیجه ذیل را روی سکرین نشان می‌دهد:

0 1

❖ حلقه داخلی به شکل مکرر اجرا می‌شود تا وقتی که شرط صحت داشته باشد. وقتی که $j = 10$ شود در آن صورت به قیمت اولی حلقه خروجی 1 اضافه شده و قیمت آن 2 می‌گردد.

i	j
1	0
1	1
1	2
	:
1	9
2	0
2	1
	:
5	9

شما می‌توانید که تا 15 مرحله حلقه ها را داخل یک دیگر (nested loops) قرار دهید.

مثال ۱۷.۶: برنامه ذیل ضرب زبانی ۲ الی ۱۰ را با استفاده از nested if دریافت می‌نماید.

```
#include <iostream.h>

main()
{
    for (int a = 2; a<=10; a++)
    {
        for (int b=1; b<=10; b++)
        {
            cout <<b <<" * " << a <<" = " <<b *a <<"\n";
            if (b>=10)
                cout <<"\n";
        }
    }
    return 0;
}
```

Output

```
1 * 2 = 2
2 * 2 = 4
....
1 * 3 = 3
2 * 3 = 6
....
1 * 4 = 4
2 * 4 = 8
....
....
1 * 10 = 10
2 * 10 = 20
3 * 10 = 30
4 * 10 = 40
```

5 * 10 = 50
6 * 10 = 60
7 * 10 = 70
8 * 10 = 80
9 * 10 = 90
10 * 10 = 100

مثال ۱۸.۶: برنامه ذیل ساعت دیجیتال را به میان می آورد.

```
#include <iostream.h>

main()
{
    int H, M, S;
    for (H = 1; H <= 12; H++)
    {
        for (M = 0; M <= 59; M++)
        {
            for (S = 0; S <= 59; S++)
            {
                for (int MS = 0; MS < 2000; MS++) // used to slow down second speed
                    cout << H << " : " << M << " : " << S << "\n";
            }
        }
    }

    return 0;
}
```

Output

1 : 0 : 0
1 : 0 : 1
1 : 0 : 2
1 : 0 : 3
1 : 0 : 4

:
:
:
12 : 59 : 59

مثال ۱۹.۶ الف: برنامه ذیل شکل هندسی را با استفاده از اعداد ترسیم می کند.

```
// Program to generate a given pattern with numbers
#include <iostream.h>
main ( )
{
    int r, c ; // r is row and c is column
    cout << " \n \n Number Pattern : \n \n \n " ;
    for ( r = 1; r <= 4; r ++ ) //Outer loop for rows
    {
        for ( c = 1; c <= r; c++ ) // Inner loop for column
        {
            cout << c << " ";
            cout << "\n ";
        }
        return 0;
    }
}
```

Output

Number Pattern

1

1 2

1 2 3

1 2 3 4

مثال ۱۹.۶ ب: برنامه ذیل شکل هندسی مورد نظر شما را با اعداد مختلف ترسیم می کند.

```
// Program to generate a given pattern
```

```
#include <iostream.h>

main ( )
{
    int r, c, num = 6 ; // r is row and c is column
    cout << "\n \n \n Number Pattern : \n \n \n " ;
    for ( r =num ; r >= 1; r -- ) //Outer loop for rows
    {
        for ( c = 1; c <= r; c++ ) // Inner loop for column
        {
            cout << r <<" ";
            cout <<"\n " ;
        }
        return 0;
    }
}
```

Output

```
Number Pattern
6 6 6 6 6 6
5 5 5 5 5
4 4 4 4
3 3 3
2 2
1
```

مثال ۱۹.۶ ج: برنامه ذیل شکل هندسی مورد نظر را با استفاده از اعداد ترسیم می کند.

```
// Program to generate a given pattern with numbers
#include <iostream.h>

main ( )
{
    int r, c1, c2, n, m, p ; // r is row and c1, c2 are columns
    cout << " \n Enter the number of rows : " ;
    cin >> n;
    cout << "\n \n Number Pattern \n \n " ;
```



```

p = 1;
for ( r =1 ; r <= n ; r++ ) //Outer loop for rows
{
    for ( c1 = 1; c1<2*n-r; c1++ ) // Inner loop for column
        cout << " ";
    for ( c2 = 1; c2 <=r; c2++, p++)
        cout <<p << " ";
    for ( c2 = r-1; c2 >= 1; c2--, p--)
    {
        m = p-2;
        cout <<m << " ";
    }
    cout <<"\n" ;
}
// return 0; }

```

Output

Enter the number of rows : 5

Number Pattern

```

      1
    2 3 2
  3 4 5 4 3
4 5 6 7 4 5 4
5 6 7 8 9 8 7 6 5

```

مثال ۲۰.۶ د: برنامه را بنویسید که اعداد اولیه بین m و n را دریافت نماید به شرطی که قیمت m کوچک‌تر از n باشد و از نوع `int` باشد.

```

// Program to find prime numbers in the range m to n where m and n are positive
integer and m < n */
#include <iostream.h>
main ( )
{
    int m, n, i, j, isprime ;

```

```

cout << "\n Enter Lower and Upper limit of the range m & n : " ;
cin >> m >> n;
if ( m<= 1)
m = 2;          // starting value of the range is made 2
cout << "\n The prime numbers are : \n \n " ;
for ( i =m ; i <= n ; i ++ )
{
    isprime = 1;
    // This loop checks I for a prime number
    for ( j = 2; j<= i/2 ; j++ )
    {
        if ( i % j == 0 )
        {
            isprime =0 ;
            break;
        }          // End of j loop
    }
    if ( isprime )
        cout << "\n " << i ;
}          // End of i loop
return 0;
}

```

Output

Enter Lower and Upper limit of the range m & n : 5 25

The prime numbers are

5
7
11
13
17
19
23

۱.۲۶ پرش در حلقه ها (Jump in Loops)

وقتی می‌رسد که نیاز باشد تا (Stroustrup, 1997):

(a) اجرای حلقه را متوقف سازید.

(b) حلقه را از بین ببرید.

(c) یک بخش از حلقه را صرف نظر کنید.

(d) از حلقه داخلی خارج و به حلقه بیرونی وارد گردید البته نظر به یک سلسله شرایط.

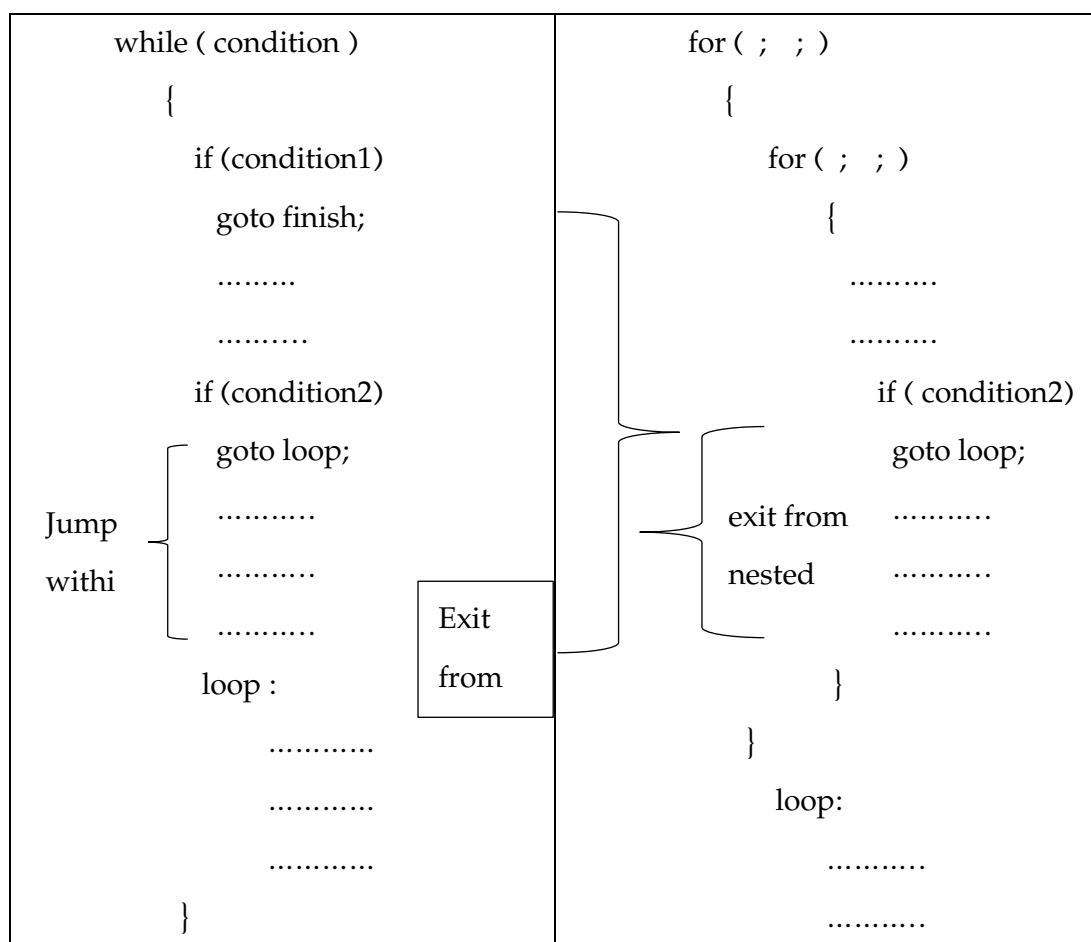
طرق مختلف وجود دارد که با استفاده از آن می‌توانید عملیات فوق را انجام دهید:

شما می‌توانید با استفاده از دستور goto، break، continue و exit عملیات فوق را انجام دهید.

