اصول برنامه‌نویسی کامپیوتر

اصول برنامه‌نویسی به زبان C++

محمّد فرشی

بخش علوم کامپیوتر - دانشکدهٔ علوم ریاضی - دانشگاه یزد

۱۴۰۳-۱

زبان برنامه‌نویسی

زبان برنامه‌نویسی

- روش حل مسئله با استفاده از اعمال اساسی را بیان کردیم.
- هدف ما این است که از کامپیوتر به عنوان مجری الگوریتم‌ها استفاده کنیم.
- لازمهٔ تشریح روش برای مجری آن است که یک زبان مشترک بین ما و مجری وجود داشته باشد.
- بیان روش به زبان کامپیوتر: ممکن است اما سخت است.
- آموزش زبان انسان به کامپیوتر و استفاده از زبان انسان: یکی از شاخه‌های هوش مصنوعی اما زبان انسانی دقیق نیست و مثلاً دارای سوء تفاهم است.
- یک راه حل، استفاده از از یک زبان مصنوعی واسط (معروف به زبان برنامه‌نویسی) و استفاده از یک مترجم (معروف با کامپایلر یا مفسر) برای ترجمه از زبان برنامه‌نویسی به زبان ماشین.

زبان برنامه‌نویسی

زبان برنامه‌نویسی

- روش حل مسئله با استفاده از اعمال اساسی را بیان کردیم.
- هدف ما این است که از کامپیوتر به عنوان مجری الگوریتم‌ها استفاده کنیم.
- لازمهٔ تشریح روش برای مجری آن است که یک زبان مشترک بین ما و مجری وجود داشته باشد.
- بیان روش به زبان کامپیوتر: ممکن است اما سخت است.
- آموزش زبان انسان به کامپیوتر و استفاده از زبان انسان: یکی از شاخه‌های هوش مصنوعی اما زبان انسانی دقیق نیست و مثلاً دارای سوء تفاهم است.
- یک راه حل، استفاده از از یک زبان مصنوعی واسط (معروف به زبان برنامه‌نویسی) و استفاده از یک مترجم (معروف با کامپایلر یا مفسر) برای ترجمه از زبان برنامه‌نویسی به زبان ماشین.

زبان برنامه‌نویسی

زبان برنامه‌نویسی

- روش حل مسئله با استفاده از اعمال اساسی را بیان کردیم.
- هدف ما این است که از کامپیوتر به عنوان مجری الگوریتم‌ها استفاده کنیم.
- لازمهٔ تشریح روش برای مجری آن است که یک زبان مشترک بین ما و مجری وجود داشته باشد.
- بیان روش به زبان کامپیوتر: ممکن است اما سخت است.
- آموزش زبان انسان به کامپیوتر و استفاده از زبان انسان: یکی از شاخه‌های هوش مصنوعی اما زبان انسانی دقیق نیست و مثلاً دارای سوء تفاهم است.
- یک راه حل، استفاده از از یک زبان مصنوعی واسط (معروف به زبان برنامه‌نویسی) و استفاده از یک مترجم (معروف با کامپایلر یا مفسر) برای ترجمه از زبان برنامه‌نویسی به زبان ماشین.

زبان برنامه‌نویسی

زبان برنامه‌نویسی

- روش حل مسئله با استفاده از اعمال اساسی را بیان کردیم.
- هدف ما این است که از کامپیوتر به عنوان مجری الگوریتم‌ها استفاده کنیم.
- لازمهٔ تشریح روش برای مجری آن است که یک زبان مشترک بین ما و مجری وجود داشته باشد.
- بیان روش به زبان کامپیوتر: ممکن است اما سخت است.
- آموزش زبان انسان به کامپیوتر و استفاده از زبان انسان: یکی از شاخه‌های هوش مصنوعی اما زبان انسانی دقیق نیست و مثلاً دارای سوء تفاهم است.
- یک راه حل، استفاده از از یک زبان مصنوعی واسط (معروف به زبان برنامه‌نویسی) و استفاده از یک مترجم (معروف با کامپایلر یا مفسر) برای ترجمه از زبان برنامه‌نویسی به زبان ماشین.

زبان برنامه‌نویسی

زبان برنامه‌نویسی

- روش حل مسئله با استفاده از اعمال اساسی را بیان کردیم.
- هدف ما این است که از کامپیوتر به عنوان مجری الگوریتم‌ها استفاده کنیم.
- لازمهٔ تشریح روش برای مجری آن است که یک زبان مشترک بین ما و مجری وجود داشته باشد.
- بیان روش به زبان کامپیوتر: ممکن است اما سخت است.
- آموزش زبان انسان به کامپیوتر و استفاده از زبان انسان: یکی از شاخه‌های هوش مصنوعی اما زبان انسانی دقیق نیست و مثلاً دارای سوء تفاهم است.
- یک راه حل، استفاده از از یک زبان مصنوعی واسط (معروف به زبان برنامه‌نویسی) و استفاده از یک مترجم (معروف با کامپایلر یا مفسر) برای ترجمه از زبان برنامه‌نویسی به زبان ماشین.

زبان برنامه‌نویسی

زبان برنامه‌نویسی

- روش حل مسئله با استفاده از اعمال اساسی را بیان کردیم.
- هدف ما این است که از کامپیوتر به عنوان مجری الگوریتم‌ها استفاده کنیم.
- لازمهٔ تشریح روش برای مجری آن است که یک زبان مشترک بین ما و مجری وجود داشته باشد.
- بیان روش به زبان کامپیوتر: ممکن است اما سخت است.
- آموزش زبان انسان به کامپیوتر و استفاده از زبان انسان: یکی از شاخه‌های هوش مصنوعی اما زبان انسانی دقیق نیست و مثلاً دارای سوء تفاهم است.
- یک راه حل، استفاده از از یک زبان مصنوعی واسط (معروف به زبان برنامه‌نویسی) و استفاده از یک مترجم (معروف با کامپایلر یا مفسر) برای ترجمه از زبان برنامه‌نویسی به زبان ماشین.

زبان‌های برنامه‌نویسی

زبان‌های برنامه‌نویسی:

- **ساختاری مشابه زبان‌های طبیعی دارند.**
- زبان‌ها دارای مجموعه کاراکترها، مجموعه کلمات و یک مجموعه دستور زبان یا گرامر هستند.
- با استفاده از آنها می‌توان جملات مختلف را ساخت و منظور خود را به طرف مقابل رساند.
- در زبان‌های برنامه‌نویسی، تعداد کلمات (که کلمات کلیدی زبان برنامه‌نویسی نامیده می‌شوند) و دستور زبان یا گرامر بسیار محدود می‌باشد و لذا یادگیری آن به مراتب ساده‌تر از زبانهای طبیعی نظیر انگلیسی یا آلمانی است.
- برخی زبان‌ها دارای خواص و امکاناتی است که آنها را برای استفاده در امور خاصی راحت‌تر می‌کند.
- زبان‌هایی نظیر C++ و پاسکال تقریباً همه منظوره هستند و در اکثر موارد قابل استفاده‌اند.

زبان‌های برنامه‌نویسی

زبان‌های برنامه‌نویسی:

- ساختاری مشابه زبان‌های طبیعی دارند.
- زبان‌ها دارای مجموعه کاراکترها، مجموعه کلمات و یک مجموعه دستور زبان یا گرامر هستند.
- با استفاده از آنها می‌توان جملات مختلف را ساخت و منظور خود را به طرف مقابل رساند.
- در زبان‌های برنامه‌نویسی، تعداد کلمات (که کلمات کلیدی زبان برنامه‌نویسی نامیده می‌شوند) و دستور زبان یا گرامر بسیار محدود می‌باشد و لذا یادگیری آن به مراتب ساده‌تر از زبانهای طبیعی نظیر انگلیسی یا آلمانی است.
- برخی زبان‌ها دارای خواص و امکاناتی است که آنها را برای استفاده در امور خاصی راحت‌تر می‌کند.
- زبان‌هایی نظیر C++ و پاسکال تقریباً همه منظوره هستند و در اکثر موارد قابل استفاده‌اند.

زبان‌های برنامه‌نویسی

زبان‌های برنامه‌نویسی:

- ساختاری مشابه زبان‌های طبیعی دارند.
- زبان‌ها دارای مجموعه کاراکترها، مجموعه کلمات و یک مجموعه دستور زبان یا گرامر هستند.
- با استفاده از آنها می‌توان جملات مختلف را ساخت و منظور خود را به طرف مقابل رساند.
- در زبان‌های برنامه‌نویسی، تعداد کلمات (که کلمات کلیدی زبان برنامه‌نویسی نامیده می‌شوند) و دستور زبان یا گرامر بسیار محدود می‌باشد و لذا یادگیری آن به مراتب ساده‌تر از زبانهای طبیعی نظیر انگلیسی یا آلمانی است.
- برخی زبان‌ها دارای خواص و امکاناتی است که آنها را برای استفاده در امور خاصی راحت‌تر می‌کند.
- زبان‌هایی نظیر C++ و پاسکال تقریباً همه منظوره هستند و در اکثر موارد قابل استفاده‌اند.

زبان‌های برنامه‌نویسی

زبان‌های برنامه‌نویسی:

- ساختاری مشابه زبان‌های طبیعی دارند.
- زبان‌ها دارای مجموعه کاراکترها، مجموعه کلمات و یک مجموعه دستور زبان یا گرامر هستند.
- با استفاده از آنها می‌توان جملات مختلف را ساخت و منظور خود را به طرف مقابل رساند.
- در زبان‌های برنامه‌نویسی، تعداد کلمات (که کلمات کلیدی زبان برنامه‌نویسی نامیده می‌شوند) و دستور زبان یا گرامر بسیار محدود می‌باشد و لذا یادگیری آن به مراتب ساده‌تر از زبانهای طبیعی نظیر انگلیسی یا آلمانی است.
- برخی زبان‌ها دارای خواص و امکاناتی است که آنها را برای استفاده در امور خاصی راحت‌تر می‌کند.
- زبان‌هایی نظیر C++ و پاسکال تقریباً همه منظوره هستند و در اکثر موارد قابل استفاده‌اند.

زبان‌های برنامه‌نویسی

زبان‌های برنامه‌نویسی:

- ساختاری مشابه زبان‌های طبیعی دارند.
- زبان‌ها دارای مجموعه کاراکترها، مجموعه کلمات و یک مجموعه دستور زبان یا گرامر هستند.
- با استفاده از آنها می‌توان جملات مختلف را ساخت و منظور خود را به طرف مقابل رساند.
- در زبان‌های برنامه‌نویسی، تعداد کلمات (که کلمات کلیدی زبان برنامه‌نویسی نامیده می‌شوند) و دستور زبان یا گرامر بسیار محدود می‌باشد و لذا یادگیری آن به مراتب ساده‌تر از زبانهای طبیعی نظیر انگلیسی یا آلمانی است.
- برخی زبان‌ها دارای خواص و امکاناتی است که آنها را برای استفاده در امور خاصی راحت‌تر می‌کند.
- زبان‌هایی نظیر C++ و پاسکال تقریباً همه منظوره هستند و در اکثر موارد قابل استفاده‌اند.

زبان‌های برنامه‌نویسی

زبان‌های برنامه‌نویسی:

- ساختاری مشابه زبان‌های طبیعی دارند.
- زبان‌ها دارای مجموعه کاراکترها، مجموعه کلمات و یک مجموعه دستور زبان یا گرامر هستند.
- با استفاده از آنها می‌توان جملات مختلف را ساخت و منظور خود را به طرف مقابل رساند.
- در زبان‌های برنامه‌نویسی، تعداد کلمات (که کلمات کلیدی زبان برنامه‌نویسی نامیده می‌شوند) و دستور زبان یا گرامر بسیار محدود می‌باشد و لذا یادگیری آن به مراتب ساده‌تر از زبانهای طبیعی نظیر انگلیسی یا آلمانی است.
- برخی زبان‌ها دارای خواص و امکاناتی است که آنها را برای استفاده در امور خاصی راحت‌تر می‌کند.
- زبان‌هایی نظیر C++ و پاسکال تقریباً همه منظوره هستند و در اکثر موارد قابل استفاده‌اند.

خصوصیات زبان C++

خصوصیات زبان C++:

- زبان C++ در اوایل دهۀ ۱۹۸۰ میلادی با توسعۀ زبان C ایجاد شد و به دلیل قابلیت‌های آن به طور وسیع در کاربردهای مختلف مورد استفاده قرار گرفت.
- قدرت این زبان به حدی است که به عنوان یک زبان برنامه‌نویسی سیستم استفاده شده و می‌شود، به این معنی که بسیاری از برنامه‌های کاربردی پایه‌ای نظیر سیستم‌عامل‌ها با این زبان نوشته شده‌اند.
- ارتباط نزدیکی بین C++ و زبان اسمبلی (به عنوان یک زبان سطح پایین) وجود دارد. به عبارتی، تمام امکانات زبان اسمبلی را می‌توان در C++ استفاده کرد.
- امکان توسعۀ امکانات جدید در زبان C++ بسیار بالاست و همین قابلیت باعث توسعۀ سریع ابزارها و قابلیت‌های جدید آن از طریق ارائۀ کتابخانه‌های متعدد است.
- این زبان به حروف بزرگ و کوچک حساس است. مثلاً در نام‌گذاری متغیرها، دو نام SUM و Sum با هم متفاوت هستند و یا کلمات کلیدی زبان باید دقیقاً به همان شکل بیان شده (از نظر حروف بزرگ و کوچک) استفاده شود.

خصوصیات زبان C++

خصوصیات زبان C++:

- زبان C++ در اوایل دهۀ ۱۹۸۰ میلادی با توسعۀ زبان C ایجاد شد و به دلیل قابلیت‌های آن به طور وسیع در کاربردهای مختلف مورد استفاده قرار گرفت.
- قدرت این زبان به حدی است که به عنوان یک زبان برنامه‌نویسی سیستم استفاده شده و می‌شود، به این معنی که بسیاری از برنامه‌های کاربردی پایه‌ای نظیر سیستم‌عامل‌ها با این زبان نوشته شده‌اند.
- ارتباط نزدیکی بین C++ و زبان اسمبلی (به عنوان یک زبان سطح پایین) وجود دارد. به عبارتی، تمام امکانات زبان اسمبلی را می‌توان در C++ استفاده کرد.
- امکان توسعۀ امکانات جدید در زبان C++ بسیار بالاست و همین قابلیت باعث توسعۀ سریع ابزارها و قابلیت‌های جدید آن از طریق ارائۀ کتابخانه‌های متعدد است.
- این زبان به حروف بزرگ و کوچک حساس است. مثلاً در نام‌گذاری متغیرها، دو نام SUM و Sum با هم متفاوت هستند و یا کلمات کلیدی زبان باید دقیقاً به همان شکل بیان شده (از نظر حروف بزرگ و کوچک) استفاده شود.

خصوصیات زبان C++

خصوصیات زبان C++:

- زبان C++ در اوایل دههٔ ۱۹۸۰ میلادی با توسعهٔ زبان C ایجاد شد و به دلیل قابلیت‌های آن به طور وسیع در کاربردهای مختلف مورد استفاده قرار گرفت.
- قدرت این زبان به حدی است که به عنوان یک زبان برنامه‌نویسی سیستم استفاده شده و می‌شود، به این معنی که بسیاری از برنامه‌های کاربردی پایه‌ای نظیر سیستم‌عامل‌ها با این زبان نوشته شده‌اند.
- ارتباط نزدیکی بین C++ و زبان اسمبلی (به عنوان یک زبان سطح پایین) وجود دارد. به عبارتی، تمام امکانات زبان اسمبلی را می‌توان در C++ استفاده کرد.
- امکان توسعهٔ امکانات جدید در زبان C++ بسیار بالاست و همین قابلیت باعث توسعهٔ سریع ابزارها و قابلیت‌های جدید آن از طریق ارائهٔ کتابخانه‌های متعدد است.
- این زبان به حروف بزرگ و کوچک حساس است. مثلاً در نام‌گذاری متغیرها، دو نام SUM و Sum با هم متفاوت هستند و یا کلمات کلیدی زبان باید دقیقاً به همان شکل بیان شده (از نظر حروف بزرگ و کوچک) استفاده شود.