۱. بیانیه goto

بیانیه goto را می‌توانید در داخل while، do-while و for برای این که اجرای یک بخش برنامه را متوقف و از یک جای دیگر آغاز نمایید استفاده نمایید. این بیانیه را می‌توانید غرض خروج از حلقه های nested نیز مورد استفاده قرار دهید (Bailey, 2007).

شکل ذیل طرز خروج از حلقه ها را با استفاده از goto نشان می‌دهد:



finish :
.....	
.....	
.....	

شکل ۸.۶ استفاده goto در داخل حلقه ها

ii. بیانیه break

این بیانیه نه تنها در switch مورد استفاده قرار می گیرد بلکه در حلقه while، do-while و for نیز از آن استفاده می توانید. break به خاطر خروج از بیانیه switch و حلقه (loop) مورد استفاده قرار می گیرد. باید گفت که break نمی تواند از حلقه nested بیرون شود و تنها می تواند حلقه که در آن وجود دارد از آن بیرون شود (Langtangen, 2006).

break ;

شکل عمومی break قرار ذیل است:

چند نمونه بیانیه break را در شکل ذیل مشاهده کنید:

while (condition)	do	for (; ;)
{	{	{
if (condition)	for (; ;)
break;	{ }
.....
exit from {	if	if
.....	(condition)	(condition)
.....	break;	break;
}	exit from loop {	exit from inner {

	}
	while	}
	(condition1);	

شکل ۹.۶ استفاده break در داخل حلقه ها

iii. بیانیه continue

بیانیه continue اجرای حلقه را ادامه داده و یا این که اجرای حلقه را متوقف نمی کند.

شکل عمومی continue قرار ذیل است:

```
continue ;
```

شما می توانید از این دستور در while، do-while و for استفاده نمایید. بیانیه goto باید در داخل بدنه حلقه (body of the loop) تعریف گردد به خاطر که بیرون از حلقه بی فایده است.

طرز استفاده continue در حلقه های مختلف طور ذیل بیان گردیده است.

<pre>while (condition1) { if (expr2) continue; }</pre>	<pre>do { if (expr1) continue; } while (condition1);</pre>	<pre>for (; ;) { if (expr1) continue; }</pre>
--	--	---

شکل ۱۰۶ استفاده continue در داخل حلقه ها

تابع () exit

این تابع به شکل فوری سبب خاتمه برنامه می گردد. شکل عمومی تابع exit() قرار ذیل است (Eckel, 2000).

```
exit ( n ) ;
```

❖ این تابع سبب بسته شدن تمام فایل های باز گردیده و قیمت n را به تابع که آن را call نموده برمی گرداند.

❖ n از حالت خروج برنامه (Exit status of program) نمایندگی می کند. پس اگر $n = 0$ باشد. این معنا را دارد که برنامه عاری از اشتباهات (Errors free) است و اگر $n \neq 0$ باشد در آن صورت این مفهوم را ارایه می کند که برنامه اشتباهات (has errors) دارد.

❖ قیمت n را می توانیم غرض شناسایی اشتباهات استفاده نماییم.

❖ قیمت default یا پیش فرض n عبارت از 0 است. به همین خاطر شکل عادی تابع exit عبارت از ; () exit می باشد.

مقایسه حلقه () while و do – while ()

جدول ۱.۶ مقایسه حلقه while و do-while

while ()	do – while ()
<p>1) شرط در اول حلقه مورد ارزیابی قرار می گیرد. این نوع حلقه به نام حلقه Entry-controlled loop یا pre-test نیز یاد می گردد.</p> <p>2) حلقه while تا وقتی اجرا می شود که شرط درست باشد.</p> <p>3) حلقه while هیچ اجرا نمی شود در صورتی که شرط نادرست باشد.</p>	<p>1) شرط در آخر حلقه مورد ارزیابی قرار می گیرد. این نوع حلقه به نام Exit-controlled loop یا post-test نیز یاد می گردد.</p> <p>2) تا وقتی که شرط درست باشد حلقه do-while اجرا می شود.</p> <p>3) حلقه do – while حد اقل یک بار اجرا می گردد در صورتی که شرط نادرست باشد.</p>



در این فصل بیانیه‌های تصمیم‌گیری و حلقه‌ها مورد بحث قرار گرفت و در نتیجه مفاهیم ذیل را آموختیم:

- ساختار حلقوی (Looping structure) عبارت از ساختار است که در آن تعداد از بیانیه‌های یک برنامه تا وقتی مکرراً اجرا می‌شود به شرط صحت داشته باشد. در زبان C++ سه نوع ساختار حلقوی وجود دارد که عبارت اند از: `while`، `do-while` و `for`.

بیانیه `while` در ابتدا شرط را مورد ارزیابی قرار می‌دهد و از جمله حلقه‌های `pre-test` محسوب می‌گردد. اما بیانیه `do-while` بعد از این که بدنه حلقه اجرا گردید شرط را ارزیابی می‌کند و از جمله حلقه‌های `post-test` یاد می‌شود.

نوع سوم بیانیه‌های حلقوی به نام `for` یاد می‌شود که در آن باز هم شرط در ابتدا مورد ارزیابی (`Pre-test`) قرار گرفته و در صورت درست بودن آن، بدنه حلقه (`Body of the loop`) اجرا می‌گردد. به یاد داشته باشید وقتی که یک حلقه در داخل یک حلقه دیگر قرار داشته باشد این حالت را به نام `Nested Loop` یاد می‌کند.

بیانیه `goto` را می‌توانید در داخل `while`، `do-while` و `for` برای این که اجرای یک بخش برنامه را متوقف و از یک جای دیگر آغاز نمایید استفاده کنید. هم‌چنان آموختیم که بیانیه `break` نه تنها در `switch` مورد استفاده قرار می‌گیرد بلکه در حلقه `while`، `do-while` و `for` نیز از آن استفاده می‌شود. `break` به خاطر خروج از بیانیه `switch` و حلقه (`loop`) مورد استفاده قرار می‌گیرد. بیانیه دیگری که به نام `continue` یاد می‌شود جهت ادامه اجرای حلقه استفاده می‌شود. و بلاخره تابع `exit` به شکل فوری سبب خاتمه برنامه می‌گردد.



سوالات و فعالیت های فصل ششم

۱. مفهوم حلقه (loop) را مختصراً توضیح دهید؟
۲. در لسان C++ چند نوع بیانیه برای حلقه وجود دارد؟
۳. هدف بیانیه while () چیست؟
۴. هدف حلقه do-while () چیست؟
۵. در کدام موارد در حلقه‌ها باید از قوس‌های جوهری بزرگ (braces) استفاده گردد؟
۶. حلقه لایتناهی (infinite loop) چیست؟
۷. تعداد دوره‌های که در یک حلقه اجرا می‌گردد چگونه محاسبه می‌شود؟
۸. مفهوم "حلقه داخل حلقه" (nested loop) چیست؟
۹. بیانیه‌های break و continue را مقایسه کنید؟

فعالیت ها

۱. تفاوت بین while ()، do-while () و for () را توضیح دهید؟
۲. طرز کار while را در یک مثال توضیح دهید؟
۳. طرز کار do-while را در یک مثال توضیح دهید؟

فصل هفتم

صف یا Array



هدف کلی: با صف یا Array آشنا شوند.

اهداف آموزشی: در پایان این فصل محصلان قادر خواهند شد تا:

۱. صف (Array) را تعریف نمایند.
۲. تفاوت بین صف و متحول عادی را بیان نمایند.
۳. انواع مختلف صف را توضیح دهند.

در این فصل مفاهیم صف، انواع صف‌ها و موارد استفاده آن مورد بحث قرار گرفته که یکی پی دیگر به تفصیل توضیح می‌گردد.

بعضی برنامه‌ها نیاز دارد تا معلومات متعدد که دارای عین نوعیت (Data type) مانند $x_1, x_2, x_3, \dots, x_n$ باشد در خود ذخیره کند. در هم‌چو حالت بهتر است که از صف یا array استفاده گردد.

صف یا array یک نوع دیتاتایپ خاص است و مجموعه عناصر که دارای عین نوعیت (Data type) باشد در خود ذخیره می‌کند.

چند خصوصیات مهم صف قرار ذیل است (Oualline, 1995):

۱. صف می‌تواند قیمت حرفی، عددی تام و اعشاری را در خود ذخیره نماید.
۲. قیمت تمام عناصر صف باید دارای عین دیتاتایپ باشد.
۳. هر عنصر انفرادی داخل یک صف به نام Subscripted Variable یاد می‌گردد.
۴. هر عنصر صف دارای یک index می‌باشد.
۵. در اثنای معرفی صف اندازه (range) آن در داخل قوس‌های متوسط (square brackets) تعیین می‌گردد.
۶. index صف همیشه از صفر (0) آغاز می‌شود.

`int marks [50];`

طور مثال:

در مثال فوق mark نام صف است که 50 موقعیت مسلسل را در حافظه کمپیوتر ریزرف می‌کند تا 50 قیمت تام در داخل آن ذخیره گردد.

هر عنصر صف را می‌توان به شکل انفرادی طور ذیل نوشت:

`marks [0], marks [1], marks [3] marks [49]`

۷. عناصر یا قیمت‌های صف همیشه به شکل مسلسل در حافظه ذخیره می‌شود.

۸. تعداد زیرنویس (subscripts) در حقیقت بُعد یک صف را تعیین می‌کند.

Chapter 23 انواع صف (Types of Array)

به صورت عموم، سه نوع صف وجود دارد که عبارت اند از: صف یک بُعدی (One Dimensional Array)، صف دو بُعدی (Two Dimensional Array) و صف چند بُعدی (Multi Dimensional Array) که هر یک آن ذیلاً توضیح می‌گردد (Aitken & Jones, 2014).

صف یک بُعدی (One Dimensional Array)

صف که عناصر آن دارای یک زیرنویس (subscript) باشد به نام صف یک بُعدی یاد می‌شود.

مثال: صف یک بُعدی ذیل را در نظر گیرید:

```
int marks [5];
```

در مثال فوق دیده می شود که یک جوهره قوس زیرنویس [] وجود دارد و این نمایان گر این است که صف بالا صف یک بُعدی است.

صف تحت نام marks طور ذیل نمرات 5 تن محصلان را در خود ذخیره می نماید:

Marks	Marks	Marks	Marks	Marks
[0]	[1]	[2]	[3]	[4]
40	38	12	44	35

قیمت های فوق را می توانیم طور ذیل در نظر گرفت:

```
Marks [ 0 ] = 40;
```

```
Marks [ 1 ] = 38;
```

```
Marks [ 2 ] = 12;
```

```
Marks [ 3 ] = 44;
```

```
Marks [ 4 ] = 35;
```

۱.۲۷ معرفی صف (Array Defination)

قبل از این که صف در برنامه استفاده گردد باید معرفی شود. صف را می توان با معرفی نوعیت، نام متحول و سائز آن معرفی کرد (Rama, 2011).

صف به شکل عمومی قرار ذیل معرفی می شود:

```
data type array _ name [ size ];
```

❖ data Type: نوعیت متحول را تعیین می کند که ممکن از نوع int, float, char و یا double باشد.

❖ array – name: نام صف است که از عین قواعد identifier پیروی می نماید.

❖ size: حد اعظمی عناصر را که در یک صف ذخیره می شود تعیین می کند.

مثال ۱.۷

```
int x [ 100 ];  
char text [ 80 ];  
float sal [ 50 ];
```

در مثال فوق صف، x تا 100 قیمت تام، test الی 80 حرف و sal الی 50 عدد اعشاری را می تواند در خود ذخیره نمایند.

متوجه باشید که:

```
int max_stud = 50 ;  
int marks [ max _ stud ] ; // not allowed
```

در مثال فوق بیانیۀ دوم درست نیست زیرا اندازه صف باید به عدد تام تعیین گردد. شما می توانید max _ stud را به شکل ثابت (Constant) طور ذیل معرفی نمایید:

```
#define MAX_STUD 50  
int marks [MAX _ STUD];
```

مثال ۲.۷: برنامه ذیل ۳ قیمت تام از کیبورد می گیرد و مجموعه آن را روی سکرین نشان می دهد.

```
/* Program to insert 3 int values from keyboard and displays sum of them */  
#include <iostream.h>  
main()  
{  
int Array [3];  
cout << "Enter 3 Integer values..?";  
cin >> Array[0];  
cin >> Array[1];  
cin >> Array[2];  
cout << "Total = " << Array [0] + Array [1] + Array [2];  
return 0;  
}
```

Output

Enter 3 Integer values..?

5

10

20

Total = 35

مثال ۳.۷: برنامه ذیل تعداد حروف انگلیسی را روی سکرین نشان می دهد.

```
#include <iostream.h>
main ( )
{
char Array [ ]= {'a','b','c','d','e'};
cout <<Array[0] <<"\n";
cout <<Array[1] <<"\n";
cout <<Array[2] <<"\n";
cout <<Array[3] <<"\n";
cout <<Array[4] <<"\n";
return 0;
}
```

Output

a

b

c

d

e

مثال ۴.۷: برنامه ذیل دیتا را به صف (array) وارد نموده و بعد آن را روی سکرین نشان می دهد.

```
/* Program to write the data into an array and print them */
#include <iostream.h>
main ( ) {
int a [ 10 ], i ;
```

```

/* Write the elements into the array */
cout << " Enter five elements " ;
for ( i = 0; i <5 ; i++)
cin >> a [i];
/* Output the data items of array a */
cout << " \n Elements of the array are : \n " ;
for ( i = 0; i < 5 ; i++)
cout << "\n " << a [ i ] ;
return 0; }

```

Output

```

Enter five elements
2
10
25
15
72
Elements of the array are :
2
10
25
15
72

```

مثال ۵.۷: برنامه ذیل مجموعه، اوسط، بزرگ‌ترین و کوچک‌ترین عدد را بین ۵ عدد وارد شده دریافت می‌نماید.

```

/* Program to find the Sum, Average, Largest and Smallest of n number */
#include <iostream.h>
main ( )
{
int a [ 10 ] , i, large, small, n ;
float avg, sum ;

```

```

/* Input elements into the array */
cout << "\n How many elements (n < 10) ? ";
cin >> n ;
cout << " Enter the elements: "<< n;
for ( i = 0; i < n ; i++)
cin >> a [i];
/* Finding sum, largest and smallest */
large = a [ 0 ] ; small = a [ 0 ] ; sum = 0 ;
for ( i = 0; i < n ; i++)
{
sum = sum + a [ i ] ;
if ( a [ i ] < small )
small = a [ i ] ;
}
/* Finding average value */
avg = sum / n ;
/* output result */
cout << "\n sum = " << sum << " Average = " << avg ;
cout << "\n largest = " << large << " Smallest = " << small ;
return 0;
}

```

Output

How many elements (n < 10) ? 5

Enter the 5 elements :

67

25

40

33

6

Sum = 171.00 Average = 34. 20

largest = 67 smallest = 6

۱.۲۸ قیمت گذاری یک صف یک بعدی (Initializing of one Dimensional Array)

عناصر صف را می توانیم در اثنای معرفی قیمت گذاری نماییم. شکل عمومی قیمت گذاری صف قرار ذیل است (Nawaz, 2006):

```
static type array_name [ size ] = { list of values } ;
```

در این جا static نوعیت ذخیره، type نوعیت دیتا، array-name نام صف و size قیمت یا اندازه اعظمی عناصر که ذخیره می شود، نشان می دهد. list of values در حقیقت لست قیمت های صف است که توسط کامه از هم دیگر جدا شده است (Nakov & Kolev, 2013).

مثال ۶.۷

The statement `static int primes [5] = { 2 , 3, 5, 7, 11 };`

Will assign the values as follows:

`primes [0] = 2, primes [1] = 3, primes [2] = 5, primes [3] = 7, and
primes [4] = 11`

Similarly `char color [] = "RED" ;`

Will assign `color [0] = "R", color [1] = 'E', color [2] = 'D' and color [3] = '\0'`

'R'	'E'	'D'	'\0'
-----	-----	-----	------

`static int num [10] = { 1, 2, 3, 4 } ;`

will initialize the first 4 array elements to 1, 2, 3, 4 and the remaining 6 must be zero.

اگر قیمت ها مطابق ترتیب فوق وارد گردد به طور ذیل نمایش داده خواهد شد:

مثال ۷.۷: اگر نمرات ۵ تن محصلان بالترتیب ۸۹، ۴۵، ۷۰، ۶۶ و ۸۱ باشد شما می توانید با استفاده از حلقه for قیمت ها را به عناصر صف طور ذیل در نظر گیرید:

consider the marks of 5 students to be 89, 45, 70, 66, and 81. A for () loop can be used to assign these values to the array elements as follow:

```
int marks [ 5 ], i ;  
cout << " \n Enter the marks: " ;  
for ( i = 0; i<5; i++ )  
{  
    cin >> marks [ i ] ;
```


دیتا که به شکل فوق وارد برنامه گردد به عین شکل چنین نمایش داده می شود:

89	45	70	66	81
----	----	----	----	----

دیتا باید همیشه به شکل حلقه (loop) به صف وارد شود به خاطر که به هر cin قیمت زیرنویس i (subscript) تغییر می کند. بار اول cin قیمت [0] marks، به تعقیب آن [1] marks و به همین طور قیمت های باقی را می خواند.