خصوصیات زبان C++

خصوصیات زبان C++:

- زبان C++ در اوایل دهۀ ۱۹۸۰ میلادی با توسعۀ زبان C ایجاد شد و به دلیل قابلیت‌های آن به طور وسیع در کاربردهای مختلف مورد استفاده قرار گرفت.
- قدرت این زبان به حدی است که به عنوان یک زبان برنامه‌نویسی سیستم استفاده شده و می‌شود، به این معنی که بسیاری از برنامه‌های کاربردی پایه‌ای نظیر سیستم‌عامل‌ها با این زبان نوشته شده‌اند.
- ارتباط نزدیکی بین C++ و زبان اسمبلی (به عنوان یک زبان سطح پایین) وجود دارد. به عبارتی، تمام امکانات زبان اسمبلی را می‌توان در C++ استفاده کرد.
- امکان توسعۀ امکانات جدید در زبان C++ بسیار بالاست و همین قابلیت باعث توسعۀ سریع ابزارها و قابلیت‌های جدید آن از طریق ارائۀ کتابخانه‌های متعدد است.
- این زبان به حروف بزرگ و کوچک حساس است. مثلاً در نام‌گذاری متغیرها، دو نام SUM و Sum با هم متفاوت هستند و یا کلمات کلیدی زبان باید دقیقاً به همان شکل بیان شده (از نظر حروف بزرگ و کوچک) استفاده شود.

خصوصیات زبان C++

خصوصیات زبان C++:

- زبان C++ در اوایل دهۀ ۱۹۸۰ میلادی با توسعۀ زبان C ایجاد شد و به دلیل قابلیت‌های آن به طور وسیع در کاربردهای مختلف مورد استفاده قرار گرفت.
- قدرت این زبان به حدی است که به عنوان یک زبان برنامه‌نویسی سیستم استفاده شده و می‌شود، به این معنی که بسیاری از برنامه‌های کاربردی پایه‌ای نظیر سیستم‌عامل‌ها با این زبان نوشته شده‌اند.
- ارتباط نزدیکی بین C++ و زبان اسمبلی (به عنوان یک زبان سطح پایین) وجود دارد. به عبارتی، تمام امکانات زبان اسمبلی را می‌توان در C++ استفاده کرد.
- امکان توسعۀ امکانات جدید در زبان C++ بسیار بالاست و همین قابلیت باعث توسعۀ سریع ابزارها و قابلیت‌های جدید آن از طریق ارائۀ کتابخانه‌های متعدد است.
- این زبان به حروف بزرگ و کوچک حساس است. مثلاً در نام‌گذاری متغیرها، دو نام SUM و Sum با هم متفاوت هستند و یا کلمات کلیدی زبان باید دقیقاً به همان شکل بیان شده (از نظر حروف بزرگ و کوچک) استفاده شود.

ساختار یک برنامه ساده

خصوصیات زبان C++:

- نوشتن یک برنامه به یک زبان برنامه‌نویسی، مشابه نوشتن یک متن در یک زبان طبیعی است.
- در برنامه‌نویسی، به هر جمله یک دستور می‌گویند.
- هر دستور، ترجمهٔ یکی از مراحل الگوریتم به زبان برنامه‌نویسی است.
- هر دستور با ; (سمی کالن) ختم می‌شود.
- نگذاشتن سمی کالن در پایان هر دستور برنامه، باعث نادرستی دستور در زبان شده که آن را یک خطای گرامری (syntax error) می‌نامند.
- قراردادن سمی کالن اضافی نیز (در برخی جاها) خطای گرامری است و در برخی جاها باعث بروز خطای منطقی می‌شود.
- زبان C++ به فضای خالی بین کلمات حساس نیست. حتی در بین کلمات یک دستور هم می‌توان یک فاصله یا بیشتر قرار داد یا هر کلمه را در یک سطر نوشت.
- امکان قرار دادن توضیحات در برنامه وجود دارد (قراردادن // در ابتدای خط، یا محدود کردن با /* و */).

ساختار یک برنامه ساده

خصوصیات زبان C++:

- نوشتن یک برنامه به یک زبان برنامه‌نویسی، مشابه نوشتن یک متن در یک زبان طبیعی است.
- در برنامه‌نویسی، به هر جمله یک دستور می‌گویند.
- هر دستور، ترجمهٔ یکی از مراحل الگوریتم به زبان برنامه‌نویسی است.
- هر دستور با ; (سمی کالن) ختم می‌شود.
- نگذاشتن سمی کالن در پایان هر دستور برنامه، باعث نادرستی دستور در زبان شده که آن را یک خطای گرامری (syntax error) می‌نامند.
- قراردادن سمی کالن اضافی نیز (در برخی جاها) خطای گرامری است و در برخی جاها باعث بروز خطای منطقی می‌شود.
- زبان C++ به فضای خالی بین کلمات حساس نیست. حتی در بین کلمات یک دستور هم می‌توان یک فاصله یا بیشتر قرار داد یا هر کلمه را در یک سطر نوشت.
- امکان قرار دادن توضیحات در برنامه وجود دارد (قراردادن // در ابتدای خط، یا محدود کردن با /* و */).

ساختار یک برنامه ساده

خصوصیات زبان C++:

- نوشتن یک برنامه به یک زبان برنامه‌نویسی، مشابه نوشتن یک متن در یک زبان طبیعی است.
- در برنامه‌نویسی، به هر جمله یک دستور می‌گویند.
- هر دستور، ترجمهٔ یکی از مراحل الگوریتم به زبان برنامه‌نویسی است.
- هر دستور با ; (سمی کالن) ختم می‌شود.
- نگذاشتن سمی کالن در پایان هر دستور برنامه، باعث نادرستی دستور در زبان شده که آن را یک خطای گرامری (syntax error) می‌نامند.
- قراردادن سمی کالن اضافی نیز (در برخی جاها) خطای گرامری است و در برخی جاها باعث بروز خطای منطقی می‌شود.
- زبان C++ به فضای خالی بین کلمات حساس نیست. حتی در بین کلمات یک دستور هم می‌توان یک فاصله یا بیشتر قرار داد یا هر کلمه را در یک سطر نوشت.
- امکان قرار دادن توضیحات در برنامه وجود دارد (قراردادن // در ابتدای خط، یا محدود کردن با /* و */).

ساختار یک برنامه ساده

خصوصیات زبان C++:

- نوشتن یک برنامه به یک زبان برنامه‌نویسی، مشابه نوشتن یک متن در یک زبان طبیعی است.
- در برنامه‌نویسی، به هر جمله یک دستور می‌گویند.
- هر دستور، ترجمهٔ یکی از مراحل الگوریتم به زبان برنامه‌نویسی است.
- هر دستور با ; (سمی کالن) ختم می‌شود.
- نگذاشتن سمی کالن در پایان هر دستور برنامه، باعث نادرستی دستور در زبان شده که آن را یک خطای گرامری (syntax error) می‌نامند.
- قراردادن سمی کالن اضافی نیز (در برخی جاها) خطای گرامری است و در برخی جاها باعث بروز خطای منطقی می‌شود.
- زبان C++ به فضای خالی بین کلمات حساس نیست. حتی در بین کلمات یک دستور هم می‌توان یک فاصله یا بیشتر قرار داد یا هر کلمه را در یک سطر نوشت.
- امکان قرار دادن توضیحات در برنامه وجود دارد (قراردادن // در ابتدای خط، یا محدود کردن با /* و */).

ساختار یک برنامه ساده

خصوصیات زبان C++:

- نوشتن یک برنامه به یک زبان برنامه‌نویسی، مشابه نوشتن یک متن در یک زبان طبیعی است.
- در برنامه‌نویسی، به هر جمله یک **دستور** می‌گویند.
- هر دستور، ترجمهٔ یکی از مراحل الگوریتم به زبان برنامه‌نویسی است.
- هر دستور با ; (سمی کالن) ختم می‌شود.
- نگذاشتن سمی کالن در پایان هر دستورِ برنامه، باعث نادرستی دستور در زبان شده که آن را یک **خطای گرامری** (syntax error) می‌نامند.
- قراردادن سمی کالن اضافی نیز (در برخی جاها) خطای گرامری است و در برخی جاها باعث بروز خطای منطقی می‌شود.
- زبان C++ به فضای خالی بین کلمات حساس نیست. حتی در بین کلمات یک دستور هم می‌توان یک فاصله یا بیشتر قرار داد یا هر کلمه را در یک سطر نوشت.
- امکان قرار دادن توضیحات در برنامه وجود دارد (قراردادن // در ابتدای خط، یا محدود کردن با /* و */).

ساختار یک برنامه ساده

خصوصیات زبان C++:

- نوشتن یک برنامه به یک زبان برنامه‌نویسی، مشابه نوشتن یک متن در یک زبان طبیعی است.
- در برنامه‌نویسی، به هر جمله یک دستور می‌گویند.
- هر دستور، ترجمهٔ یکی از مراحل الگوریتم به زبان برنامه‌نویسی است.
- هر دستور با ; (سمی کالن) ختم می‌شود.
- نگذاشتن سمی کالن در پایان هر دستورِ برنامه، باعث نادرستی دستور در زبان شده که آن را یک خطای گرامری (syntax error) می‌نامند.
- قراردادن سمی کالن اضافی نیز (در برخی جاها) خطای گرامری است و در برخی جاها باعث بروز خطای منطقی می‌شود.
- زبان C++ به فضای خالی بین کلمات حساس نیست. حتی در بین کلمات یک دستور هم می‌توان یک فاصله یا بیشتر قرار داد یا هر کلمه را در یک سطر نوشت.
- امکان قرار دادن توضیحات در برنامه وجود دارد (قراردادن // در ابتدای خط، یا محدود کردن با /* و */).

ساختار یک برنامه ساده

خصوصیات زبان C++:

- نوشتن یک برنامه به یک زبان برنامه‌نویسی، مشابه نوشتن یک متن در یک زبان طبیعی است.
- در برنامه‌نویسی، به هر جمله یک دستور می‌گویند.
- هر دستور، ترجمهٔ یکی از مراحل الگوریتم به زبان برنامه‌نویسی است.
- هر دستور با ; (سمی کالن) ختم می‌شود.
- نگذاشتن سمی کالن در پایان هر دستورِ برنامه، باعث نادرستی دستور در زبان شده که آن را یک خطای گرامری (syntax error) می‌نامند.
- قراردادن سمی کالن اضافی نیز (در برخی جاها) خطای گرامری است و در برخی جاها باعث بروز خطای منطقی می‌شود.
- زبان C++ به فضای خالی بین کلمات حساس نیست. حتی در بین کلمات یک دستور هم می‌توان یک فاصله یا بیشتر قرار داد یا هر کلمه را در یک سطر نوشت.
- امکان قرار دادن توضیحات در برنامه وجود دارد (قراردادن // در ابتدای خط، یا محدود کردن با /* و */).

ساختار یک برنامه ساده

خصوصیات زبان C++:

- نوشتن یک برنامه به یک زبان برنامه‌نویسی، مشابه نوشتن یک متن در یک زبان طبیعی است.
- در برنامه‌نویسی، به هر جمله یک دستور می‌گویند.
- هر دستور، ترجمهٔ یکی از مراحل الگوریتم به زبان برنامه‌نویسی است.
- هر دستور با ; (سمی کالن) ختم می‌شود.
- نگذاشتن سمی کالن در پایان هر دستورِ برنامه، باعث نادرستی دستور در زبان شده که آن را یک خطای گرامری (syntax error) می‌نامند.
- قراردادن سمی کالن اضافی نیز (در برخی جاها) خطای گرامری است و در برخی جاها باعث بروز خطای منطقی می‌شود.
- زبان C++ به فضای خالی بین کلمات حساس نیست. حتی در بین کلمات یک دستور هم می‌توان یک فاصله یا بیشتر قرار داد یا هر کلمه را در یک سطر نوشت.
- امکان قرار دادن توضیحات در برنامه وجود دارد (قراردادن // در ابتدای خط، یا محدود کردن با /* و */).

انواع داده‌ها و متغیرها

انواع داده‌ها و متغیرها:

- هر زبان برنامه‌نویسی، یک سری از انواع داده‌ها را پشتیبانی می‌کند (ذخیره، محاسبات و مقایسه).
- انواع داده‌هایی که زبان C++ پشتیبانی می‌کند:

نوع	کلمه کلیدی	بازه قابل قبول/دقت	فضای حافظه (بایت)
صحیح	int , long , long int	$[-2^{31}, 2^{31}-1]$	۴
صحیح بزرگ	long long	$[-2^{63}, 2^{63}-1]$	۸
اعشاری	float	۷ رقم اعشار	۴
اعشاری با دقت مضاعف	double	۱۵ رقم اعشار	۸
اعشاری با دقت بیشتر	long double	۱۵ رقم اعشار	۸
کاراکتر	char	$[-127, 127]$	۱
دودویی	bool	true یا false	۱

- برای داده‌های صحیح و کاراکتری می‌توان از کلمات کلیدی signed یا unsigned نیز جلو آنها استفاده کرد. مقدار پیش‌فرض، signed است.

انواع داده‌ها و متغیرها

انواع داده‌ها و متغیرها:

- هر زبان برنامه‌نویسی، یک سری از انواع داده‌ها را پشتیبانی می‌کند (ذخیره، محاسبات و مقایسه).
- انواع داده‌هایی که زبان C++ پشتیبانی می‌کند:

نوع	کلمه کلیدی	بازه قابل قبول/دقت	فضای حافظه (بایت)
صحیح	int , long , long int	$[-۲^{۳۱}, ۲^{۳۱}-۱]$	۴
صحیح بزرگ	long long	$[-۲^{۶۳}, ۲^{۶۳}-۱]$	۸
اعشاری	float	۷ رقم اعشار	۴
اعشاری با دقت مضاعف	double	۱۵ رقم اعشار	۸
اعشاری با دقت بیشتر	long double	۱۵ رقم اعشار	۸
کاراکتر	char	$[-۱۲۷, ۱۲۷]$	۱
دودویی	bool	true یا false	۱

- برای داده‌های صحیح و کاراکتری می‌توان از کلمات کلیدی signed یا unsigned نیز جلو آنها استفاده کرد. مقدار پیش‌فرض، signed است.

انواع داده‌ها و متغیرها

انواع داده‌ها و متغیرها:

- هر زبان برنامه‌نویسی، یک سری از انواع داده‌ها را پشتیبانی می‌کند (ذخیره، محاسبات و مقایسه).
- انواع داده‌هایی که زبان C++ پشتیبانی می‌کند:

نوع	کلمه کلیدی	بازه قابل قبول/دقت	فضای حافظه (بایت)
صحیح	int , long , long int	$[-2^{31}, 2^{31}-1]$	۴
صحیح بزرگ	long long	$[-2^{63}, 2^{63}-1]$	۸
اعشاری	float	۷ رقم اعشار	۴
اعشاری با دقت مضاعف	double	۱۵ رقم اعشار	۸
اعشاری با دقت بیشتر	long double	۱۵ رقم اعشار	۸
کاراکتر	char	$[-127, 127]$	۱
دودویی	bool	true یا false	۱

- برای داده‌های صحیح و کاراکتری می‌توان از کلمات کلیدی signed یا unsigned نیز جلو آنها استفاده کرد. مقدار پیش‌فرض، signed است.

انواع داده‌ها و متغیرها

متغیرها:

- برای ذخیره‌سازی داده‌ها در حافظه، از متغیرها استفاده می‌کنیم.
- در زبان C++ لازم است قبل از استفاده از هر متغیر، نوع داده‌ای که قرار است در متغیر ذخیره شود، مشخص شود. این کار را اصطلاحاً تعریف متغیر می‌نامند.
- گرامر دستور تعریف متغیر:

نام متغیرها که با کاما از هم جدا شده‌اند نوع متغیر;

مثال: int A,B;

- امکان تعریف متغیرها در هر مکانی قبل از اولین استفاده از متغیر ممکن است.
- تعریف متغیر در هر قسمت از برنامه، تاثیری در روند اجرای برنامه ایجاد نمی‌کند.

قوانین نامگذاری متغیرها:

- نام متغیر، هر کلمه‌ای که کلیدی زبان نباشد.
- قانوناً ترکیبی از حروف و اعداد و کاراکتر _ باشد (دقت کنید که خط پایین یا underline است و علامت تفریق نیست).
- با یک حرف یا _ شروع شود و به عبارت دیگر با عدد شروع نشود.
- زبان C++ به حروف بزرگ و کوچک حساس است.