۱.۲۹ پروسسی یک صف یک بعدی (Processing of one Dimensional Array)

یک عملیه حسابی را نمی توانید بالای تمام صف اجرا نمایید. طور مثال، اگر خواسته باشید که 2 صف a و b که عین سائز را دارد باهم جمع نمایید در این صورت عملیه جمع باید بالای هر عنصر – جداگانه صورت گیرد (Overland, 2013).

مثلا:

$$c[0] = a[0] + b[0]$$

$$c[1] = a[1] + b[1]$$

به همین خاطر، 2 صف که دارای عین دیتا تایپ، عین بُعد، عین اندازه، عین عملیه، عین عمل گر مقایسوی ... باشد نیاز است که باید عنصر به عنصر اجرا گردد. برای این کار می توانیم از حلقه for (استفاده نمود که در هر بار اجرا می تواند یک عنصر – صف را پروسس نماید. تعداد اجرای حلقه و تعداد عناصر که پروسس می شود باهم مساوی خواهد بود.

مثال ۸.۷: برنامه ذیل ۳ عدد را از صفحه کلید می گیرد و آن را دوباره نشان می دهد.

```
#include <iostream.h>
main ( )
{
int Array [3];
cout << "Enter 3 Integer values..?"; // Input from KEYBOARD Scanner
cin>> Array[0];
cin>> Array[1];
cin>> Array[2];
cout << Array [0] << "\n" << Array [1] << "\n" << Array [2];
```

```
return 0;
```

```
}
```

Output

Enter 3 Integer values..? 1 2 3

1

2

3

مثال ۹.۷: برنامه ذیل عناصر دو صف یک بعدی را جمع می‌نماید.

```
/* Programs to add Two 1-D array */
#include <iostream.h>
main ( )
{
    int a [ 10 ], b [ 10 ], c [ 10 ];
    int i ;
    cout << " \n Enter 10 elements of array a : \n " ;
    for ( i = 0; i < 10 ; i++)
        cin >> a [i];
    cout << " \n Enter 10 elements of array b : \n " ;
    for ( i = 0; i < 10 ; i++)
        cin >> b [i];
    /* Finding sum of array a and b */
    for ( i = 0; i < 10 ; i++)
```

```

c [ i ] = a [ i ] + b [ i ];
// Printing the resultant array c
cout << "\n Resultant array:\n ";
for ( i = 0; i < 10 ; i++)
cout << "\t" << c [ i ];

return 0;
}

```

Dry Run

Finding sum of array a & b

i	a [i]	b [i]	c [i]
0	a [0] =1	b [0] =2	c [0] =3
1	a [1] =2	b [1] =3	c [1] =5
2	a [2] =3	b [2] =5	c [2] =8
3	a [3] =4	b [3] =7	c [3] =11
:			
:			
9	a [9] =10	b [9] =31	c [9] =41

Output

Enter 10 elements of array A

1 2 3 4 5 6 7 8 9 10

Enter 10 elements of array B

2 3 5 7 11 13 19 23 29 31

Resultant array :

3 5 8 11 16 19 26 31 38 41

مثال ۱۰.۷: برنامه ذیل n سلسله fabonacci را در یک صف تشکیل می دهد.

```

/* Programs to generate first n Fabonacci terms using an anrray */

```

```

#include <iostream.h>

```

```

main ( )
{
    int fibo [50], i, n;
    cout << " \n How many terms of the series ? " ;
    cin >>n;
    // Initialize the first and second term
    fibo [0] = 0; fibo [1] =1;
    for ( i = 2; i < n ; i++)
        fibo [ i] = fibo [ i-2 ] + fibo [ i-1 ] ;
    cout <<"\n The Fibonacci terms are " ;
    for ( i = 0; i < n ; i++)
        cout << fibo [i] << " ";
    return 0;
}

```

Output

How many terms of the series ? 10

The Fibonacci terms are :

0 1 1 2 3 5 8 13 21 34

مثال ۱۱.۷: برنامه ذیل عدد قاعده اعشاری (۱۰) را به هر قاعده دل خواه تبدیل می نماید.

```

/* Programs to convert a Decimal number to Hexa Decimal */
#include <iostream.h>
main ( )
{
    int num, rem, a[10], base, i=0, k;
    cout << " \n Enter a number: ";
    cin >>num;
    cout <<" \n Enter Base: ";
    cin >> base;
    cout <<"\n The output: ";
    while (num > 0 )

```

```

{
    rem = num % base ; // converting to base equivalent
    a [i] = rem;
    i++;
    num = num / base ;
}
i--;          // Print the converted number
for (k=1; k>=0; k--)
{
    if ( a[k] <=9 )
        cout << a[k];
    else if (a[k] ==10)
        cout << "A";
    else if (a[k] ==11)
        cout << "B";
    else if (a[k] ==12)
        cout << "C";
    else if (a[k] ==13)
        cout << "D";
    else if (a[k] ==14)
        cout << "E";
    else if (a[k] ==15)
        cout << "F";

    else
    {
        cout << "\n Error ";
        exit (0);
    }
}
return 0;

```

```
}
```

Output

Enter a number : 105

Enter Base : 16

The Output : 69

صف دو بُعدی (Two Dimensional Arrays)

صف دو بُعدی شامل سطر (row) و ستون (column) می باشد. هر یک از عناصر دو بُعدی را می توانیم با 2 زیرنویس (subscripts) حاصل کرد که زیرنویس اول از سطر و زیرنویس دوم از ستون نمایندگی می کند (Lippman, 2002).

`type array – name [row size] [col size] ;`

❖ در این جا row size و col size نشان دهنده تعداد اعظم قیمت سطر و ستون می باشد که در خود ذخیره می کند.

❖ type نشان دهنده نوعیت دیتا است.

❖ array – name نام صف را در خود ذخیره می کند.

مثال ۱۲.۷: تعریف یک صف دو بُعدی که دارای ۵ سطر و ۶ ستون می باشد.

معرفی صف 2 بُعدی

```
float mat [5][6];
```

به اندازه 120 بایت حافظه را به شکل مسلسل ریزرف می کند. به تعداد $6 * 5 = 30$ عناصر صف وجود دارد که هر عنصر به اندازه 4 بایت حافظه به شکل اعشاری غرض ذخیره شدن نیاز دارد. عناصر انفرادی عبارت اند از:

```
mat[0][0]   mat[0][1]   mat[0][2]   mat[0][3]   mat[0][4]   mat[0][5]
mat[1][0]   mat[1][1]   mat[1][2]   mat[1][3]   mat[1][4]   mat[1][5]
:
:
mat[4][0]   mat[4][1]   mat[4][2]   mat[4][3]   mat[4][4]   mat[4][5]
```

❖ قیمت صف از 0 آغاز و الی (size - 1) ادامه می یابد.

❖ عناصر صف 2 بُعدی به شکل سطر ذخیره می گردد. مثلا `int a [2][3]`

شیوه ذخیره آن قرار ذیل است :

a [0] [0]	a [0] [1]	a [0] [2]	a [1] [0]	a [1] [1]	a [1] [2]

قیمت‌گذاری صف دو بعدی (Initializing Two Dimensional Arrays)

در صف 2 بعدی قیمت‌گذاری اولیه با درمیان گذاشتن قیمت در بین قوس‌های بزرگ { } معرفی می‌گردد (Horton, 2012).

مثال:

```
int mat [3][3] = { 19, 8, 11, 25, 4, 16, 0, 8, 5 };  
mat [0] [0] = 19, mat [0] [1] = 8, mat [0] [2] = 11  
mat [1] [0] = 25, mat [1] [1] = 4, mat [1] [2] = 16  
mat [2] [0] = 0, mat [2] [1] = 8, mat [2] [2] = 5
```

می‌توانید مثال فوق را به طور مترکس به صورت ذیل نوشت:

```
int mat [3] [3] = {  
    { 19, 8, 11 },  
    { 25, 4, 16 },  
    { 0 , 8 , 5 },  
};
```

به یاد داشته باشید که بعد از ختم هر جوهره قوس کامه باید اضافه گردد. به صورت عموم، برای این که هر عنصر در یک صف دو بعدی پروسس گردد نیاز است تا از Nested loop استفاده صورت گیرد.

```
int a [ 5 ] [ 3 ], i;
```

```
for (i = 0; i < 5; i++)
```

```
for (j = 0; j < 3; j++)
```

```
    a [ i ] [ j ] = 0;
```

حلقه (Loop) اول به خاطر تغییر قیمت سطر (Row) و حلقه دوم به خاطر تغییر قیمت ستون (Column) استفاده می‌شود. شما می‌توانید عملیات ساده مانند جمع، تفریق، ضرب و بالآخره تبدیل سطر به ستون و یا ستون به سطر را با استفاده از صف 2 بعدی اجرا کنید. قابل ذکر است که سازگاری مترکس باید در نظر گرفته شود. به این معنا که تعداد سطر و ستون در بعضی موارد باید مساوی باشد در غیر آن یک سلسله عملیات فوق اجرا نمی‌گردد.