انواع داده‌ها و متغیرها

متغیرها:

- برای ذخیره‌سازی داده‌ها در حافظه، از متغیرها استفاده می‌کنیم.
- در زبان C++ لازم است قبل از استفاده از هر متغیر، نوع داده‌ای که قرار است در متغیر ذخیره شود، مشخص شود. این کار را اصطلاحاً تعریف متغیر می‌نامند.
- گرامر دستور تعریف متغیر:

نام متغیرها که با کاما از هم جدا شده‌اند نوع متغیر;

مثال: int A,B;

- امکان تعریف متغیرها در هر مکانی قبل از اولین استفاده از متغیر ممکن است.
- تعریف متغیر در هر قسمت از برنامه، تاثیری در روند اجرای برنامه ایجاد نمی‌کند.

قوانین نامگذاری متغیرها:

- نام متغیر هر کلمه‌ای که جزئی از زبان نباشد.
- تا حد زیادی از حروف و اعداد و کاراکتر _ باشد (دقت کنید که خط پایین یا underline است و علامت تفریق نیست).
- با یک حرف یا _ شروع شود و نه عبارت دیگر با عدد شروع نشود.
- زبان C++ به حروف بزرگ و کوچک حساس است.

انواع داده‌ها و متغیرها

متغیرها:

- برای ذخیره‌سازی داده‌ها در حافظه، از متغیرها استفاده می‌کنیم.
- در زبان C++ لازم است قبل از استفاده از هر متغیر، نوع داده‌ای که قرار است در متغیر ذخیره شود، مشخص شود. این کار را اصطلاحاً تعریف متغیر می‌نامند.
- گرامر دستور تعریف متغیر:

نام متغیرها که با کاما از هم جدا شده‌اند نوع متغیر;

مثال: int A,B;

- امکان تعریف متغیرها در هر مکانی قبل از اولین استفاده از متغیر ممکن است.
- تعریف متغیر در هر قسمت از برنامه، تاثیری در روند اجرای برنامه ایجاد نمی‌کند.

قوانین نامگذاری متغیرها:

- نام متغیر هر کلمه‌ای که جزئی از زبان نباشد.
- شامل ترکیبی از حروف و اعداد و کاراکتر _ باشد (دقت کنید که خط پایین یا underline است و علامت تفریق نیست).
- با یک حرف یا _ شروع شود و نه عبارت دیگر با عدد شروع نشود.
- زبان C++ به حروف بزرگ و کوچک حساس است.

انواع داده‌ها و متغیرها

متغیرها:

- برای ذخیره‌سازی داده‌ها در حافظه، از متغیرها استفاده می‌کنیم.
- در زبان C++ لازم است قبل از استفاده از هر متغیر، نوع داده‌ای که قرار است در متغیر ذخیره شود، مشخص شود. این کار را اصطلاحاً تعریف متغیر می‌نامند.
- گرامر دستور تعریف متغیر:

نام متغیرها که با کاما از هم جدا شده‌اند نوع متغیر;

مثال: `int A,B;`

- امکان تعریف متغیرها در هر مکانی قبل از اولین استفاده از متغیر ممکن است.
- تعریف متغیر در هر قسمت از برنامه، تاثیری در روند اجرای برنامه ایجاد نمی‌کند.

قوانین نامگذاری متغیرها:

انواع داده‌ها و متغیرها

متغیرها:

- برای ذخیره‌سازی داده‌ها در حافظه، از متغیرها استفاده می‌کنیم.
- در زبان C++ لازم است قبل از استفاده از هر متغیر، نوع داده‌ای که قرار است در متغیر ذخیره شود، مشخص شود. این کار را اصطلاحاً تعریف متغیر می‌نامند.
- گرامر دستور تعریف متغیر:

نام متغیرها که با کاما از هم جدا شده‌اند نوع متغیر;

مثال: `int A,B;`

- امکان تعریف متغیرها در هر مکانی قبل از اولین استفاده از متغیر ممکن است.
- تعریف متغیر در هر قسمت از برنامه، تاثیری در روند اجرای برنامه ایجاد نمی‌کند.

قوانین نامگذاری متغیرها:

انواع داده‌ها و متغیرها

متغیرها:

- برای ذخیره‌سازی داده‌ها در حافظه، از متغیرها استفاده می‌کنیم.
- در زبان C++ لازم است قبل از استفاده از هر متغیر، نوع داده‌ای که قرار است در متغیر ذخیره شود، مشخص شود. این کار را اصطلاحاً تعریف متغیر می‌نامند.
- گرامر دستور تعریف متغیر:

نام متغیرها که با کاما از هم جدا شده‌اند نوع متغیر;

مثال: `int A,B;`

- امکان تعریف متغیرها در هر مکانی قبل از اولین استفاده از متغیر ممکن است.
- تعریف متغیر در هر قسمت از برنامه، تاثیری در روند اجرای برنامه ایجاد نمی‌کند.

قوانین نامگذاری متغیرها:

انواع داده‌ها و متغیرها

متغیرها:

- برای ذخیره‌سازی داده‌ها در حافظه، از متغیرها استفاده می‌کنیم.
- در زبان C++ لازم است قبل از استفاده از هر متغیر، نوع داده‌ای که قرار است در متغیر ذخیره شود، مشخص شود. این کار را اصطلاحاً تعریف متغیر می‌نامند.
- گرامر دستور تعریف متغیر:

نام متغیرها که با کاما از هم جدا شده‌اند نوع متغیر;

مثال: int A,B;

- امکان تعریف متغیرها در هر مکانی قبل از اولین استفاده از متغیر ممکن است.
- تعریف متغیر در هر قسمت از برنامه، تاثیری در روند اجرای برنامه ایجاد نمی‌کند.

قوانین نامگذاری متغیرها:

- نام متغیر، جزء کلمات کلیدی زبان نباشد.
- ثانیاً ترکیبی از حروف و اعداد و کاراکتر _ باشد (دقت کنید که خط پایین یا underline است و علامت تفریق نیست).
- با یک حرف یا _ شروع شود یا به عبارت دیگر با عدد شروع نشود.
- توجه: زبان C++ به حروف بزرگ و کوچک حساس است.

انواع داده‌ها و متغیرها

متغیرها:

- برای ذخیره‌سازی داده‌ها در حافظه، از متغیرها استفاده می‌کنیم.
- در زبان C++ لازم است قبل از استفاده از هر متغیر، نوع داده‌ای که قرار است در متغیر ذخیره شود، مشخص شود. این کار را اصطلاحاً تعریف متغیر می‌نامند.
- گرامر دستور تعریف متغیر:

نام متغیرها که با کاما از هم جدا شده‌اند نوع متغیر;

مثال: int A,B;

- امکان تعریف متغیرها در هر مکانی قبل از اولین استفاده از متغیر ممکن است.
- تعریف متغیر در هر قسمت از برنامه، تاثیری در روند اجرای برنامه ایجاد نمی‌کند.

قوانین نامگذاری متغیرها:

- نام متغیر، جزء کلمات کلیدی زبان نباشد.
- ثانیاً ترکیبی از حروف و اعداد و کاراکتر _ باشد (دقت کنید که خط پایین یا underline است و علامت تفریق نیست).
- با یک حرف یا _ شروع شود یا به عبارت دیگر با عدد شروع نشود.
- توجه: زبان C++ به حروف بزرگ و کوچک حساس است.

انواع داده‌ها و متغیرها

متغیرها:

- برای ذخیره‌سازی داده‌ها در حافظه، از متغیرها استفاده می‌کنیم.
- در زبان C++ لازم است قبل از استفاده از هر متغیر، نوع داده‌ای که قرار است در متغیر ذخیره شود، مشخص شود. این کار را اصطلاحاً تعریف متغیر می‌نامند.
- گرامر دستور تعریف متغیر:

نام متغیرها که با کاما از هم جدا شده‌اند نوع متغیر;

مثال: int A,B;

- امکان تعریف متغیرها در هر مکانی قبل از اولین استفاده از متغیر ممکن است.
- تعریف متغیر در هر قسمت از برنامه، تاثیری در روند اجرای برنامه ایجاد نمی‌کند.

قوانین نامگذاری متغیرها:

- نام متغیر، جزء کلمات کلیدی زبان نباشد.
- ثانیاً ترکیبی از حروف و اعداد و کاراکتر _ باشد (دقت کنید که خط پایین یا underline است و علامت تفریق نیست).
- با یک حرف یا _ شروع شود یا به عبارت دیگر با عدد شروع نشود.
- توجه: زبان C++ به حروف بزرگ و کوچک حساس است.

انواع داده‌ها و متغیرها

متغیرها:

- برای ذخیره‌سازی داده‌ها در حافظه، از متغیرها استفاده می‌کنیم.
- در زبان C++ لازم است قبل از استفاده از هر متغیر، نوع داده‌ای که قرار است در متغیر ذخیره شود، مشخص شود. این کار را اصطلاحاً تعریف متغیر می‌نامند.
- گرامر دستور تعریف متغیر:

نام متغیرها که با کاما از هم جدا شده‌اند نوع متغیر;

مثال: int A,B;

- امکان تعریف متغیرها در هر مکانی قبل از اولین استفاده از متغیر ممکن است.
- تعریف متغیر در هر قسمت از برنامه، تاثیری در روند اجرای برنامه ایجاد نمی‌کند.

قوانین نامگذاری متغیرها:

- نام متغیر، جزء کلمات کلیدی زبان نباشد.
- ثانیاً ترکیبی از حروف و اعداد و کاراکتر _ باشد (دقت کنید که خط پایین یا underline است و علامت تفریق نیست).
- با یک حرف یا _ شروع شود یا به عبارت دیگر با عدد شروع نشود.
- توجه: زبان C++ به حروف بزرگ و کوچک حساس است.

عملگرهای زبان C++

عملگرهای زبان C++

- عملگر انتساب: برای دادن یک مقدار به یک متغیر. عملگر =

```
double i , j=0,k;
```

- عملگرهای محاسباتی:

عملگر	نماد	مثال
جمع	+	x+y
تفریق و تفریق یکانی	−	x−y یا −x
ضرب	*	x*y
تقسیم	/	x/y
باقیماندهٔ تقسیم	%	x%y
یک واحد افزایش	++	x++ یا ++x
یک واحد کاهش	−−	x−− یا −−x

- در زبان C++ عملگری برای محاسبهٔ توان وجود ندارد.

عملگرهای زبان C++

عملگرهای زبان C++

- عملگر انتساب: برای دادن یک مقدار به یک متغیر. عملگر =

```
double  i , j=0,k;
```

- عملگرهای محاسباتی:

عملگر	نماد	مثال
جمع	+	x+y
تفریق و تفریق یکانی	−	x−y یا −x
ضرب	*	x*y
تقسیم	/	x/y
باقیماندهٔ تقسیم	%	x%y
یک واحد افزایش	++	x++ یا ++x
یک واحد کاهش	−−	x−− یا −−x

- در زبان C++ عملگری برای محاسبهٔ توان وجود ندارد.

عملگرهای زبان C++

عملگرهای زبان C++

- عملگر انتساب: برای دادن یک مقدار به یک متغیر. عملگر =

```
double  i , j=0,k;
```

- عملگرهای محاسباتی:

عملگر	نماد	مثال
جمع	+	x+y
تفریق و تفریق یکانی	−	x−y یا −x
ضرب	*	x*y
تقسیم	/	x/y
باقیماندهٔ تقسیم	%	x%y
یک واحد افزایش	++	x++ یا ++x
یک واحد کاهش	−−	x−− یا −−x

- در زبان C++ عملگری برای محاسبهٔ توان وجود ندارد.

عملگرهای زبان C++

اولویت انجام عملیات محاسباتی:

فرض کنید متغیرهای x و y به ترتیب حاوی مقادیر ۱ و ۲ است. حاصل عبارت زیر چیست؟

$$x+y*2-4$$

جدول: جدول اولویت انجام عملگرها.

اولویت	عملگرها
اول	++ و --
دوم	- منهای یکانی
سوم	* و / و ٪
چهارم	+ و -

می‌توان ترتیب اجرا عملگرها را با استفاده از پرانتز عوض کرد. برای این منظور، عبارتی که داخلِ داخلی‌ترین پرانتز است ابتدا محاسبه می‌شود و سپس سایر عملگرها محاسبه می‌شوند.

$$((x+y)*2)-4$$

عملگرهای زبان C++

اولویت انجام عملیات محاسباتی:

فرض کنید متغیرهای x و y به ترتیب حاوی مقادیر ۱ و ۲ است. حاصل عبارت زیر چیست؟

```
x+y*2-4
```

جدول: جدول اولویت انجام عملگرها.

اولویت	عملگرها
اول	-- و ++
دوم	- منهای یکانی
سوم	* و / و ٪
چهارم	- و +

می‌توان ترتیب اجرا عملگرها را با استفاده از پرانتز عوض کرد. برای این منظور، عبارتی که داخلِ داخلی‌ترین پرانتز است ابتدا محاسبه می‌شود و سپس سایر عملگرها محاسبه می‌شوند.

```
((x+y)*2)-4
```

عملگرهای زبان C++

اولویت انجام عملیات محاسباتی:

فرض کنید متغیرهای x و y به ترتیب حاوی مقادیر ۱ و ۲ است. حاصل عبارت زیر چیست؟

```
x+y*2-4
```

جدول: جدول اولویت انجام عملگرها.

اولویت	عملگرها
اول	-- و ++
دوم	- منهای یکانی
سوم	* و / و %
چهارم	- و +

می‌توان ترتیب اجرا عملگرها را با استفاده از پرانتز عوض کرد. برای این منظور، عبارتی که داخلِ داخلی‌ترین پرانتز است ابتدا محاسبه می‌شود و سپس سایر عملگرها محاسبه می‌شوند.

```
((x+y)*2)-4
```

عملگرهای زبان C++

مود انجام محاسبات:

- همچنان که داده‌های مختلف، در فضاهای مختلف از حافظه و با فرمت‌های مختلف نگهداری می‌شوند، انجام عملیات محاسباتی روی آنها نیز متفاوت است. مثلاً جمع دو عدد صحیح با جمع دو عدد اعشاری متفاوت انجام می‌شود.
- اگر عملوندهای یک عمل جمع از نوع صحیح باشند، آنگاه عملیات اصطلاحاً در مود صحیح انجام شده و حاصل نیز یک عدد صحیح خواهد بود.
- پس از اجرای دستور x=1/2;، چه مقداری در متغیر x قرار می‌گیرد؟ خواسته صفر
- عدم دقت در این زمینه می‌تواند باعث بروز خطاهای منطقی در برنامه شود.
- راه حل: تغییر نوع متغیر یا ثابت.

```
x=1.0/2;
x=(float)1/2;
x=float(1)/2;
```

عملگرهای زبان C++

مود انجام محاسبات:

- همچنان که داده‌های مختلف، در فضاهای مختلف از حافظه و با فرمت‌های مختلف نگهداری می‌شوند، انجام عملیات محاسباتی روی آنها نیز متفاوت است. مثلاً جمع دو عدد صحیح با جمع دو عدد اعشاری متفاوت انجام می‌شود.
- اگر عملوندهای یک عمل جمع از نوع صحیح باشند، آنگاه عملیات اصطلاحاً در مود صحیح انجام شده و حاصل نیز یک عدد صحیح خواهد بود.
- پس از اجرای دستور x=1/2;، چه مقداری در متغیر x قرار می‌گیرد؟ خواهد صفر!
- عدم دقت در این زمینه می‌تواند باعث بروز خطاهای منطقی در برنامه شود.
- راه حل: تغییر نوع متغیر یا ثابت.

```
x=1.0/2;
x=(float)1/2;
x=float(1)/2;
```

عملگرهای زبان C++

مود انجام محاسبات:

- همچنان که داده‌های مختلف، در فضاهای مختلف از حافظه و با فرمت‌های مختلف نگهداری می‌شوند، انجام عملیات محاسباتی روی آنها نیز متفاوت است. مثلاً جمع دو عدد صحیح با جمع دو عدد اعشاری متفاوت انجام می‌شود.
- اگر عملوندهای یک عمل جمع از نوع صحیح باشند، آنگاه عملیات اصطلاحاً در مود صحیح انجام شده و حاصل نیز یک عدد صحیح خواهد بود.
- پس از اجرای دستور x=1/2;، چه مقداری در متغیر x قرار می‌گیرد؟ جواب: صفر!
- عدم دقت در این زمینه می‌تواند باعث بروز خطاهای منطقی در برنامه شود.
- راه حل: تغییر نوع متغیر یا ثابت.

```
x=1.0/2;
x=(float)1/2;
x=float(1)/2;
```