مثال ۱۳.۷: برنامه ذیل مترکس A و B را می‌خواند و مجموعه آن را بعد از محاسبه روی سکرین نشان می‌دهد.

```
// Programs to read two matrices A and B and display their sum
#include <iostream.h>

main ( )
{
    int a [ 10 ] [ 10 ], b [ 10 ] [ 10 ], sum [ 10 ] [ 10 ], i, j, n ;
    cout << " \n Enter the order of matrix : " ;
    cin >> n;
    /* Reading the elements of matrices a and b */
    cout << " \n Enter the elements of matix a : \n ";
    for ( i = 0; i < n; i++)
        for ( j = 0; j < n; j++)
            cin >> a [ i ] [ j ];

    cout << " \n Enter the elements of matix b : \n ";
    for ( i = 0; i < n; i++)
        for ( j = 0; j < n; j++)
            cin >> b [ i ] [ j ];

    // Finding the sum of the two matrices
    for ( i = 0; i < n; i++)
        for ( j = 0; j < n; j++)
            sum [ i ] [ j ] = a [ i ] [ j ] + b [ i ] [ j ];

    /* Printing the resultant matrix */
    cout << " \n The Summation matrix : \n " ;
    for ( i = 0; i < n; i++)
    {
        for ( j = 0; j < n; j++)
        {
            cout << sum [ i ] [ j ] << " ";
        }
    }
```



```

    cout << "\n" ; // generate line after each row
}
    return 0;
}

```

Output

Enter the order to matix : 3

Enter the elements of matrix A:

1 2 3 4 5 6 7 8 9

Enter the elements of matrix B:

2 4 6 8 10 12 14 16 18

The summation matrix :

3 6 9

12 15 18

21 24 27

مثال ۱۴.۷: برنامه ذیل مترکس را می‌خواند و در نتیجه سطر را به ستون و ستون را به سطر تبدیل می‌کند.

```

// Program to read a matrix and output its transpose
#include <iostream.h>
main ( )
{
    int a [ 10 ][ 10 ], trans [ 10 ][ 10 ];
    int i, j, r, c;
    cout << " \n Enter no of rows and no of columns of a matrix : " ;
    cin >> r >> c ;
    cout << "\n Enter the matix elements of a : \n";
    // Rending and assigning to trans matrix
    for (i =0; i <r; i++)
        for ( j=0; j<c; j++)
        {
            cin >> a [ i ][ j ];
            trans [ j ][ i ] = a [ i ][ j ]; // Exchange rows and column positions
        }
}

```

```

    }
    cout << "\n The given matix a : \n";
    for ( i = 0 ; i<r ; i++)
    {
        for ( j=0; j <c ; j++ )
        {
            cout << a[ i ][ j ] << " ";
        }
        cout << "\n";
    }
    cout << "\n The Transposed matrix : \n";
    for ( i=0; i <c ; i++ )
    {
        for (j=0; j<r; j++)
        {
            cout << trans [ i ][ j ] << " ";
        }
        cout<< "\n";
    } }

```

Output

Enter no of rows and no of columns of a matrix : 3 2

Enter the matrix elements of a :

1 3

5 7

9 11

The given matrix A:

1 3

5 7

9 11

The Transposed matrix :

1 5 9

3 7 11

مثال ۱۵.۷: برنامه ذیل دو مترکس را می خواند و حاصل ضرب آن را دریافت می کند.

```
// Program to read two matrices and find their product
#include <iostream.h>

main ( )
{
    int a [ 10 ] [ 10 ], b [ 10 ] [ 10 ], prod [ 10 ] [ 10 ], i , j , k , n ;
    cout << " \n Enter the order of matrix: " ;
    cin >> n;
    cout << " \n Enter the elements of matrix a : \n";
    for (i =0; i <n; i++)
        for ( j=0; j<n; j++)
            cin >> a [ i ] [ j ];

    cout << " \n Enter the elements of matrix b : \n";
    for (i =0; i <n; i++)
        for ( j=0; j<n; j++)
            cin >> b [ i ] [ j ];

    // Compute product of matrix a & b
    for ( i = 0 ; i<n ; i++)
        for ( j=0; j<n; j++)

    {
        prod [ i ] [ j ] = 0 ;
        for (k=0; k <n; k++)
            prod [ i ] [ j ] = prod [ i ] [ j ] + a [ i ] [ k ] * b [ k ] [ j ];
    }

    cout << " \n The product of a and b matrix : \n" ;
    for ( i=0; i<n; i++)
    {
        for ( j=0; j<n; j++)
```

```

{
    cout << prod [ i ] [ j ] << " ";
}
cout << "\n";
}
return 0;
}

```

Output

Enter the order of matrix : 3

Enter the elements of matrix a :

1 2 3

1 2 3

1 2 3

Enter the elements of matrix b :

2 4 6

2 4 6

2 4 6

The product of a and b matrix :

12 24 36

12 24 36

12 24 36

مثال ۱۶.۷: برنامه ذیل مجموعه اعداد مایل را در مترکس مربعی دریافت می نماید.

```

// Program to print the sum of the diagonals of a squar matrix
#include <iostream.h>
main ( )
{

```

```

int mat[ 10 ][ 10 ], i , j , n ;
int dsum1 = 0, dsum2 = 0; // sum of diagonals
cout << " \n Enter the order of matrix: " ;
cin >> n;
cout << " \n Enter the elements of matrix a : \n";
    for (i =0; i <n; i++)
        for ( j=0; j<n; j++)
        {
            cin >> mat [ i ][ j ];
        }
// To calculate sum of left to right diagonal elements
if ( i == j )
    dsum1 = dsum1 + mat [ i ][ j ];
// T o calculate sum o right to left diagonal elements
if ( i + j == n-1 )
    dsum2 = dsum2 + mat [ i ][ j ];
}
cout << " \n Sum of left to right diagonal = " << dsum1;
cout << " \n Sum of right to left diagonal = " << dsum2;
return 0;
}

```

Output

Enter the order of matrix : 3

Enter the elements of matrix :

1 5 9

2 3 7

6 8 10

Sum of left to right diagonal = 14

Sum of right to left diagonal = 18

صف چند بُعدی (Multi-Dimensional Arrays)

وقتی که یک صف بیش‌تر از 2 بُعد داشته باشد به نام صف چند بُعدی (Multidimension Array) یاد می‌شود.

شکل عمومی آن قرار ذیل است:

```
data – type array – name [ S1 ] [ S2 ] ..... [ Sn ] ;
```

در این جا s_1, s_2, \dots, s_n عبارت از اعداد تام مثبت است که نشان‌دهندهٔ تعداد عناصر در یک صف می‌باشد و هر یک آن دارای یک زیرنویس (subscript) می‌باشد. Data type نوعیت متحول و array-name نام صف می‌باشد. به همین خاطر در صف 2 بُعدی دو زیرنویس (subscripts) در صف 3 بُعدی 3 زیرنویس و در صف n بُعدی n زیرنویس وجود دارد.

مثال ۱۷.۷: سه نمونه تعریف صف چند بُعدی.

```
int lamps [ 3 ] [ 3 ] [ 4 ] ;
float tri  [ 3 ] [ 3 ] [ 3 ] ;
int table  [ 4 ] [ 4 ] [ 5 ] [ 2 ] ;
```

در مثال فوق lamps یک صف 3 بُعدی است که حاوی $36 = 4 * 3 * 3$ عنصر تام می‌باشد. tri نیز یک صف 3 بُعدی است که حاوی 27 عنصر و table یک صف 4 بُعدی است که شامل 160 عنصر از نوع تام (int) می‌باشد.

صف 2 بُعدی را می‌توانیم طور ذیل نشان دهیم:

```
int boat [ 5 ] [ 5 ] ;
```

boat [0] [0]	boat [0] [1]	boat [0] [2]	boat [0] [3]	boat [0] [4]	boat [1] [0]	boat [1] [1]	boat [1] [2]	boat [1] [3]	boat [1] [4]	boat [2] [0]	boat [2] [1]	boat [2] [2]	boat [2] [3]	boat [2] [4]	boat [3] [0]	boat [3] [1]	boat [3] [2]	boat [3] [3]	boat [3] [4]	boat [4] [0]	boat [4] [1]	boat [4] [2]	boat [4] [3]	boat [4] [4]

صف 2 بُعدی ذیل را نیز در نظر گیرید:

```
int std [ 50 ] [ 5 ] ;
```

در این مثال نمرات 50 تن محصلان در 5 مضمون نمایش داده شده است. $\text{std}[0][0]$ نمرات محصل اول که در مضمون اول به دست آورده نشان می‌دهد. $\text{std}[1][0]$ نمرات محصل اول در مضمون دوم را نشان می‌دهد و به همین‌طور الی‌آخر. $\text{std}[4][49]$ نمرات که توسط محصل 50 م در مضمون پنجم کسب نموده نشان می‌دهد.

Marks	Sub 0	Sub 1	Sub 2	Sub 3	Sub 4
Students 0					
Students 1					
Students 2					
:					
:					
Students 49					

عناصر صف در حافظه کمیوتر طور ذیل ذخیره می‌گردد.

[illegible]

همچنان صف ۳ بعدی

```
int lamps [3][3][4];
```

به طور ذیل نشان داده می‌شود:

lamps [0] [0] [0]	
lamps [0] [0] [1]	
lamps [0] [0] [2]	
lamps [0] [0] [3]	
lamps [0] [1] [0]	
lamps [0] [1] [1]	
lamps [0] [1] [2]	
lamps [0] [1] [3]	
lamps [0] [2] [0]	
lamps [0] [2] [1]	
lamps [0] [2] [2]	
lamps [0] [2] [3]	
lamps [2] [2] [0]	
	[1]
	[2]
	[3]



در این فصل صف یا Array و انواع آن مورد بحث قرار گرفت و در نتیجه مفاهیم ذیل را آموختیم:

صف یک دیتاتایپ خاص است و مجموعه‌ی عناصری که دارای عین نوعیت (Data type) باشد در خود ذخیره می‌کند.

به صورت عموم، سه نوع صف وجود دارد که عبارت اند از صف یک بُعدی (One Dimensional Array)، صف دو بُعدی (Two Dimensional Array) و صف چند بُعدی (Multi Dimensional Array) که هر یک آن به طور خلص ذیلاً بیان گردیده است:

صف که عناصر آن تنها به شکل سطر و یا ستون باشد به نام صف یک بُعدی یاد می‌گردد. اما صف که شامل سطر (row) و ستون (column) باشد به نام صف دو بُعدی یاد می‌گردد.

و بالاخره وقتی که یک صف بیش‌تر از 2 بُعد (سطر و ستون) داشته باشد به نام صف چند بُعدی (Multidimension Array) یاد می‌گردد.



سوالات و فعالیت های فصل هفتم

۱. صف یا array را تعریف نمایید؟
 ۲. تفاوت بین صف و متحول عادی چیست؟
 ۳. صف یک بعدی (one dimensional array) را تعریف نمایید؟
 ۴. یک صف را تعریف نمایید که ۱۰ عدد اعشاری در آن ذخیره گردد؟
 ۵. صف دو بعدی (two dimensional array) را تعریف نمایید و شکل عمومی آن را بیان دارید؟
 ۶. صف چند بعدی (multi dimensiuonal array) را تعریف و شکل عمومی آن را بیان دارید؟
 ۷. تفاوت بین صف یک بعدی و دو بعدی چیست، بیان دارید؟
 ۸. پروگرامی را بنویسید که ۱۰ عدد را از کیبورد بگیرد، اول اعداد تاق و بعد اعداد جفت آن را روی سکرین نشان دهد؟
 ۹. پروگرام را بنویسید که بزرگترین و کوچکترین عنصر در یک مترکس را دریافت نماید؟
 ۱۰. پروگرام را بنویسید که مجموعه عناصر سطر و مجموعه عناصر ستون را در صف 2 بعدی دریافت و روی سکرین نشان دهد؟
- ### فعالیت ها

۱. پروگرام را بنویسید که حاصل ضرب ۲ مترکس را دریافت نماید؟
۲. پروگرام را بنویسید که مثلث پاسکال را ترسیم نماید؟

1						
1			1			
1		1		1		
1		3	3	1		
1	4	6	4	1		
1	5	10	10	5	1	

فصل هشتم

تابع (Function)



هدف کلی: با تابع (Function) در برنامه‌نویسی C++ آشنا شوند.

اهداف آموزشی: در پایان این فصل محصلان قادر خواهند شد تا:

۱. تابع را تعریف نمایند.
۲. روش ایجاد کردن تابع را توضیح دهند.
۳. روش صدا یا Call نمودن تابع را تشریح نمایند.
۴. قیمت‌ها (Values) را به تابع ارسال کنند.
۵. قیمت‌ها (Values) را از تابع برگردان کنند.

این فصل حاوی مفاهیم مرتبط به تابع مانند: ایجاد تابع، استفاده از تابع، ارسال قیمت به تابع و برگرداندن قیمت‌ها از تابع می‌باشد.

توابع که تا کنون نوشته شدند فقط شامل یک تابع اصلی به نام (main) بوده اند. در برنامه‌های طولانی و پی‌چیده که شامل چندین بخش منطقی و مستقل از هم هستند، بهتر است برای هر قسمت منطقی، برنامه جداگانه نوشته شود. برنامه که برای هر یک از بخش‌ها نوشته می‌شود، تابع نام دارد. در واقع تابع برنامه است که برای حل بخشی از مسئله نوشته می‌شود.

یک تابع عبارت است از تعدادی از دستورهای برنامه که در یک واحد گرد آمده اند و یک کار مشخص را انجام می‌دهند. هر برنامه در زبان C++ حداقل دارای یک تابع بوده که همان تابع (main) است، اما یک برنامه می‌تواند شامل هر تعداد تابعی باشد. تابع (main) نقطه آغاز یک برنامه است.

در زبان برنامه نویسی C++ به صورت کلی توابع به دو دسته تقسیم می‌شوند:

۱. توابع کتابخانه‌ای

۲. توابع تعریف شده توسط کاربر

۱.۳۰ توابع کتابخانه‌ای

تعدادی از توابع که در اغلب برنامه‌ها مورد استفاده قرار می‌گیرند و کاربرد زیادی دارند، از قبل نوشته شده، به همراه کامپایلر C ارائه می‌شود. مثل توابع (sin(), cos()) که برای محاسبه زاویه به کار می‌روند و یا تابع (clrscr()) که صفحه نمایش را پاک می‌کند. این توابع را توابع کتابخانه‌ای می‌گوییم. این توابع از قبل در هر فایل موجود است که ما همواره آن‌را در ابتدای برنامه ضمیمه می‌کنیم. تنها کافی است نام تابع را نوشته و با یک دستور زبان مناسب به کار گیریم.

مثال ۱.۸: برنامه ذیل با استفاده از تابع کتابخانه‌ای <iostream.h> پیام "hello world" را نمایش می‌دهد.

```
// program to Print "Hello World!" using Library Function
#include <iostream.h>

main ( )
{
    cout << " Hello World! ";
    return 0;
}
```

Output

Hello world!

مثال ۲.۸: برنامه ذیل جذر مربع هر عدد وارد شده از صفحه کلید را با استفاده از تابع کتابخانه‌ای `<math.h>` دریافت می‌کند.

```
// program to find Square Root of any number inserted from keyboard
#include <iostream.h>
// required for sqrt function #include <math.h>
main ( )
{
    // Declaration
    double Number, SQR;
    cout << "Enter a number to find its Square Root : ";
    cin >> Number;
    SQR = sqrt (Number);
    cout << " The square root of "<< Number << " = " << SQR;
    return 0;
}
```

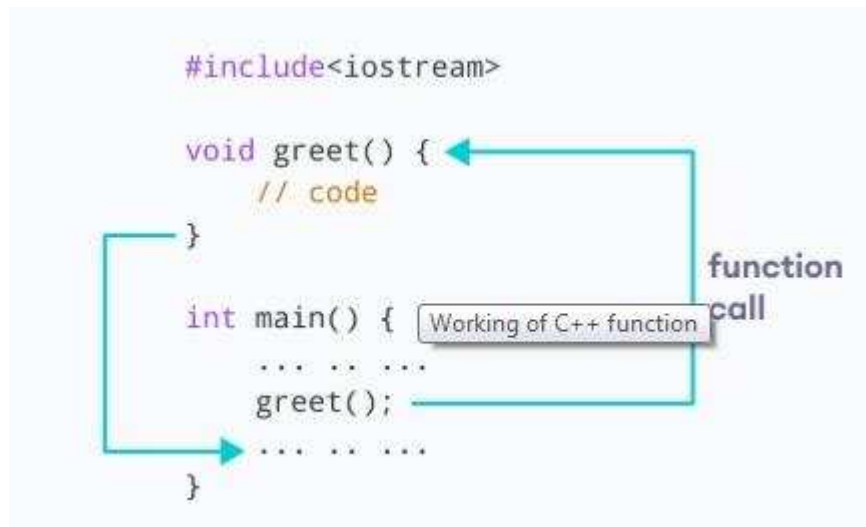
Output

Enter a number to find its Square Root : 36

The square root of 36 = 6

۱.۳۱ توابع تعریف شده توسط کاربر

تابع تعریف شده توسط کاربر، تابعی است که بدنه آن را نوشته و موقع نیاز به انجام عملیات در برنامه خود، آن را صدا کردن (call) می‌کنیم. برای ایجاد یک تابع در C++ تابع را به مثل تابع `main()` در یک قسمت برنامه معرفی می‌کنیم و بعداً آن را در بدنه تابع اصلی `main()` در قسمت دل‌خواه `call()` می‌کنیم.



شکل ۱.۸ کارکرد یک تابع

مثال ۳.۸: برنامه ذیل "Hello World" را با استفاده از تابع معرفی شده توسط کاربر نمایش می‌دهد.

```

// program to Print "Hello World!" using user defined function
#include <iostream.h>

void greet() {
    cout <<"Hello World!";
}

main ( )
{
    // calling the function
    greet();
    return 0;
}

```

Output

Hello World!