عملگرهای زبان C++

مود انجام محاسبات:

- همچنان که داده‌های مختلف، در فضاهای مختلف از حافظه و با فرمت‌های مختلف نگهداری می‌شوند، انجام عملیات محاسباتی روی آنها نیز متفاوت است. مثلاً جمع دو عدد صحیح با جمع دو عدد اعشاری متفاوت انجام می‌شود.
- اگر عملوندهای یک عمل جمع از نوع صحیح باشند، آنگاه عملیات اصطلاحاً در مود صحیح انجام شده و حاصل نیز یک عدد صحیح خواهد بود.
- پس از اجرای دستور x=1/2;، چه مقداری در متغیر x قرار می‌گیرد؟ جواب: صفر!
- عدم دقت در این زمینه می‌تواند باعث بروز خطاهای منطقی در برنامه شود.
- راه حل: تغییر نوع متغیر یا ثابت.

```
x=1.0/2;
x=(float)1/2;
x=float(1)/2;
```

عملگرهای زبان C++

مود انجام محاسبات:

- همچنان که داده‌های مختلف، در فضاهای مختلف از حافظه و با فرمت‌های مختلف نگهداری می‌شوند، انجام عملیات محاسباتی روی آنها نیز متفاوت است. مثلاً جمع دو عدد صحیح با جمع دو عدد اعشاری متفاوت انجام می‌شود.
- اگر عملوندهای یک عمل جمع از نوع صحیح باشند، آنگاه عملیات اصطلاحاً در مود صحیح انجام شده و حاصل نیز یک عدد صحیح خواهد بود.
- پس از اجرای دستور x=1/2;، چه مقداری در متغیر x قرار می‌گیرد؟ جواب: صفر!
- عدم دقت در این زمینه می‌تواند باعث بروز خطاهای منطقی در برنامه شود.
- راه حل: تغییر نوع متغیر یا ثابت.

```
x=1.0/2;
x=(float)1/2;
x=float(1)/2;
```

عملگرهای زبان C++

مود انجام محاسبات:

- همچنان که داده‌های مختلف، در فضاهای مختلف از حافظه و با فرمت‌های مختلف نگهداری می‌شوند، انجام عملیات محاسباتی روی آنها نیز متفاوت است. مثلاً جمع دو عدد صحیح با جمع دو عدد اعشاری متفاوت انجام می‌شود.
- اگر عملوندهای یک عمل جمع از نوع صحیح باشند، آنگاه عملیات اصطلاحاً در مود صحیح انجام شده و حاصل نیز یک عدد صحیح خواهد بود.
- پس از اجرای دستور x=1/2;، چه مقداری در متغیر x قرار می‌گیرد؟ **جواب: صفر!**
- عدم دقت در این زمینه می‌تواند باعث بروز خطاهای منطقی در برنامه شود.
- راه حل: تغییر نوع متغیر یا ثابت.

```
x=1.0/2;
x=(float)1/2;
x=float(1)/2;
```

عملگرهای زبان C++

عملگرهای مقایسه‌ای:

عملگر	نماد	مثال
بزرگ‌تر	>	x>y
بزرگ‌تر یا مساوی	>=	x>=y
کوچک‌تر	<	x<y
کوچک‌تر یا مساوی	<=	x<=y
تساوی	==	x==y
نامساوی	!=	x!=y

- در عملگرهایی که از ترکیب دو نماد تشکیل شده است؛ مثل =>، بین دو نماد نباید فاصله‌ای قرار بگیرد.
- عملگر تساوی به صورت == است. یکی از اشتباهات رایج در استفاده از عملگر تساوی، استفاده از یک نماد مساوی است که عملاً به جای بررسی دو مقدار، به عمل انتساب تبدیل می‌شود.

عملگرهای زبان C++

عملگرهای مقایسه‌ای:

عملگر	نماد	مثال
بزرگ‌تر	>	x>y
بزرگ‌تر یا مساوی	>=	x>=y
کوچک‌تر	<	x<y
کوچک‌تر یا مساوی	<=	x<=y
تساوی	==	x==y
نامساوی	!=	x!=y

- در عملگرهایی که از ترکیب دو نماد تشکیل شده است؛ مثل =>، بین دو نماد نباید فاصله‌ای قرار بگیرد.
- عملگر تساوی به صورت == است. یکی از اشتباهات رایج در استفاده از عملگر تساوی، استفاده از یک نماد مساوی است که عملاً به جای بررسی دو مقدار، به عمل انتساب تبدیل می‌شود.

عملگرهای زبان C++

عملگرهای مقایسه‌ای:

عملگر	نماد	مثال
بزرگ‌تر	>	x>y
بزرگ‌تر یا مساوی	>=	x>=y
کوچک‌تر	<	x<y
کوچک‌تر یا مساوی	<=	x<=y
تساوی	==	x==y
نامساوی	!=	x=y!

- در عملگرهایی که از ترکیب دو نماد تشکیل شده است؛ مثل =>، بین دو نماد نباید فاصله‌ای قرار بگیرد.
- عملگر تساوی به صورت == است. یکی از اشتباهات رایج در استفاده از عملگر تساوی، استفاده از یک نماد مساوی است که عملاً به جای بررسی دو مقدار، به عمل انتساب تبدیل می‌شود.

عملگرهای زبان C++

عملگرهای منطقی:

عملگرهای منطقی:

عملگر	نماد	مثال
نقیض	!	!(x==y)
و	&&	x<0 && y>1
یا	||	x<0 || y>1

اولویت عملگرهای منطقی و رابطه‌ای:

عملگرهای زبان C++

عملگرهای منطقی:

- عملگرهای منطقی:

عملگر	نماد	مثال
نقیض	!	!(x==y)
و	&&	x<0 && y>1
یا	||	x<0 || y>1

- اولویت عملگرهای منطقی و رابطه‌ای:

اولویت	عملگرها
اول	!
دوم	<, >, <=, >=
سوم	==, ! =
چهارم	&&
پنجم	||

عملگرهای زبان C++

عملگرهای ترکیبی:

عملگر	نماد	مثال	معادل با
انتساب جمع	+=	x+=y	x=x+y
انتساب تفریق	−=	x−=y	x=x−y
انتساب ضرب	*=	x*=y	x=x*y
انتساب تقسیم	/=	x/=y	x=x/y
انتساب باقیماندهٔ تقسیم	%=	x%=y	x=x%y

اولویت عملگرها:

عملگرهای زبان C++

عملگرهای ترکیبی:

عملگر	نماد	مثال	معادل با
انتساب جمع	+=	x+=y	x=x+y
انتساب تفریق	−=	x−=y	x=x−y
انتساب ضرب	*=	x*=y	x=x*y
انتساب تقسیم	/=	x/=y	x=x/y
انتساب باقیماندهٔ تقسیم	%=	x%=y	x=x%y

اولویت عملگرها:

اولویت	عملگرها
اول	−, !, ++, −− (منهای یکانی)
دوم	*, /, %
سوم	+, −
چهارم	<, <=, >, >=
پنجم	==, ! =
ششم	&&, ||
هفتم	=, + =, − =, * =, / =, % =

ورودی و خروجی

ورودی و خروجی:

- در زبان C++ دستوری جهت ورودی و خروجی وجود ندارد ولی از یک کتابخانه به نام iostream جهت این منظور استفاده می‌کنیم.
- گرفتن یک مقدار و قراردادن آن در متغیر x:

```
cin>>x;
```

```
cin>>x>>y>>z;
```

- جهت خروجی:

```
cout<<x;
```

```
cout<<x<<y<<z;
```

- جهت خروجی یک عبارت، یعنی نوشتن یک عبارت متنی در خروجی،

```
cout<<"The Number is Prime.";
```

ورودی و خروجی

ورودی و خروجی:

- در زبان C++ دستوری جهت ورودی و خروجی وجود ندارد ولی از یک کتابخانه به نام iostream جهت این منظور استفاده می‌کنیم.
- گرفتن یک مقدار و قراردادن آن در متغیر x:

```
cin>>x;
cin>>x>>y>>z;
```

- جهت خروجی:

```
cout<<x;
cout<<x<<y<<z;
```

- جهت خروجی یک عبارت، یعنی نوشتن یک عبارت متنی در خروجی،

```
cout<<"The Number is Prime.";
```

ورودی و خروجی

ورودی و خروجی:

- در زبان C++ دستوری جهت ورودی و خروجی وجود ندارد ولی از یک کتابخانه به نام iostream جهت این منظور استفاده می‌کنیم.
- گرفتن یک مقدار و قراردادن آن در متغیر x:

```
cin>>x;
cin>>x>>y>>z;
```

- جهت خروجی:

```
cout<<x;
cout<<x<<y<<z;
```

- جهت خروجی یک عبارت، یعنی نوشتن یک عبارت متنی در خروجی،

```
cout<<"The Number is Prime.";
```

ورودی و خروجی

ورودی و خروجی:

- در زبان C++ دستوری جهت ورودی و خروجی وجود ندارد ولی از یک کتابخانه به نام iostream جهت این منظور استفاده می‌کنیم.
- گرفتن یک مقدار و قراردادن آن در متغیر x:

```
cin>>x;
cin>>x>>y>>z;
```

- جهت خروجی:

```
cout<<x;
cout<<x<<y<<z;
```

- جهت خروجی یک عبارت، یعنی نوشتن یک عبارت متنی در خروجی،

```
cout<<"The Number is Prime.";
```

ساختار یک برنامه

ساختار یک برنامه:

توجه: شماره خطوط (قرمز رنگ) جزء برنامه نیستند و صرفاً برای ارجاع به خطوط برنامه اضافه شده‌اند.

```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5       :
6       :
7       :
8       return 0;
9 }
```

شکل: ساختار کلی یک برنامه به زبان C++.

مثال ۱

برنامه‌ای بنویسید که محیط یک مستطیل به طول ۳ و عرض ۴ را محاسبه و چاپ کند.

الگوریتم:

۱ شروع
۲ $(۴ + ۳) \times ۲$ را چاپ کن
۳ پایان

برنامه:

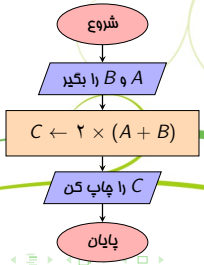
```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5     cout<< 2*(3+4);
6     return 0;
7 }
```

مثال ۲

برنامه‌ای بنویسید که طول و عرض یک مستطیل را دریافت کرده و محیط مستطیل را محاسبه و چاپ کند.

الگوریتم:

۱ شروع
۲ طول و عرض مستطیل را بگیر و در متغیرهای A و B قرار بده.
۳ قرار بده $C \leftarrow ۲ \times (A + B)$
۴ C را چاپ کن
۵ پایان



برنامه:

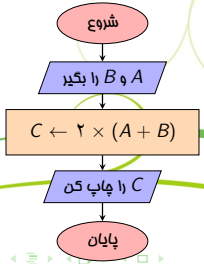
```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      double A,B,C;
6      cout<<"Enter Height and Width of
           rectangle:";
7      cin>>A>>B;
8      C=2*(A+B);
9      cout<<C;
10     return 0;
11 }
```

مثال ۲

برنامه‌ای بنویسید که طول و عرض یک مستطیل را دریافت کرده و محیط مستطیل را محاسبه و چاپ کند.

الگوریتم:

۱ شروع
۲ طول و عرض مستطیل را بگیر و در متغیرهای A و B قرار بده.
۳ قرار بده $C \leftarrow ۲ \times (A + B)$
۴ C را چاپ کن
۵ پایان



برنامه:

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5       double A,B,C;
6       cout<<"Enter Height and Width of
            rectangle:";
7       cin>>A>>B;
8       C=2*(A+B);
9       cout<<C;
10      return 0;
11 }
```

مثال ۳

برنامه‌ای بنویسید که سه عدد را دریافت کند و مجموع و میانگین آنها را محاسبه و چاپ کند.

الگوریتم:

۱ شروع
۲ سه عدد را از ورودی را بگیر و آنها را در متغیرهای A و B و C قرار بده.
۳ قرار بده $S \leftarrow A + B + C$
۴ قرار بده $AVE \leftarrow S/۳$
۵ S و AVE را چاپ کن
۶ پایان

مثال ۳

برنامه‌ای بنویسید که سه عدد را دریافت کند و مجموع و میانگین آنها را محاسبه و چاپ کند.

الگوریتم:

۱ شروع
۲ سه عدد را از ورودی را بگیر و آنها را در متغیرهای A و B و C قرار بده.
۳ قرار بده $S \leftarrow A + B + C$
۴ قرار بده $AVE \leftarrow S/۳$
۵ S و AVE را چاپ کن
۶ پایان

برنامه:

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5       float A,B,C ;
6      double AVE,S;
7      cout <<"Enter three numbers:";
8      cin>>A>>B>>C;
9      S=A+B+C;
10     AVE=S/3;
11     cout<<"Sum is:"<<S;
12     cout<<" Average is: "<<AVE;
13     return 0;
14 }
```

مثال ۴

برنامه‌ای بنویسید که دو عدد را دریافت کرده و در متغیرهای A و B ذخیره کند و سپس محتوای دو متغیر را جابجا کرده و نتیجه را چاپ کند.

الگوریتم:

۱ شروع
۲ دو عدد را از ورودی را بگیر و آن را در متغیرهای A و B قرار بده.
۳ $C \leftarrow A$
۴ $A \leftarrow B$
۵ $B \leftarrow C$
۶ A و B را چاپ کن
۷ پایان

مثال ۴

برنامه‌ای بنویسید که دو عدد را دریافت کرده و در متغیرهای A و B ذخیره کند و سپس محتوای دو متغیر را جابجا کرده و نتیجه را چاپ کند.

الگوریتم:

۱ شروع
۲ دو عدد را از ورودی را بگیر و آن را در متغیرهای A و B قرار بده.
۳ $C \leftarrow A$
۴ $A \leftarrow B$
۵ $B \leftarrow C$
۶ A و B را چاپ کن
۷ پایان

برنامه:

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      double A,B,C;
6      cout<<"Enter two numbers:";
7      cin>>A>>B;
8      C=A;
9      A=B;
10     B=C;
11     cout<<"A="<<A<<
12       ", B="<<B;
13     return 0;
14 }
```

مثال ۴

برنامه‌ای بنویسید که دو عدد را دریافت کرده و در متغیرهای A و B ذخیره کند و سپس محتوای دو متغیر را جابجا کرده و نتیجه را چاپ کند.

الگوریتم:

۱ شروع
۲ دو عدد را از ورودی را بگیر و آن را در متغیرهای A و B قرار بده.
۳ $A \leftarrow A + B$
۴ $B \leftarrow A - B$
۵ $A \leftarrow A - B$
۶ A و B را چاپ کن
۷ پایان

مثال ۴

برنامه‌ای بنویسید که دو عدد را دریافت کرده و در متغیرهای A و B ذخیره کند و سپس محتوای دو متغیر را جابجا کرده و نتیجه را چاپ کند.

الگوریتم:

۱ شروع
۲ دو عدد را از ورودی را بگیر و آن را در متغیرهای A و B قرار بده.
۳ $A \leftarrow A + B$
۴ $B \leftarrow A - B$
۵ $A \leftarrow A - B$
۶ A و B را چاپ کن
۷ پایان

برنامه:

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      double A, B ;
6      cout<<" Enter two numbers:";
7      cin>> A>>B;
8      A=A+B;
9      B=A-B;
10     A=A-B;
11     cout<<"A="<<A
12         <<", B="<<B;
13     return 0;
14 }
```

یک دستور یا یک بلوک از دستورات

یک دستور یا یک بلوک از دستورات:

- منظور از یک دستور در زبان C++ ، قسمتی از برنامه است که به یک سمی کالن ختم می‌شود.
- منظور از یک بلوک از دستورات، یک یا چند دستور است که در بین یک آکولاد باز و یک آکولاد بسته محصور شده است.
- در زبان C++ ، یک بلوک از دستورات دقیقاً کارکردی مشابه یک دستور دارد و هر کجا یک دستور قرار می‌گیرد را می‌توان با یک بلوک از دستورات نیز جایگزین کرد.
- نکته مهم در خصوص یک بلوک از دستورات، وجود آکولاد باز و آکولاد بسته اول وآخر بلوک است.