۱.۳۲ نوشتن توابع

هر تابع چهار بخش متفاوت دارد که عبارتند از:

۱. اعلان تابع (function declaration)
۲. تعریف تابع (function definition)
۳. قیمت‌های برگشتی تابع (return value)
۴. صدا کردن تابع (call function)

اعلان تابع (Function Declaration)

برای آن که کامپایلر بداند که چه توابعی در یک برنامه وجود دارد و به نوعی باشناسایی این توابع امکان استفاده از این توابع در توابع دیگر برنامه که از قبل تعریف تابع مذکور قرار دارند، از جمله داخل تابع `main()` فراهم شود باید توابع را قبل از تابع `main()` اعلان کرد.

اعلان تابع به کامپایلر در مورد نام تابع، نوعیت قیمت بازگشتی تابع و نوعیت پارامترهای تابع می‌گوید.

- **نام تابع:** هر تابع باید دارای یک نام باشد. نام گذاری توابع از قوانین نام گذاری متغیرها پیروی می‌کند. از آن جا که صدا کردن توابع توسط نام توابع صورت می‌گیرد، بهتر است نام تابع معرف عملیاتی باشد که آن تابع انجام می‌دهد، تا کد برنامه برای دیگری که آن را می‌خوانند روشن و بدون ابهام باشد. به عنوان مثال اگر تابعی عملیات جمع را انجام می‌دهد بهتر است نام آن را `add` یا `sum` انتخاب کنید.

- **نوعیت قیمت‌های بازگشتی تابع:** هر تابع باید با دریافت قیمت‌های ورودی خود، نتیجه حاصل از عملیات بر روی ورودی‌ها را به فراخواننده خود باز گرداند. هر تابع می‌تواند هیچ قیمت، یک قیمت و یا چند قیمت را بازگرداند. اما توابع که هیچ قیمت را بر نمی‌گردانند از نوع `void` می‌باشند و توابعی که یک مقدار را توسط دستور `return` باز می‌گردانند هم‌نوع با نوع بازگشتی خود می‌باشند. به عنوان مثال اگر تابعی یک عدد `int` را باز گرداند، از نوع `int` می‌باشد.

- **نوع پارامترهای تابع:** هر تابع در ریاضیات ممکن است یک یا چند ورودی داشته باشد. اما در `C++` تابع حتی می‌تواند بدون هیچ گونه قیمت ورودی باشد. چیزی که در اعلان تابع بسیار مهم است، تعیین تعداد و نوع قیمت‌های ورودی (`arguments`) تابع است. در `C++` استاندارد تأکید شده که اگر تابعی هیچ قیمت ورودی ندارد، کلمه `void` را به عنوان (`argument`) های آن تابع ذکر کنیم، تا عدم وجود (`argument`) های تابع به شکل واضح‌تر مشخص شود. شکل کلی اعلان یک تابع به صورت ذیل است:

`return_type function_name(parameter list)`

تعریف تابع (Function Definition)

این بخش بدنه اصلی تابع را شکل می‌دهد و تمام دستورات که باید یک تابع انجام دهد درین بخش معرفی می‌شود.

```
return_type function_name( parameter list ) {  
    //Function Definition  
    body of the function  
}
```

قیمت‌های برگشتی تابع (Function Return Value)

قیمت مشخصی که از یک تابع برگشت می‌کند به نام قیمت برگشتی تابع یاد می‌شود. وقتی بیانیه برگشتی اجرا شد، قیمت برگشت کننده از تابع به صدا کننده تابع برگشت می‌کند. یک تابع می‌تواند هیچ قیمتی را برگشت ندهد یا یک و چند قیمت را برگشت دهد.

توابعی که هیچ مقداری را بر نمی‌گردانند: بعضاً در برنامه از توابعی استفاده می‌کنیم که آن توابعی پس از صدا کردن عملیات مورد نظر را انجام دهند و خروجی‌های مورد انتظار را تولید و چاپ نمایند و هیچ مقداری را به تابع فراخوان تحویل نمی‌دهند.

مثال ۴.۸: برنامه ذیل تابع بدون ورودی و خروجی را نشان می‌دهد و فقط پیام را در تابع ساخته شده توسط کاربر پرنٹ می‌کند.

```
// Function that return no value  
#include <iostream.h>  
void print(void);  
int main()  
{  
    print();  
    return 0;  
}  
void print( void )  
{  
    cout << " No value Function!";  
}
```

Output

No value Function!

توابعی که یک مقدار را برمی گردانند: این توابع پس از صدا شدن یک قیمت را به تابع صدا کننده بازگشت می دهند.

مثال ۵.۸: برنامه ذیل قیمت متحول X به تابع Scope() ارسال نموده و برگشت می دهد.

```
// program to Program to return value of A from function Scope()
```

```
#include <iostream.h>
```

```
void scope(int);
```

```
int main()
```

```
{
```

```
    int x = 10;
```

```
    cout << "first value of A=" << x << "\n";
```

```
    scope(x);
```

```
    cout << "third value of A=" << x;
```

```
    return 0;
```

```
}
```

```
void scope(int a)
```

```
{
```

```
    a++;
```

```
    cout << "second value of A=" << a << "\n";
```

```
}
```

Output

first value of A= 10

second value of A= 11

third value of A= 10

مثال ۶.۸: برنامه ذیل یک عدد را دریافت نموده در تابع factorial() فکتوریل آن را دریافت نموده دوباره برگشت می دهد.

```
// Program to Find Factorial using user define Function
```

```
#include <iostream.h>
```



```

int factorial(int);

int main()
{
    int n;
    cout<<"Enter a number to find factorial: ";
    cin >> n;
    cout << "Factorial of " << n << " = " << factorial(n);
    return 0;
}

int factorial(int n)
{
    if (n > 1)
    {
        return n*factorial(n-1);
    }
    else
    {
        return 1;
    }
}

```

Output

Enter a number to find factorial: 5
 Factorial of 5 = 120

صدا کردن تابع (Function Call)

زمانی که می‌خواهیم از تابع استفاده کنیم (در برنامه اصلی) باید با استفاده از نام تابع آن را صدا کنیم. به این صورت که نام تابع مورد نظر را می‌نویسیم و در جلوی نام تابع داخل پارامتر مقادیری که می‌خواهیم برای تابع ارسال کنیم را می‌نویسیم. تا اکنون توابع زیاد کتاب‌خانه‌ای را در C++ صدا کرده‌ایم. هر تابع قادر است تابع دیگری را صدا یا call کند، به شرط آن که اعلان تابع صدا شده قبل از تابع صدا کننده موجود باشد. لذا از نقطه‌ای که اعلان یک تابع قرار دارد این تابع برای تمامی توابع دیگر برنامه قابل شناسایی و صدا شدن است. و هم‌چنان هر تابع فرعی را می‌توان از داخل تابع فرعی دیگری صدا کرد، و تنها تابع main() است که توسط توابع برنامه نمی‌توانند آن را صدا کنند.

مثال ۷.۸: برنامه ذیل کوچک‌ترین عدد بین دو عدد را در یک فنکشن معرفی شده توسط کاربر نمایش می‌دهد. (نحوه صدا کردن تابع)

```
// program to the minimum number using function
#include <iostream.h>

int minimum(int a , int b);

int main()
{
    int num1 , num2 , result;
    cout<<"Enter 2 numbers to find the minimum : "<<endl;
    cin>>num1>>num2;
    result=minimum(num1 , num2);
    cout<<"the minimum number between "<<num1<<" and "<<num2<<" is
    "<<result<<endl;
    system("pause");
    return 0;
}

int minimum(int a , int b)
{
    if(a<b)
        return a;
    if(b<a)
        return b;
    return 1;
}
```

Output

first value of A= 10

second value of A= 11

third value of A= 10

۱.۳۳ روش‌های ارسال پارامترها به توابع

پارامترها را به دو طریق می‌توان از تابع فراخوان به تابع صدا شونده ارسال کرد. این روش عبارتند از:

۱- روش صدا کردن با قیمت (call by value)

۲- روش صدا کردن با آدرس (call by reference)

صدا کردن با قیمت (Call by Value)

در این روش، یک کپی از آن پارامتر در حافظه کامپیوتر قرار می‌گیرد و تغییرات بر روی آن متغیر در تعریف تابع به هیچ‌عنوان در مقدار آن در خارج از تابع تاثیری نخواهد داشت.

مثال ۸.۸: برنامه ذیل عدد ۱۰ را مربع میکند و در خروجی نشان می‌دهد.

```
// Call by Value
#include <iostream.h>
int sqr(int x);

int main()
{
    int num = 10;
    cout << "square = " << sqr(num) << "and number = " << num;
    return 0;
}

int sqr(int x);
{
    x = x * x;
    return x;
}
```

Output

square = 100 and number = 10

مثال ۹.۸: برنامه ذیل قیمت X را بر می‌گرداند.

```
// Call by value
#include <iostream.h>

int returnEight()
```

```

{
    return 8; // return the specific value 8 back to the caller
}
int main()
{
    cout <<"return value is: "<< returnEight() << '\n';
    cout <<"return value +2 is: "<< returnEight() + 2 << '\n';
    returnEight();
    return 0;
}

```

Output

```

return value is: 8
return value +2 is:10

```

صدا کردن با آدرس (Call by reference)

در این روش، تغییرات در متغیر در آدرس موجود کاپی شده و تغییرات مربوط به پارامتر در خارج از تابع هم وابسته به تغییرات متغیر در درون تابع می باشد.