یک دستور یا یک بلوک از دستورات

یک دستور یا یک بلوک از دستورات:

- منظور از یک دستور در زبان C++ ، قسمتی از برنامه است که به یک سمی کالن ختم می‌شود.
- منظور از یک بلوک از دستورات، یک یا چند دستور است که در بین یک آکولاد باز و یک آکولاد بسته محصور شده است.
- در زبان C++ ، یک بلوک از دستورات دقیقاً کارکردی مشابه یک دستور دارد و هر کجا یک دستور قرار می‌گیرد را می‌توان با یک بلوک از دستورات نیز جایگزین کرد.
- نکته مهم در خصوص یک بلوک از دستورات، وجود آکولاد باز و آکولاد بسته اول وآخر بلوک است.

یک دستور یا یک بلوک از دستورات

یک دستور یا یک بلوک از دستورات:

- منظور از یک دستور در زبان C++ ، قسمتی از برنامه است که به یک سمی کالن ختم می‌شود.
- منظور از یک بلوک از دستورات، یک یا چند دستور است که در بین یک آکولاد باز و یک آکولاد بسته محصور شده است.
- در زبان C++ ، یک بلوک از دستورات دقیقاً کارکردی مشابه یک دستور دارد و هر کجا یک دستور قرار می‌گیرد را می‌توان با یک بلوک از دستورات نیز جایگزین کرد.
- نکته مهم در خصوص یک بلوک از دستورات، وجود آکولاد باز و آکولاد بسته اول وآخر بلوک است.

یک دستور یا یک بلوک از دستورات

یک دستور یا یک بلوک از دستورات:

- منظور از یک دستور در زبان C++ ، قسمتی از برنامه است که به یک سمی کالن ختم می‌شود.
- منظور از یک بلوک از دستورات، یک یا چند دستور است که در بین یک آکولاد باز و یک آکولاد بسته محصور شده است.
- در زبان C++ ، یک بلوک از دستورات دقیقاً کارکردی مشابه یک دستور دارد و هر کجا یک دستور قرار می‌گیرد را می‌توان با یک بلوک از دستورات نیز جایگزین کرد.
- نکته مهم در خصوص یک بلوک از دستورات، وجود آکولاد باز و آکولاد بسته اول وآخر بلوک است.

دستور شرطی

دستور شرطی:

- در برخی از الگوریتم‌ها نیاز به بررسی یک شرط داریم و بر مبنای درستی یا نادرستی آن شرط عملیات مختلفی انجام می‌شود. (لوزی در فلوچارت)
- در C++ دستوری جهت بررسی شروط و انجام کارهایی بر مبنای آن وجود دارد. این دستور که به دستور شرطی معروف است به شکل زیر است:

```
if (شرط)
    یک دستور یا یک بلوک از دستورات
else
    یک دستور یا یک بلوک از دستورات
```

- قسمت مربوط به else در صورت عدم نیاز قابل حذف شدن است.
- دقت شود که دستور شرطی هیچ سمی کالنی ندارد، اما در قسمت دستور یا بلوک دستورات، بسته به دستور استفاده شده، ممکن است سمی کالن نیاز باشد.

دستور شرطی

دستور شرطی:

- در برخی از الگوریتم‌ها نیاز به بررسی یک شرط داریم و بر مبنای درستی یا نادرستی آن شرط عملیات مختلفی انجام می‌شود. (لوزی در فلوچارت)
- در C++ دستوری جهت بررسی شروط و انجام کارهایی بر مبنای آن وجود دارد. این دستور که به دستور شرطی معروف است به شکل زیر است:

```
if (شرط)
    یک دستور یا یک بلوک از دستورات
else
    یک دستور یا یک بلوک از دستورات
```

- قسمت مربوط به else در صورت عدم نیاز قابل حذف شدن است.
- دقت شود که دستور شرطی هیچ سمی کالنی ندارد، اما در قسمت دستور یا بلوک دستورات، بسته به دستور استفاده شده، ممکن است سمی کالن نیاز باشد.

دستور شرطی

دستور شرطی:

- در برخی از الگوریتم‌ها نیاز به بررسی یک شرط داریم و بر مبنای درستی یا نادرستی آن شرط عملیات مختلفی انجام می‌شود. (لوزی در فلوچارت)
- در C++ دستوری جهت بررسی شروط و انجام کارهایی بر مبنای آن وجود دارد. این دستور که به دستور شرطی معروف است به شکل زیر است:

```
if (شرط)
    یک دستور یا یک بلوک از دستورات
else
    یک دستور یا یک بلوک از دستورات
```

- قسمت مربوط به else در صورت عدم نیاز قابل حذف شدن است.
- دقت شود که دستور شرطی هیچ سمی کالنی ندارد، اما در قسمت دستور یا بلوک دستورات، بسته به دستور استفاده شده، ممکن است سمی کالن نیاز باشد.

دستور شرطی

دستور شرطی:

- در برخی از الگوریتم‌ها نیاز به بررسی یک شرط داریم و بر مبنای درستی یا نادرستی آن شرط عملیات مختلفی انجام می‌شود. (لوزی در فلوچارت)
- در C++ دستوری جهت بررسی شروط و انجام کارهایی بر مبنای آن وجود دارد. این دستور که به دستور شرطی معروف است به شکل زیر است:

```
if (شرط)
    یک دستور یا یک بلوک از دستورات
else
    یک دستور یا یک بلوک از دستورات
```

- قسمت مربوط به else در صورت عدم نیاز قابل حذف شدن است.
- دقت شود که دستور شرطی هیچ سمی کالنی ندارد، اما در قسمت دستور یا بلوک دستورات، بسته به دستور استفاده شده، ممکن است سمی کالن نیاز باشد.

مثال ۵

برنامه‌ای بنویسید که یک عدد صحیح را از ورودی دریافت کند و مشخص کند عدد وارد شده زوج است یا فرد.

الگوریتم:

۱ شروع
۲ n را بگیر
۳ قرار بده $r \leftarrow n - [n/۲] * ۲$
۴ اگر $r = ۰$ آنگاه
۵ | «زوج» را چاپ کن
۶ درغیراینصورت
۷ | «فرد» را چاپ کن
۸ پایان اگر
۹ پایان

مثال ۵

برنامه‌ای بنویسید که یک عدد صحیح را از ورودی دریافت کند و مشخص کند عدد وارد شده زوج است یا فرد.

الگوریتم:

۱ شروع
۲ n را بگیر
۳ قرار بده $r \leftarrow n - [n/۲] * ۲$
۴ اگر $r = ۰$ آنگاه
۵ | «زوج» را چاپ کن
۶ درغیراینصورت
۷ | «فرد» را چاپ کن
۸ پایان اگر
۹ پایان

برنامه:

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5       int   n , r;
6       cout<<"Enter a number";
7       cin>>n;
8       r=n%2;
9       if (r==0)
10           cout<<"Even";
11      else
12           cout<<"Odd";
13      return 0;
14 }
```

مثال ۶

برنامه‌ای بنویسید که یک عدد را از ورودی دریافت کند و قدرمطلق آن را محاسبه و چاپ کند.

الگوریتم:

۱ شروع
۲ x را بگیر
۳ اگر $x < ۰$ آنگاه
۴ | قرار بده $y \leftarrow -۱ \times x$
۵ درغیراینصورت
۶ | قرار بده $y \leftarrow x$
۷ پایان اگر
۸ y را چاپ کن
۹ پایان

مثال ۶

برنامه‌ای بنویسید که یک عدد را از ورودی دریافت کند و قدرمطلق آن را محاسبه و چاپ کند.

الگوریتم:

۱ شروع
۲ x را بگیر
۳ اگر $x < ۰$ آنگاه
۴ | قرار بده $y \leftarrow x \times -۱$
۵ درغیراینصورت
۶ | قرار بده $y \leftarrow x$
۷ پایان اگر
۸ y را چاپ کن
۹ پایان

برنامه:

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      double x,y;
6      cout <<" Enter x :";
7      cin >> x ;
8      if ( x < 0 )
9          y = -1 * x;
10     else
11         y = x ;
12     cout << y;
13     return 0;
14 }
```

مثال ۷

برنامه‌ای بنویسید که سه عدد را از ورودی دریافت کند و بیشترین عدد را محاسبه و چاپ کند.

الگوریتم:

۱ شروع
۲ A و B و C را بگیر
۳ $MAX \leftarrow A$
۴ اگر $B > MAX$ آنگاه
۵ | قرار بده $MAX \leftarrow B$
۶ پایان اگر
۷ اگر $C > MAX$ آنگاه
۸ | قرار بده $MAX \leftarrow C$
۹ پایان اگر
۱۰ MAX را چاپ کن
۱۱ پایان

مثال ۷

برنامه‌ای بنویسید که سه عدد را از ورودی دریافت کند و بیشترین عدد را محاسبه و چاپ کند.

الگوریتم:

۱ شروع
۲ A و B و C را بگیر
۳ $MAX \leftarrow A$
۴ اگر $B > MAX$ آنگاه
۵ | قرار بده $MAX \leftarrow B$
۶ پایان اگر
۷ اگر $C > MAX$ آنگاه
۸ | قرار بده $MAX \leftarrow C$
۹ پایان اگر
۱۰ MAX را چاپ کن
۱۱ پایان

برنامه:

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      double   A,B,C,MAX ;
6      cout<<"Enter three numbers:";
7      cin>>A>>B>>C;
8      MAX=A;
9      if ( B > MAX )
10         MAX=B;
11     if ( C > MAX )
12         MAX =C;
13     cout<<"The Maximum is "<<
           MAX;
14     return 0;
15 }
```

مثال ۸

برنامه‌ای بنویسید که سه عدد را از ورودی دریافت کند و مشخص کند آیا مثلث قائم‌الزاویه‌ای با آن طول اضلاع وجود دارد یا خیر.

الگوریتم:

۱ شروع
۲ A و B و C را بگیر
۳ قرار بده $flag \leftarrow 0$
۴ اگر $A^2 = B^2 + C^2$ آنگاه
۵ | قرار بده $flag \leftarrow 1$
۶ پایان اگر
۷ اگر $B^2 = A^2 + C^2$ آنگاه
۸ | قرار بده $flag \leftarrow 1$
۹ پایان اگر
۱۰ اگر $C^2 = A^2 + B^2$ آنگاه
۱۱ | قرار بده $flag \leftarrow 1$
۱۲ پایان اگر
۱۳ اگر $flag = 1$ آنگاه
۱۴ | YES را چاپ کن
۱۵ درغیراینصورت
۱۶ | NO را چاپ کن
۱۷ پایان اگر
۱۸ پایان

مثال ۸

برنامه‌ای بنویسید که سه عدد را از ورودی دریافت کند و مشخص کند آیا مثلث قائم‌الزاویه‌ای با آن طول اضلاع وجود دارد یا خیر.

الگوریتم:

۱ شروع
۲ A و B و C را بگیر
۳ قرار بده $flag \leftarrow 0$
۴ اگر $A^2 = B^2 + C^2$ آنگاه
۵ | قرار بده $flag \leftarrow 1$
۶ پایان اگر
۷ اگر $B^2 = A^2 + C^2$ آنگاه
۸ | قرار بده $flag \leftarrow 1$
۹ پایان اگر
۱۰ اگر $C^2 = A^2 + B^2$ آنگاه
۱۱ | قرار بده $flag \leftarrow 1$
۱۲ پایان اگر
۱۳ اگر $flag = 1$ آنگاه
۱۴ | YES را چاپ کن
۱۵ درغیراینصورت
۱۶ | NO را چاپ کن
۱۷ پایان اگر
۱۸ پایان

برنامه:

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      double A,B,C, flag =0;
6      cout<<"Enter three
           numbers:";
7      cin>>A>>B>>C;
8      if (A*A ==B*B +C*C)
9          flag=1;
10     if ( B*B==A*A +C*C)
11         flag =1;
12     if (C*C==B*B+A*A)
13         flag=1;
14     if ( flag==1)
15         cout<<"YES";
16     else
17         cout<<"NO";
18     return 0;
19 }
```

مثال ۹

برنامه‌ای بنویسید که سه عدد را از ورودی دریافت کند و آنها را به ترتیب صعودی در خروجی چاپ کند.

الگوریتم:

```
۱  شروع
۲  A و B و C را بگیر
۳  اگر A ≥ B آنگاه
۴      قرار بده x ← B و
       y ← A
۵  درغیراینصورت
۶      قرار بده x ← A و
       y ← B
۷  پایان اگر
۸  اگر x ≥ C آنگاه
۹      قرار بده y ← z و
       x ← y و C ← x
۱۰ درغیراینصورت
۱۱     اگر y ≥ C آنگاه
۱۲         قرار بده z ← y و
           y ← C
۱۳     درغیراینصورت
۱۴         قرار بده z ← C
۱۵     پایان اگر
۱۶ پایان اگر
۱۷ اول x و سپس y و سپس z
   را چاپ کن
۱۸ پایان
```

مثال ۹

برنامه‌ای بنویسید که سه عدد را از ورودی دریافت کند و آنها را به ترتیب صعودی در خروجی چاپ کند.

الگوریتم:

۱ شروع
۲ A و B و C را بگیر
۳ اگر $B \geq A$ آنگاه
۴ | قرار بده $B \leftarrow x$ و $A \leftarrow y$
۵ درغیراینصورت
۶ | قرار بده $A \leftarrow x$ و $B \leftarrow y$
۷ پایان اگر
۸ اگر $C \geq x$ آنگاه
۹ | قرار بده $y \leftarrow z$ و $C \leftarrow y$ و $x \leftarrow z$
۱۰ درغیراینصورت
۱۱ | اگر $C \geq y$ آنگاه
۱۲ | | قرار بده $y \leftarrow z$ و $C \leftarrow y$
۱۳ | درغیراینصورت
۱۴ | | قرار بده $C \leftarrow z$
۱۵ | پایان اگر
۱۶ پایان اگر
۱۷ اول x و سپس y و سپس z را چاپ کن
۱۸ پایان

برنامه:

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      double A,B,C,x,y,z;
6      cout<<"Enter three
           numbers:";
7      cin>>A>>B>>C;
8      if (A>=B)
9          { x=B; y=A; }
10     else
11         { x=A; y=B; }
12     if (x>=C)
13         { z=y; y=x; x=C; }
14     else
15         if (y>=C)
16             { z=y; y=C; }
17         else
18             z=C;
19     cout<<x<<","<<y<<","
           <<z;
20     return 0;
21 }
```

دستور switch

دستور switch:

در حالتی که می‌خواهیم بر مبنای مقدار یک متغیر یا یک عبارت محاسباتی تعداد محدودی حالت را بررسی و بر مبنای آن کار خاصی انجام دهیم، می‌توانیم از دستور switch استفاده کنیم.

این کار با استفاده از دستور if نیز امکان‌پذیر است ولی در برخی حالات، استفاده از دستور switch راحت‌تر است.

ساختار گرامری دستور switch:

```
switch( expression )
{
    case constant-expression1:
        statement(s);
        break;
    case constant-expression2:
        statement(s);
        break;
    ...
    default :
        statement(s);
}
```

دستور switch

دستور switch:

- در حالتی که می‌خواهیم بر مبنای مقدار یک متغیر یا یک عبارت محاسباتی تعداد محدودی حالت را بررسی و بر مبنای آن کار خاصی انجام دهیم، می‌توانیم از دستور switch استفاده کنیم.
- این کار با استفاده از دستور if نیز امکان‌پذیر است ولی در برخی حالات، استفاده از دستور switch راحت‌تر است.
- ساختار گرامری دستور switch:

```
switch ( expression )
{
    case constant-expression1 :
        statement(s);
        break;
    case constant-expression2 :
        statement(s);
        break;
    ...
    default :
        statement(s);
}
```

دستور switch

دستور switch:

- در حالتی که می‌خواهیم بر مبنای مقدار یک متغیر یا یک عبارت محاسباتی تعداد محدودی حالت را بررسی و بر مبنای آن کار خاصی انجام دهیم، می‌توانیم از دستور switch استفاده کنیم.
- این کار با استفاده از دستور if نیز امکان‌پذیر است ولی در برخی حالات، استفاده از دستور switch راحت‌تر است.
- ساختار گرامری دستور switch:

```
1  switch ( expression )
2  {
3      case constant-expression1 :
4          statement(s);
5          break;
6      case constant-expression2 :
7          statement(s);
8          break;
9          ...
10     default :
11         statement(s);
12 }
```

دستور switch

دستور switch:

ساختار گرامری دستور switch:

```
switch ( expression )
{
    case constant−expression1 :
        statement(s);
        break;
    case constant−expression2 :
        statement(s);
        break;
    ...
    default :
        statement(s);
}
```

عملکرد این دستور به این صورت است که مقدار عبارت expression بررسی شده و در صورتی که این مقدار برابر با constant−expression1 باشد، دستورات بعد از آن دستور تا دستور break اجرا شده و سپس اجرای برنامه به اولین دستور بعد از دستور switch منتقل می‌شود. این کار برای سایر عبارت‌ها نیز تکرار می‌شود.

در صورتی که expression با هیچ یک از عبارات جلو caseها برابر نباشد، دستورات قسمت default اجرا می‌شود.

دستور switch

دستور switch:

- ساختار گرامری دستور switch:

```
switch ( expression )
{
    case constant−expression1 :
        statement(s);
        break;
    case constant−expression2 :
        statement(s);
        break;
    ...
    default :
        statement(s);
}
```

- عملکرد این دستور به این صورت است که مقدار عبارت expression بررسی شده و در صورتی که این مقدار برابر با constant−expression1 باشد، دستورات بعد از آن دستور تا دستور break اجرا شده و سپس اجرای برنامه به اولین دستور بعد از دستور switch منتقل می‌شود. این کار برای سایر عبارت‌ها نیز تکرار می‌شود.
- در صورتی که expression با هیچ یک از عبارات جلو caseها برابر نباشد، دستورات قسمت default اجرا می‌شود.

دستور switch

دستور switch:

- ساختار گرامری دستور switch:

```
switch ( expression )
{
    case constant−expression1 :
        statement(s);
        break;
    case constant−expression2 :
        statement(s);
        break;
    ...
    default :
        statement(s);
}
```

- عملکرد این دستور به این صورت است که مقدار عبارت expression بررسی شده و در صورتی که این مقدار برابر با constant−expression1 باشد، دستورات بعد از آن دستور تا دستور break اجرا شده و سپس اجرای برنامه به اولین دستور بعد از دستور switch منتقل می‌شود. این کار برای سایر عبارت‌ها نیز تکرار می‌شود.
- در صورتی که expression با هیچ یک از عبارات جلو caseها برابر نباشد، دستورات قسمت default اجرا می‌شود.

دستور switch

نکات زیر در خصوص استفاده از دستور switch باید رعایت شود:

- حاصل عبارت expression باید یک مقدار صحیح یا از یک نوع مقدار قابل شمارش باشد.
- حاصل عبارات constant—expression باید از همان نوع expression و یک مقدار ثابت یا متغیر باشد.
- دستور break در انتهای دستورات هر case اختیاری است. قانون کلی این است که در صورت ورود به یک حالت case، دستورات بعدی تا رسیدن به یک دستور break اجرا می‌شود.
- استفاده از بخش default نیز اختیاری است. این بخش در صورتی اجرا می‌شود که عبارت expression با هیچکدام از عبارات مقابل caseها برابر نباشد.

دستور switch

نکات زیر در خصوص استفاده از دستور switch باید رعایت شود:

- حاصل عبارت expression باید یک مقدار صحیح یا از یک نوع مقدار قابل شمارش باشد.
- حاصل عبارات constant—expression باید از همان نوع expression و یک مقدار ثابت یا متغیر باشد.
- دستور break در انتهای دستورات هر case اختیاری است. قانون کلی این است که در صورت ورود به یک حالت case، دستورات بعدی تا رسیدن به یک دستور break اجرا می‌شود.
- استفاده از بخش default نیز اختیاری است. این بخش در صورتی اجرا می‌شود که عبارت expression با هیچکدام از عبارات مقابل caseها برابر نباشد.

دستور switch

نکات زیر در خصوص استفاده از دستور switch باید رعایت شود:

- حاصل عبارت expression باید یک مقدار صحیح یا از یک نوع مقدار قابل شمارش باشد.
- حاصل عبارات constant—expression باید از همان نوع expression و یک مقدار ثابت یا متغیر باشد.
- دستور break در انتهای دستورات هر case اختیاری است. قانون کلی این است که در صورت ورود به یک حالت case، دستورات بعدی تا رسیدن به یک دستور break اجرا می‌شود.
- استفاده از بخش default نیز اختیاری است. این بخش در صورتی اجرا می‌شود که عبارت expression با هیچکدام از عبارات مقابل caseها برابر نباشد.

دستور switch

نکات زیر در خصوص استفاده از دستور switch باید رعایت شود:

- حاصل عبارت expression باید یک مقدار صحیح یا از یک نوع مقدار قابل شمارش باشد.
- حاصل عبارات constant—expression باید از همان نوع expression و یک مقدار ثابت یا متغیر باشد.
- دستور break در انتهای دستورات هر case اختیاری است. قانون کلی این است که در صورت ورود به یک حالت case، دستورات بعدی تا رسیدن به یک دستور break اجرا می‌شود.
- استفاده از بخش default نیز اختیاری است. این بخش در صورتی اجرا می‌شود که عبارت expression با هیچکدام از عبارات مقابل caseها برابر نباشد.

http://pws.yazd.ac.ir/farshi/ محمّد فرشی- دانشگاه یزد

دستور switch

مثال:

```
1  #include <iostream>
2  using namespace std;
3  int main ()
4  {
5      char grade;
6      cout<<"Enter grade:";
7      cin>>grade;
8      switch(grade)
9      {
10       case 'A' :
11         cout<<"Excellent!";
12         break;
13       case 'B' :
14       case 'C' :
15         cout<<"Well done";
16         break;
17       case 'D' :
18         cout<<"You passed";
19         break;
20       case 'F' :
21         cout<<"Better try again";
22         break;
23       default :
24         cout<<"Invalid grade";
25     }
26     cout<<"Your grade is "<<grade;
27     return 0;
28 }
```

ساختار حلقه تکرار

ساختار حلقه تکرار:

- در حل بسیاری از مسائل نیاز داشتیم با استفاده از حلقه‌های تکرار، تعدادی دستور را چندین بار اجرا کنیم.
- با دانستن دستور شرطی در زبان C++ و معرفی دستوری که معادل «برو به مرحلهٔ ....» باشد، می‌توان حلقه‌ها را در زبان برنامه‌نویسی پیاده کرد. البته چنین دستوری نیز در C++ وجود دارد اما به دلیل ساختاریافته نبودن این شیوه برای حلقه‌های تکرار، از دستورات حلقه تکرار در C++ استفاده خواهیم کرد.
- در زبان C++ سه نوع حلقهٔ تکرار وجود دارد. (حلقهٔ for، حلقهٔ while و حلقهٔ do while).

ساختار حلقه تکرار

ساختار حلقه تکرار:

- در حل بسیاری از مسائل نیاز داشتیم با استفاده از حلقه‌های تکرار، تعدادی دستور را چندین بار اجرا کنیم.
- با دانستن دستور شرطی در زبان C++ و معرفی دستوری که معادل «برو به مرحلهٔ ....» باشد، می‌توان حلقه‌ها را در زبان برنامه‌نویسی پیاده کرد. البته چنین دستوری نیز در C++ وجود دارد اما به دلیل ساختاریافته نبودن این شیوه برای حلقه‌های تکرار، از دستورات حلقه تکرار در C++ استفاده خواهیم کرد.
- در زبان C++ سه نوع حلقهٔ تکرار وجود دارد. (حلقهٔ for، حلقهٔ while و حلقهٔ do while).

ساختار حلقه تکرار

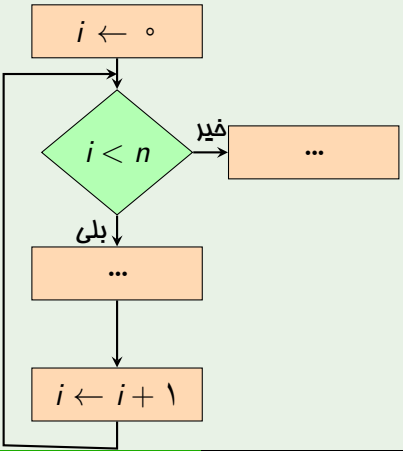
ساختار حلقه تکرار:

- در حل بسیاری از مسائل نیاز داشتیم با استفاده از حلقه‌های تکرار، تعدادی دستور را چندین بار اجرا کنیم.
- با دانستن دستور شرطی در زبان C++ و معرفی دستوری که معادل «برو به مرحلهٔ ....» باشد، می‌توان حلقه‌ها را در زبان برنامه‌نویسی پیاده کرد. البته چنین دستوری نیز در C++ وجود دارد اما به دلیل ساختاریافته نبودن این شیوه برای حلقه‌های تکرار، از دستورات حلقه تکرار در C++ استفاده خواهیم کرد.
- در زبان C++ سه نوع حلقهٔ تکرار وجود دارد. (حلقهٔ for، حلقهٔ while و حلقهٔ do while).

ساختار حلقه تکرار

دستور حلقهٔ for

دستور حلقهٔ for:



ساختار حلقه تکرار

دستور حلقهٔ for

دستور حلقهٔ for:

for (دستور ; شرط ; دستور)
یک دستور یا یک بلوک از دستورات

- دقت کنید که در پرانتز روبروی عبارت for دو سمی کالن وجود دارد که سه عبارت را از هم جدا می‌کند.
- حتی با نبود دستورات و شرط سمی کالن را باید گذاشت.
- دستور اول درداخل پرانتز، معادل دستوری است که قبل از حلقه می‌آید.
- دستور دوم، شرط حلقه یعنی همان شرطی که برقرار بودن آن به تکرار حلقه و برقرار نبودن آن به پایان حلقه منجر می‌شود قرار می‌گیرد.
- دستور سوم، دستوری است که پس از هر بار تکرار حلقه اجرا می‌شود.
- دستور یا یک بلوک از دستورات مربوط به دستور حلقهٔ for را بدنه حلقه می‌نامند. بدنهٔ حلقه در صورت درستی شرط تکرار می‌شود.

ساختار حلقه تکرار

دستور حلقهٔ for

دستور حلقهٔ for:

for (دستور ; شرط ; دستور)
یک دستور یا یک بلوک از دستورات

دقت کنید که در پرانتز روبروی عبارت for دو سمی کالن وجود دارد که سه عبارت را از هم جدا می‌کند.

- حتی با نبود دستورات و شرط سمی کالن را باید گذاشت.
- دستور اول درداخل پرانتز، معادل دستوری است که قبل از حلقه می‌آید.
- دستور دوم، شرط حلقه یعنی همان شرطی که برقرار بودن آن به تکرار حلقه و برقرار نبودن آن به پایان حلقه منجر می‌شود قرار می‌گیرد.
- دستور سوم، دستوری است که پس از هر بار تکرار حلقه اجرا می‌شود.
- دستور یا یک بلوک از دستورات مربوط به دستور حلقهٔ for را بدنه حلقه می‌نامند. بدنهٔ حلقه در صورت درستی شرط تکرار می‌شود.

ساختار حلقه تکرار

دستور حلقهٔ for

دستور حلقهٔ for:

for (دستور ; شرط ; دستور)
یک دستور یا یک بلوک از دستورات

- دقت کنید که در پرانتز روبروی عبارت for دو سمی کالن وجود دارد که سه عبارت را از هم جدا می‌کند.
- حتی با نبود دستورات و شرط سمی کالن را باید گذاشت.
- دستور اول درداخل پرانتز، معادل دستوری است که قبل از حلقه می‌آید.
- دستور دوم، شرط حلقه یعنی همان شرطی که برقرار بودن آن به تکرار حلقه و برقرار نبودن آن به پایان حلقه منجر می‌شود قرار می‌گیرد.
- دستور سوم، دستوری است که پس از هر بار تکرار حلقه اجرا می‌شود.
- دستور یا یک بلوک از دستورات مربوط به دستور حلقهٔ for را بدنه حلقه می‌نامند. بدنهٔ حلقه در صورت درستی شرط تکرار می‌شود.

ساختار حلقه تکرار

دستور حلقهٔ for

دستور حلقهٔ for:

for (دستور ; شرط ; دستور)
یک دستور یا یک بلوک از دستورات

- دقت کنید که در پرانتز روبروی عبارت for دو سمی کالن وجود دارد که سه عبارت را از هم جدا می‌کند.
- حتی با نبود دستورات و شرط سمی کالن را باید گذاشت.
- دستور اول درداخل پرانتز، معادل دستوری است که قبل از حلقه می‌آید.
- دستور دوم، شرط حلقه یعنی همان شرطی که برقرار بودن آن به تکرار حلقه و برقرار نبودن آن به پایان حلقه منجر می‌شود قرار می‌گیرد.
- دستور سوم، دستوری است که پس از هر بار تکرار حلقه اجرا می‌شود.
- دستور یا یک بلوک از دستورات مربوط به دستور حلقهٔ for را بدنه حلقه می‌نامند. بدنهٔ حلقه در صورت درستی شرط تکرار می‌شود.

ساختار حلقه تکرار

دستور حلقهٔ for

دستور حلقهٔ for:

for (دستور ; شرط ; دستور)
یک دستور یا یک بلوک از دستورات

- دقت کنید که در پرانتز روبروی عبارت for دو سمی کالن وجود دارد که سه عبارت را از هم جدا می‌کند.
- حتی با نبود دستورات و شرط سمی کالن را باید گذاشت.
- دستور اول درداخل پرانتز، معادل دستوری است که قبل از حلقه می‌آید.
- دستور دوم، شرط حلقه یعنی همان شرطی که برقرار بودن آن به تکرار حلقه و برقرار نبودن آن به پایان حلقه منجر می‌شود قرار می‌گیرد.
- دستور سوم، دستوری است که پس از هر بار تکرار حلقه اجرا می‌شود.
- دستور یا یک بلوک از دستورات مربوط به دستور حلقهٔ for را بدنه حلقه می‌نامند. بدنهٔ حلقه در صورت درستی شرط تکرار می‌شود.

ساختار حلقه تکرار

دستور حلقهٔ for

دستور حلقهٔ for:

for (دستور ; شرط ; دستور)
یک دستور یا یک بلوک از دستورات

- دقت کنید که در پرانتز روبروی عبارت for دو سمی کالن وجود دارد که سه عبارت را از هم جدا می‌کند.
- حتی با نبود دستورات و شرط سمی کالن را باید گذاشت.
- دستور اول درداخل پرانتز، معادل دستوری است که قبل از حلقه می‌آید.
- دستور دوم، شرط حلقه یعنی همان شرطی که برقرار بودن آن به تکرار حلقه و برقرار نبودن آن به پایان حلقه منجر می‌شود قرار می‌گیرد.
- دستور سوم، دستوری است که پس از هر بار تکرار حلقه اجرا می‌شود.
- دستور یا یک بلوک از دستورات مربوط به دستور حلقهٔ for را بدنه حلقه می‌نامند. بدنهٔ حلقه در صورت درستی شرط تکرار می‌شود.

ساختار حلقه تکرار

دستور حلقهٔ for

دستور حلقهٔ for:

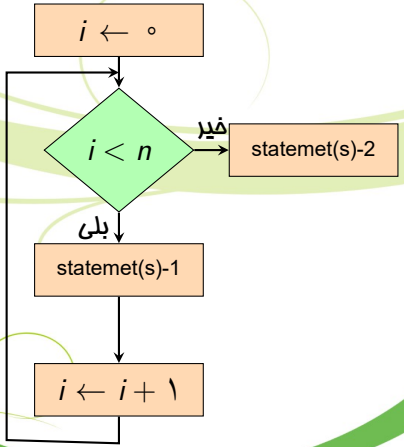
for (دستور ; شرط ; دستور)
یک دستور یا یک بلوک از دستورات

- دقت کنید که در پرانتز روبروی عبارت for دو سمی کالن وجود دارد که سه عبارت را از هم جدا می‌کند.
- حتی با نبود دستورات و شرط سمی کالن را باید گذاشت.
- دستور اول درداخل پرانتز، معادل دستوری است که قبل از حلقه می‌آید.
- دستور دوم، شرط حلقه یعنی همان شرطی که برقرار بودن آن به تکرار حلقه و برقرار نبودن آن به پایان حلقه منجر می‌شود قرار می‌گیرد.
- دستور سوم، دستوری است که پس از هر بار تکرار حلقه اجرا می‌شود.
- دستور یا یک بلوک از دستورات مربوط به دستور حلقهٔ for را بدنه حلقه می‌نامند. بدنهٔ حلقه در صورت درستی شرط تکرار می‌شود.

ساختار حلقه تکرار

دستور حلقهٔ for

```
1  for (i=0; i<n; i++)
2  {
3      statemet(s)-1
4  }
5  statemet(s)-2
```



ساختار حلقه تکرار

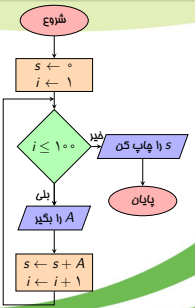
دستور حلقهٔ for

برنامه:

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int i;
6      float A, s=0;
7      for (i=1; i<=100;i++)
8      {
9          cout<<"\n Enter a number:";
10         cin>>A;
11         s+=A;
12     }
13     cout<<s;
14     return 0;
15 }
```

مثال ۱۰

برنامه‌ای بنویسید که ۱۰۰ عدد را از ورودی دریافت کند و مجموع آنها را محاسبه و چاپ کند.



ساختار حلقه تکرار

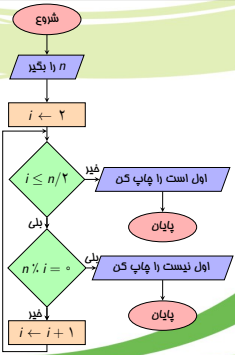
دستور حلقهٔ for

برنامه:

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int i , n;
6      cout<<"Enter n:";
7      cin >> n;
8      if (n ==1)
9      {
10         cout<<" Not Prime ";
11         return 0;
12     }
13     for (i=2 ; i <= n/2 ; i++)
14         if (n%i ==0)
15         {
16             cout<<" Not Prime ";
17             return 0;
18         }
19     cout<<" Prime ";
20     return 0;
21 }
```

مثال ۱۱

برنامه‌ای بنویسید که یک عدد صحیح بزرگ‌تر از ۱ را از ورودی دریافت کند و مشخص کند اول است یا خیر.

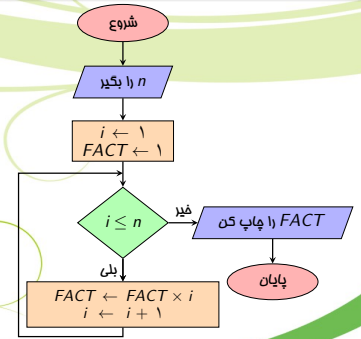


ساختار حلقه تکرار

دستور حلقهٔ for

مثال ۱۲

برنامه‌ای بنویسید که یک عدد طبیعی را از ورودی دریافت کند و فاکتوریل آن را محاسبه و چاپ کند.



برنامه:

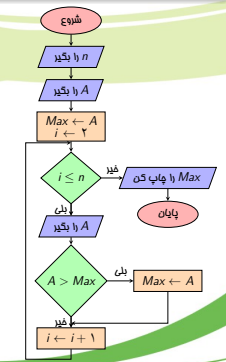
```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int i , n;
6      long int FACT =1 ;
7      cout<<"Enter n:";
8      cin >> n;
9      for ( i=1 ; i <=n ; i++)
10         FACT*=i;
11     cout<<" \n Factorial is :"<<
           FACT;
12     return 0;
13 }
```

ساختار حلقه تکرار

دستور حلقهٔ for

مثال ۱۳

برنامه‌ای بنویسید که ابتدا عدد طبیعی n و سپس n عدد را از ورودی دریافت کند و بزرگ‌ترین آنها را محاسبه و چاپ کند.



برنامه:

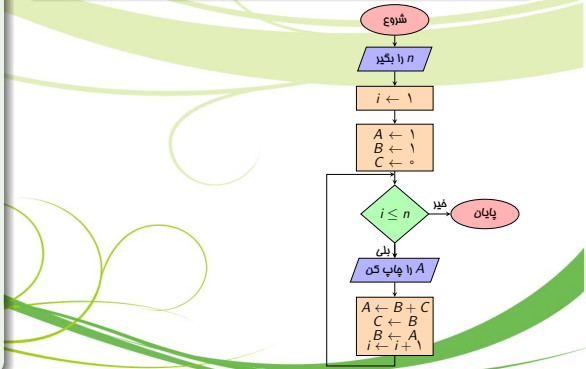
```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int i, n;
6      float A, Max;
7      cout<<"Enter n:";
8      cin>>n;
9      cout<<"Enter a number:";
10     cin>>A;
11     Max=A;
12     for(i=2;i<=n ;i++)
13     {
14         cout<<"Enter Next
               number:";
15         cin>>A;
16         if(A > Max)
17             Max=A;
18     }
19     cout<<Max;
20     return 0;
21 }
```

ساختار حلقه تکرار

دستور حلقهٔ for

مثال ۱۴

برنامه‌ای بنویسید که عدد طبیعی n را دریافت کند و n جملهٔ اول دنبالهٔ فیبوناچی را محاسبه و چاپ کند.



برنامه:

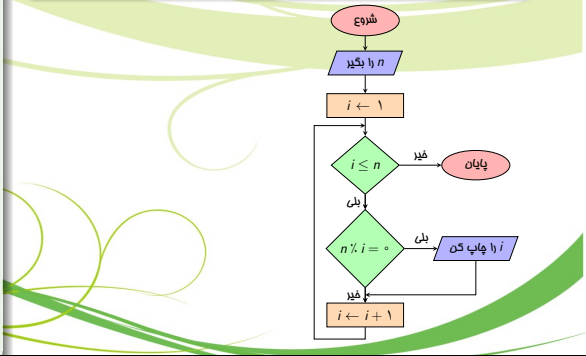
```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int i, n, A=1, B=1, C=0;
6      cout<<"Enter n:";
7      cin>>n;
8      for(i=1;i<=n;i++)
9      {
10         cout<<A<<"\n";
11         A=B+C;
12         C=B;
13         B=A;
14     }
15     return 0;
16 }
```

ساختار حلقه تکرار

دستور حلقهٔ for

مثال ۱۵

برنامه‌ای بنویسید که یک عدد صحیح را دریافت کرده و تمام مقسوم علیه‌های آن را محاسبه و چاپ کند.



برنامه:

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int i, n;
6      cout<<"Enter n:";
7      cin>>n;
8      for(i=1;i<=n;i++)
9          if(n%i==0)
10             cout<<i<<"\n";
11     return 0;
12 }
```

ساختار حلقه تکرار

دستور حلقهٔ while

دستور حلقهٔ while:

- دستور حلقهٔ while با کمی تفاوت نسبت به حلقهٔ for است.
- این دستور حلقه، صرفاً شرط تکرار حلقه را دارد و در صورت برقرار بودن شرط، دستورات حلقه تکرار می‌شود.
- به محض اشتباه بودن شرط، اجرای برنامه از اولین دستور بعد از بدنهٔ حلقه ادامه خواهد یافت.

while (شرط)
یک دستور یا یک بلوک از دستورات

ساختار حلقه تکرار

دستور حلقهٔ while

دستور حلقهٔ while:

- دستور حلقهٔ while با کمی تفاوت نسبت به حلقهٔ for است.
- این دستور حلقه، صرفاً شرط تکرار حلقه را دارد و در صورت برقرار بودن شرط، دستورات حلقه تکرار می‌شود.
- به محض اشتباه بودن شرط، اجرای برنامه از اولین دستور بعد از بدنهٔ حلقه ادامه خواهد یافت.

while (شرط)
یک دستور یا یک بلوک از دستورات

ساختار حلقه تکرار

دستور حلقهٔ while

دستور حلقهٔ while:

- دستور حلقهٔ while با کمی تفاوت نسبت به حلقهٔ for است.
- این دستور حلقه، صرفاً شرط تکرار حلقه را دارد و در صورت برقرار بودن شرط، دستورات حلقه تکرار می‌شود.
- به محض اشتباه بودن شرط اجرای برنامه از اولین دستور بعد از بدنهٔ حلقه ادامه خواهد یافت.

while (شرط)
یک دستور یا یک بلوک از دستورات

ساختار حلقه تکرار

دستور حلقهٔ while

دستور حلقهٔ while:

- دستور حلقهٔ while با کمی تفاوت نسبت به حلقهٔ for است.
- این دستور حلقه، صرفاً شرط تکرار حلقه را دارد و در صورت برقرار بودن شرط، دستورات حلقه تکرار می‌شود.
- به محض اشتباه بودن شرط اجرای برنامه از اولین دستور بعد از بدنهٔ حلقه ادامه خواهد یافت.

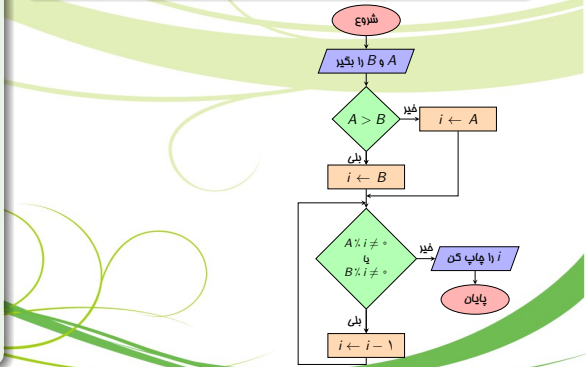
while (شرط)
یک دستور یا یک بلوک از دستورات

ساختار حلقه تکرار

دستور حلقهٔ while

مثال ۱۶

برنامه‌ای بنویسید که دو عدد صحیح را از ورودی دریافت کند و ب.م.م. آنها را محاسبه و چاپ کند.



برنامه:

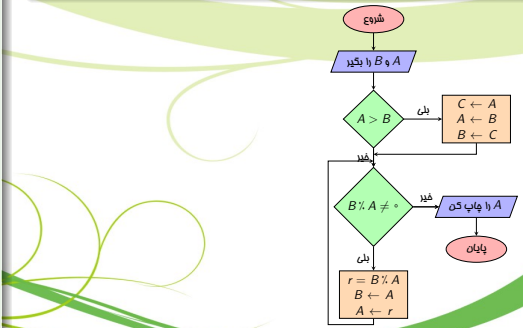
```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int A,B,i;
6      cout<<"Enter two numbers:";
7      cin>>A>>B;
8      if (A>B)
9          i=B;
10     else
11         i=A;
12     while(A%i !=0 || B%i !=0)
13         i--;
14     cout<<i;
15     return 0;
16 }
```

ساختار حلقه تکرار

دستور حلقهٔ while

مثال ۱۶

برنامه‌ای بنویسید که دو عدد صحیح را از ورودی دریافت کند و ب.م.م. آنها را محاسبه و چاپ کند.



برنامه:

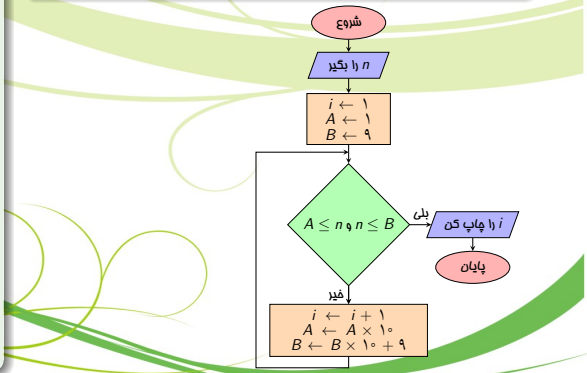
```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int r, A,B,C;
6      cout<<"enter two numbers:";
7      cin>>A>>B;
8      if (A>B)
9      {
10         C=A; A=B; B=C;
11     }
12     while(B%A !=0)
13     {
14         r=B%A; B=A; A=r;
15     }
16     cout<<A;
17     return 0;
18 }
```

ساختار حلقه تکرار

دستور حلقهٔ while

مثال ۱۷

برنامه‌ای بنویسید که یک عدد طبیعی را دریافت کرده و تعداد ارقام آن را محاسبه و چاپ کند.



برنامه:

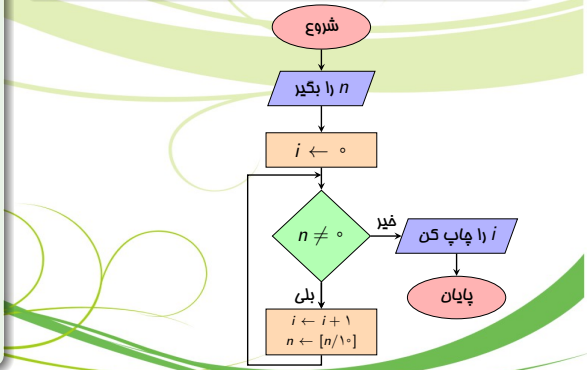
```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int i=1 , n , A=1,B=9;
6      cout<<"Enter n:";
7      cin>>n;
8      while (!(A<=n && n<=B))
9      {
10         i++;
11         A*=10;
12         B=B*10+9;
13     }
14     cout<<i;
15     return 0;
16 }
```

ساختار حلقه تکرار

دستور حلقهٔ while

مثال ۱۷

برنامه‌ای بنویسید که یک عدد طبیعی را دریافت کرده و تعداد ارقام آن را محاسبه و چاپ کند.



برنامه:

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int i=0 , n ;
6      cout<<"Enter n:";
7      cin>>n;
8      while(n!=0)
9      {
10         i++;
11         n/=10;
12     }
13     cout<<i;
14     return 0;
15 }
```

ساختار حلقه تکرار

دستور حلقهٔ while .. do

دستور حلقهٔ while .. do:

- در دستورات قبلی حلقهٔ، شرط تکرار حلقه در ابتدای حلقه بررسی می‌شد و در صورت برقرار بودن آن، بدنهٔ حلقه تکرار می‌شد.
- اما در برخی الگوریتم‌ها، شرط تکرار حلقه در انتها بررسی می‌شود. برای پیاده‌سازی چنین الگوریتم‌هایی از حلقهٔ while ... do استفاده می‌کنیم.

```
do
    یک دستور یا یک بلوک از دستورات
while( شرط);
```

- دقت کنید که در انتهای این دستور، سمی کالن نیاز است.
- عملکرد دستور while ... do مشابه دستور while است با این تفاوت که شرط حلقه در انتهای اجرای بدنه حلقه بررسی می‌شود.

ساختار حلقه تکرار

دستور حلقهٔ while .. do

دستور حلقهٔ while .. do:

- در دستورات قبلی حلقهٔ، شرط تکرار حلقه در ابتدای حلقه بررسی می‌شد و در صورت برقرار بودن آن، بدنهٔ حلقه تکرار می‌شد.
- اما در برخی الگوریتم‌ها، شرط تکرار حلقه در انتها بررسی می‌شود. برای پیاده‌سازی چنین الگوریتم‌هایی از حلقهٔ while ... do استفاده می‌کنیم.

```
do
    یک دستور یا یک بلوک از دستورات
while( شرط );
```

- دقت کنید که در انتهای این دستور، سمی کالن نیاز است.
- عملکرد دستور while ... do مشابه دستور while است با این تفاوت که شرط حلقه در انتهای اجرای بدنه حلقه بررسی می‌شود.

ساختار حلقه تکرار

دستور حلقهٔ while .. do

دستور حلقهٔ while .. do:

- در دستورات قبلی حلقهٔ، شرط تکرار حلقه در ابتدای حلقه بررسی می‌شد و در صورت برقرار بودن آن، بدنهٔ حلقه تکرار می‌شد.
- اما در برخی الگوریتم‌ها، شرط تکرار حلقه در انتها بررسی می‌شود. برای پیاده‌سازی چنین الگوریتم‌هایی از حلقهٔ while ... do استفاده می‌کنیم.

```
do
    یک دستور یا یک بلوک از دستورات
while (شرط);
```

- دقت کنید که در انتهای این دستور، سمی کالن نیاز است.
- عملکرد دستور while ... do مشابه دستور while است با این تفاوت که شرط حلقه در انتهای اجرای بدنه حلقه بررسی می‌شود.

ساختار حلقه تکرار

دستور حلقهٔ while .. do

دستور حلقهٔ while .. do:

- در دستورات قبلی حلقهٔ، شرط تکرار حلقه در ابتدای حلقه بررسی می‌شد و در صورت برقرار بودن آن، بدنهٔ حلقه تکرار می‌شد.
- اما در برخی الگوریتم‌ها، شرط تکرار حلقه در انتها بررسی می‌شود. برای پیاده‌سازی چنین الگوریتم‌هایی از حلقهٔ while ... do استفاده می‌کنیم.