مثال ۸.۱۰: برنامه ذیل قیمت های X و Y را با هم تعویض می کند.

```

// Call by reference
#include<iostream.h>
void swap(int *x, int *y)
{
    int swap;
    swap=*x;
    *x=*y;
    *y=swap;
}
int main()
{
    int x=500, y=100;
    swap(&x, &y); // passing value to function

```

```
cout<<"Value of x is: "<<x<<endl;
cout<<"Value of y is: "<<y<<endl;
return 0;
}
```

Output

Value of x is: 100

Value of y is: 500

مثال ۱۱.۸: برنامه ذیل زمان را بر حسب ثانیه می گیرد و توسط یک تابع آن را به ساعت، دقیقه و ثانیه تبدیل نموده در اسکرین نشان می دهد.

```
// The program change the seconds from integer to Clock format
#include <iostream.h>
void convert(int s);

int main()
{
    int seconds;
    cin>> seconds;
    convert(seconds);
    return 0;
}

void convert(int s)
{
    int second,h,m;
    second = s % 60;
    h= s/3600;
    m=(s/60) % 60;
    cout<< "Time is: " << h<< ":"<< m << ":" << second;
}
```

Output

2500

0:41:40

مثال ۱۲.۸: برنامه ذیل بزرگ‌ترین عدد بین دو عدد وارد شده از کیبورد را با استفاده از یک تابع معرفی شده توسط کاربر نشان می‌دهد.

```
// The program find the maximum number between two numbers
#include <iostream.h>
int max(int num1, int num2);

int main () {
    int a;
    int b;
    int ret;
    cout<<"Enter two values for a and b \n";
    cin>>a>>b;
    ret = max(a, b);
    cout << "Max value is : " << ret << endl;

    return 0;
}
int max(int num1, int num2) {
    int result;
    if (num1 > num2)
        result = num1;
    else
        result = num2;
    return result;
}
```

Output

Enter two values for a and b
560 500
Max value is : 560

مثال ۱۳.۸: برنامه ذیل دو عدد را از کیبورد گرفته به تابع `sum` ارسال می کند و بعداً جمع آن ها را به تابع اصلی ارسال می کند.

```
// The program take to numbers sum them in a function and return the sum
#include <iostream>
using namespace std;
float sum(float, float);    // = float sum(float num1, float num2);

int main()
{
    float num1, num2 ,numSum;
    cout << "Enter first number :";
    cin >> num1;
    cout << "Enter second number :";
    cin >> num2;
    numSum = sum(num1, num2);
    cout << "The sum is: " << numSum;

    return 0;
}

float sum(float f1, float f2)    // float sum(float num1, float num2);
{
    float fSum = f1 + f2;
    return fSum;
}
```

Output

Enter first number: 20
Enter second number: 25
The sum is: 45



درین فصل درمورد مفهوم تابع معلومات حاصل نمودیم. در ادامه با انواع تابع در C++ و طریقه استفاده از آن معلومات حاصل کردیم. هر تابع یک برنامهٔ کوچک یا یک زیر برنامه است. انواع توابع، توابع کتابخانه‌ای و توابع تعریف شده توسط کاربر هستند. یک تابع شامل اعلان، بدنه تابع و بخش صدا کردن تابع می‌باشد. اعلان تابع و بدنه تابع بخش‌های ضروری تابع هستند و صدا کردن تابع در یک برنامه اختیاری است. برنامه‌های c++ حداقل یک تابع دارند که این تابع، تابع `main()` می‌باشد. هر تابع باید در برنامه c++ تعریف و اعلان شود.



سوالات و فعالیت های فصل هشتم

۱. تابع چیست؟ شرح دهید.
۲. بخش های ضروری در ایجاد یک تابع کدام ها اند؟
۳. در مورد اعلان تابع معلومات ارایه کنید.
۴. در مورد قیمت های برگشتی تابع معلومات ارایه کنید.
۵. چگونه تابع را صدا می نمایم؟ معلومات دهید.
۶. به چند نوع به تابع قیمت ارسال می توانیم؟ شرح دهید.

فعالیت ها

۱. پروگرامی را بنویسید که مکعب یک عدد را در یک تابع جداگانه دریافت و حاصل آن را به تابع `main()` ارسال کند.
۲. برنامه بنویسید که دو عدد را در یک تابع ضرب و در تابع دگر تقسیم نموده و نتایج را به تابع `main()` انتقال دهد.
۳. برنامه بنویسید که در یک تابع جذردوم و در تابع دیگر همان عدد را بتوان 4 بالا ببرد و نتیجه آن را به تابع اصلی انتقال دهد.

۱. عدلیار، س. ح. (۱۳۸۱). لسان پروگرام نویسی پیشرفته. ++C کابل، کابل، افغانستان: پوهنتون کابل.

1. Backman, K. (2012). *Structured Programming with C++*. Trollhattan, Trollhattan, Sweden: Ventus Publishing Aps.
2. Bailey, T. (2005). *An Introduction to the C Programming Language and Software Design*. United States: Prentice-Hall.
3. Balagtas, F. T. (2006). *Introduction to Programming 1*. Pepper Pike, Ohio, United States: Ursuline .
4. Buard, B. (2005). *Beginning Programming with Java* (2nd Edition ed.). Indianapolis, Indiana, United States: Wiley Publishing .
5. Deitel, H. M. (2007). *JAVA How to Program* (7th Edition ed.). Upper Saddle River, New Jersey, United States: Pearson.
6. Deitel, P., & Deitel, H. (2012). *JAVA How to Program* (9th ed.). Boston, Massachusetts, United States: Peason.
7. Eckel, B. (1999). *Thinking in C++* (2nd Edition ed.). Upper Saddle River, New Jersey, United States: Prentice Hall.
8. Gaddis, T. (2008). *Starting out with JAVA From Control Structures through Object5* (5th ed.). Upper Saddle River, New Jersey, United States: Pearson.
9. Gemmell, M. (2002). *Introduction to Programming*. Scotland, Scotland, Scotland: Sams.
10. Gilster, R. (2001). *PC Hardware : A Beginner's Guide*. NY, New York, United States: McGraw-Hill.
11. Glassborow, F. (2004). *A Beginner's Introduction to Computer Programming* . Chichester,, Chichester,, England: John Wiley.
12. Halterman, R. L. (2018). *Fundamentals of C++ Programming*. TN, Tennessee, United States: Southern Adventist.
13. Horton, I. (2012). *Beginning C* (5th Edition ed.). Belgium: Apress.
14. Jones, B., & Aitken, P. (2014). *Sams Teach Yourself C++ in One Hour a Day* (7th Edition ed.). Indianapolis, Indiana, United States: Sams Publishing.
15. Klousen, P. (2017). *JAVA 1 Basic Syntax and Semantics*. NA: BookBone.
16. Langtangen, H. P. (2006). *Introduction to C++ Programming*. Oslo, Oslo2, Narvey: Simula.
17. Lippman, S. B. (1996). *Inside the C++ Object Model*. New York, New York, United States: Addison Wesley.
18. Lippman, S. B. (2002). *Essential C++*. MA, Massachusetts, United States: Addison Wesley.
19. Love, T. (2011). *Advanced Programming with C++*. London, UK: Murrell.
20. Meyers, A. (2002). *Introduction to Computer & Programming*. New York, New York, United States: Deluxe.
21. Meyers, S. (2005). *Effective C++* (3rd Edition ed.). Boston, Massachusetts, United States: Addison Wesley.
22. Nakov, S., & Kolev, V. (2013). *Fundamentals of Programming with C#*. Sofia, Sofia, Bulgarian: Teodor Bozhikov.
23. Nawaz, A. (2006). *Programming in C++* (3rd ed.). Peshawar, Peshawar, Pakistan: Discount Book Shop.
24. Oualline, S. (1995). *Practical C++ Programming*. MA, Massachusetts, United States: O'Reilly & Associates, Inc.

25. Overland, B. (2013). *C++ without Fear: A Beginner's Guide to Make you Feel Smart*. NA, NA: Overland.
26. Prinz, P. (2002). *A Complete Guide to Programming in C++*. Bjorkliden,, Bjorkliden,, Sweden: JONES AND BARTLETT PUBLISHERS.
27. Rama, M. A. (2011). *Programming Concepts Using C++*. Malleswaram, Bangalore, India: Subhas.
28. Soulié, J. (2007). *C++ Language Tutorial*. NA, NA: cplusplus.
29. Stroustrup, B. (1997). *The C++ Programming Language* (3rd Edition ed.). Murray Hill, New Jersey, United States: AddisonWesley.
30. Stroustrup, B. (2013). *A Tour of C++*. College Station, Texas, United States: Addison-Wesley.