```
do
    یک دستور یا یک بلوک از دستورات
while (شرط);
```

- دقت کنید که در انتهای این دستور، سمی کالن نیاز است.
- عملکرد دستور while ... do مشابه دستور while است با این تفاوت که شرط حلقه در انتهای اجرای بدنه حلقه بررسی می‌شود.

ساختار حلقه تکرار

دستور حلقهٔ while .. do

دستور حلقهٔ while .. do:

- در دستورات قبلی حلقهٔ، شرط تکرار حلقه در ابتدای حلقه بررسی می‌شد و در صورت برقرار بودن آن، بدنهٔ حلقه تکرار می‌شد.
- اما در برخی الگوریتم‌ها، شرط تکرار حلقه در انتها بررسی می‌شود. برای پیاده‌سازی چنین الگوریتم‌هایی از حلقهٔ while ... do استفاده می‌کنیم.

```
do
    یک دستور یا یک بلوک از دستورات
while (شرط);
```

- دقت کنید که در انتهای این دستور، سمی کالن نیاز است.
- عملکرد دستور while ... do مشابه دستور while است با این تفاوت که شرط حلقه در انتهای اجرای بدنه حلقه بررسی می‌شود.

ساختار حلقه تکرار

دستور حلقهٔ do .. while

مثال:

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5       float number, sum = 0.0;
6       do {
7            cout<<"Enter a number: ";
8            cin>>number;
9            sum += number;
10      }while(number != 0.0);
11      cout<<"Total sum = "<<sum;
12      return 0;
13 }
```

آرایه‌ها

آرایه‌ها

- آرایه‌ها نیز مشابه سایر متغیرها باید قبل از استفاده تعریف شوند.
- اضافه بر نوع آرایه، باید فضای مورد نیاز برای آرایه نیز مشخص شود. مثال: `int A[100];`
- می‌توان چند آرایه را که از یک نوع هستند در یک دستور تعریف قرار داد و با کاما از یکدیگر جدا کرد. همچنین امکان تعریف متغیرهای معمول و آرایه‌ها در یک دستور تعریف امکان‌پذیر است.
- تعداد درایه‌های یک آرایه باید یک عدد طبیعی باشد و امکان متغیر بودن این مقدار نیست.
- اندیس آرایه از صفر شروع می‌شود، یعنی $A[3]$ درایهٔ چهارمِ آرایهٔ A است.
- برای ارجاع به درایهٔ یک آرایه، بررسی نمی‌شود که آیا درایه‌ای با این شماره در آرایه وجود دارد یا خیر. لذا برنامه‌نویس باید مواظب باشد در برنامه چنین اتفاقی نیفتد. این موضوع یکی از کانون‌های خطای منطقی در برنامه‌نویسی است.

آرایه‌ها

آرایه‌ها

- آرایه‌ها نیز مشابه سایر متغیرها باید قبل از استفاده تعریف شوند.
- اضافه بر نوع آرایه، باید فضای مورد نیاز برای آرایه نیز مشخص شود. مثال: `int A[100];`
- می‌توان چند آرایه را که از یک نوع هستند در یک دستور تعریف قرار داد و با کاما از یکدیگر جدا کرد. همچنین امکان تعریف متغیرهای معمول و آرایه‌ها در یک دستور تعریف امکان‌پذیر است.
- تعداد درایه‌های یک آرایه باید یک عدد طبیعی باشد و امکان متغیر بودن این مقدار نیست.
- اندیس آرایه از صفر شروع می‌شود، یعنی $A[3]$ درایهٔ چهارمِ آرایهٔ A است.
- برای ارجاع به درایهٔ یک آرایه، بررسی نمی‌شود که آیا درایه‌ای با این شماره در آرایه وجود دارد یا خیر. لذا برنامه‌نویس باید مواظب باشد در برنامه چنین اتفاقی نیفتد. این موضوع یکی از کانون‌های خطای منطقی در برنامه‌نویسی است.

آرایه‌ها

آرایه‌ها

- آرایه‌ها نیز مشابه سایر متغیرها باید قبل از استفاده تعریف شوند.
- اضافه بر نوع آرایه، باید فضای مورد نیاز برای آرایه نیز مشخص شود. مثال: `int A[100];`
- می‌توان چند آرایه را که از یک نوع هستند در یک دستور تعریف قرار داد و با کاما از یکدیگر جدا کرد. همچنین امکان تعریف متغیرهای معمول و آرایه‌ها در یک دستور تعریف امکان‌پذیر است.
- تعداد درایه‌های یک آرایه باید یک عدد طبیعی باشد و امکان متغیر بودن این مقدار نیست.
- اندیس آرایه از صفر شروع می‌شود، یعنی $A[3]$ درایهٔ چهارمِ آرایهٔ A است.
- برای ارجاع به درایهٔ یک آرایه، بررسی نمی‌شود که آیا درایه‌ای با این شماره در آرایه وجود دارد یا خیر. لذا برنامه‌نویس باید مواظب باشد در برنامه چنین اتفاقی نیفتد. این موضوع یکی از کانون‌های خطای منطقی در برنامه‌نویسی است.

آرایه‌ها

آرایه‌ها

- آرایه‌ها نیز مشابه سایر متغیرها باید قبل از استفاده تعریف شوند.
- اضافه بر نوع آرایه، باید فضای مورد نیاز برای آرایه نیز مشخص شود. مثال: `int A[100];`
- می‌توان چند آرایه را که از یک نوع هستند در یک دستور تعریف قرار داد و با کاما از یکدیگر جدا کرد. همچنین امکان تعریف متغیرهای معمول و آرایه‌ها در یک دستور تعریف امکان‌پذیر است.
- تعداد درایه‌های یک آرایه باید یک عدد طبیعی باشد و امکان متغیر بودن این مقدار نیست.
- اندیس آرایه از صفر شروع می‌شود، یعنی $A[3]$ درایهٔ چهارمِ آرایهٔ A است.
- برای ارجاع به درایهٔ یک آرایه، بررسی نمی‌شود که آیا درایه‌ای با این شماره در آرایه وجود دارد یا خیر. لذا برنامه‌نویس باید مواظب باشد در برنامه چنین اتفاقی نیفتد. این موضوع یکی از کانون‌های خطای منطقی در برنامه‌نویسی است.

آرایه‌ها

آرایه‌ها

- آرایه‌ها نیز مشابه سایر متغیرها باید قبل از استفاده تعریف شوند.
- اضافه بر نوع آرایه، باید فضای مورد نیاز برای آرایه نیز مشخص شود. مثال: `int A[100];`
- می‌توان چند آرایه را که از یک نوع هستند در یک دستور تعریف قرار داد و با کاما از یکدیگر جدا کرد. همچنین امکان تعریف متغیرهای معمول و آرایه‌ها در یک دستور تعریف امکان‌پذیر است.
- تعداد درایه‌های یک آرایه باید یک عدد طبیعی باشد و امکان متغیر بودن این مقدار نیست.
- اندیس آرایه از صفر شروع می‌شود، یعنی $A[3]$ درایهٔ چهارمِ آرایهٔ A است.
- برای ارجاع به درایهٔ یک آرایه، بررسی نمی‌شود که آیا درایه‌ای با این شماره در آرایه وجود دارد یا خیر. لذا برنامه‌نویس باید مواظب باشد در برنامه چنین اتفاقی نیفتد. این موضوع یکی از کانون‌های خطای منطقی در برنامه‌نویسی است.

آرایه‌ها

آرایه‌ها

- آرایه‌ها نیز مشابه سایر متغیرها باید قبل از استفاده تعریف شوند.
- اضافه بر نوع آرایه، باید فضای مورد نیاز برای آرایه نیز مشخص شود. مثال: `int A[100];`
- می‌توان چند آرایه را که از یک نوع هستند در یک دستور تعریف قرار داد و با کاما از یکدیگر جدا کرد. همچنین امکان تعریف متغیرهای معمول و آرایه‌ها در یک دستور تعریف امکان‌پذیر است.
- تعداد درایه‌های یک آرایه باید یک عدد طبیعی باشد و امکان متغیر بودن این مقدار نیست.
- اندیس آرایه از صفر شروع می‌شود، یعنی $A[3]$ درایهٔ چهارمِ آرایهٔ A است.
- برای ارجاع به درایهٔ یک آرایه، بررسی نمی‌شود که آیا درایه‌ای با این شماره در آرایه وجود دارد یا خیر. لذا برنامه‌نویس باید مواظب باشد در برنامه چنین اتفاقی نیفتد. این موضوع یکی از کانون‌های خطای منطقی در برنامه‌نویسی است.

مثال ۱۸

الگوریتمی بنویسید که ابتدا عدد n و سپس n عدد را بگیرد و آن‌ها را به ترتیب عکس داده شده، چاپ کند.

الگوریتم:

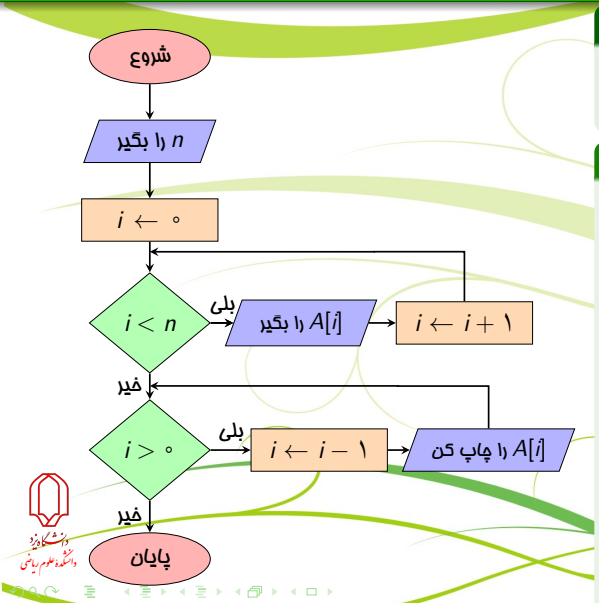
```
۱  شروع
۲  n را بگیر
۳  i ← ۰
۴  اگر i < n آنگاه
۵  |  A[i] را بگیر
۶  |  i ← i + ۱
۷  |  به مرحلهٔ ۴ برو
۸  پایان اگر
۹  اگر i > ۰ آنگاه
۱۰ |  i ← i − ۱
۱۱ |  A[i] را چاپ کن
۱۲ |  به مرحلهٔ ۹ برو
۱۳ پایان اگر
۱۴ پایان
```

مثال ۱۸

الگوریتمی بنویسید که ابتدا عدد n و سپس n عدد را بگیرد و آن‌ها را به ترتیب عکس داده شده، چاپ کند.

الگوریتم:

۱ شروع
۲ n را بگیر
۳ $i \leftarrow 0$
۴ اگر $i < n$ آنگاه
۵ | $A[i]$ را بگیر
۶ | $i \leftarrow i + 1$
۷ | به مرحلهٔ ۴ برو
۸ پایان اگر
۹ اگر $i > 0$ آنگاه
۱۰ | $i \leftarrow i - 1$
۱۱ | $A[i]$ را چاپ کن
۱۲ | به مرحلهٔ ۹ برو
۱۳ پایان اگر
۱۴ پایان



مثال ۱۸

الگوریتمی بنویسید که ابتدا عدد n و سپس n عدد را بگیرد و آن‌ها را به ترتیب عکس داده شده، چاپ کند.

الگوریتم:

۱ شروع
۲ n را بگیر
۳ $i \leftarrow 0$
۴ اگر $i < n$ آنگاه
۵ | $A[i]$ را بگیر
۶ | $i \leftarrow i + 1$
۷ | به مرحلهٔ ۴ برو
۸ پایان اگر
۹ اگر $i > 0$ آنگاه
۱۰ | $i \leftarrow i - 1$
۱۱ | $A[i]$ را چاپ کن
۱۲ | به مرحلهٔ ۹ برو
۱۳ پایان اگر
۱۴ پایان

برنامه:

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int i,n;
6      float A[200];
7      cout<<"Enter An Integer
           (<=200):";
8      cin>>n;
9      for (i=0; i<n;i++)
10     {
11         cout<<"\n Enter Next
               number:";
12         cin>>A[i];
13     }
14     while(i>0)
15     {
16         i--;
17         cout<<A[i]<<",";
18     }
19     return 0;
20 }
```

مثال ۱۹

الگوریتم و فلوچارتی ارائه کنید که ابتدا n و سپس n نمره را دریافت کرده و تعداد نمرات بالاتر از میانگین کلاس را چاپ کند.

الگوریتم:

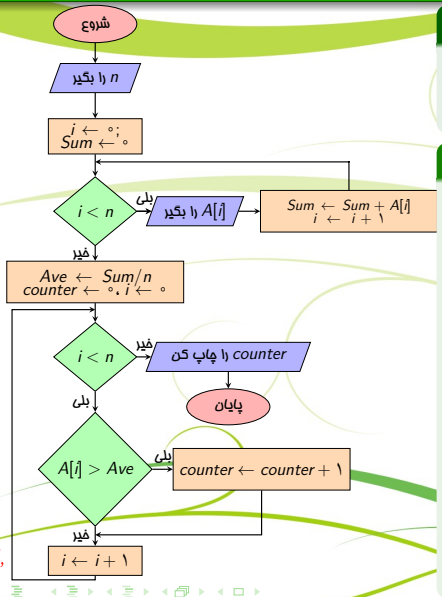
۱ شروع
۲ n را بگیر
۳ $i \leftarrow 0$، $Sum \leftarrow 0$
۴ اگر $i < n$ آنگاه
۵ | $A[i]$ را بگیر
۶ | $Sum \leftarrow Sum + A[i]$
۷ | $i \leftarrow i + 1$
۸ | به مرحلهٔ ۴ برو
۹ پایان اگر
۱۰ $Ave \leftarrow Sum/n$
۱۱ $i \leftarrow 0$، $counter \leftarrow 0$
۱۲ اگر $i < n$ آنگاه
۱۳ | اگر $A[i] > Ave$ آنگاه
۱۴ | | $counter \leftarrow counter + 1$
۱۵ | پایان اگر
۱۶ | $i \leftarrow i + 1$
۱۷ | به مرحلهٔ ۱۲ برو
۱۸ پایان اگر
۱۹ $counter$ را چاپ کن
۲۰ پایان

مثال ۱۹

الگوریتم و فلوچارتی ارائه کنید که ابتدا n و سپس n نمره را دریافت کرده و تعداد نمرات بالاتر از میانگین کلاس را چاپ کند.

الگوریتم:

۱ شروع
۲ n را بگیر
۳ $i \leftarrow ۰$، $Sum \leftarrow ۰$
۴ اگر $i < n$ آنگاه
۵ | $A[i]$ را بگیر
۶ | $Sum \leftarrow Sum + A[i]$
۷ | $i \leftarrow i + ۱$
۸ | به مرحلهٔ ۴ برو
۹ پایان اگر
۱۰ $Ave \leftarrow Sum/n$
۱۱ $i \leftarrow ۰$، $counter \leftarrow ۰$
۱۲ اگر $i < n$ آنگاه
۱۳ | اگر $A[i] > Ave$ آنگاه
۱۴ | | $counter \leftarrow counter + ۱$
۱۵ | پایان اگر
۱۶ | $i \leftarrow i + ۱$
۱۷ | به مرحلهٔ ۱۲ برو
۱۸ پایان اگر
۱۹ $counter$ را چاپ کن
۲۰ پایان



مثال ۱۹

الگوریتم و فلوچارتی ارائه کنید که ابتدا n و سپس n نمره را دریافت کرده و تعداد نمرات بالاتر از میانگین کلاس را چاپ کند.

الگوریتم:

۱ شروع
۲ n را بگیر
۳ $i \leftarrow 0$، $Sum \leftarrow 0$
۴ اگر $i < n$ آنگاه
۵ | $A[i]$ را بگیر
۶ | $Sum \leftarrow Sum + A[i]$
۷ | $i \leftarrow i + 1$
۸ | به مرحلهٔ ۴ برو
۹ پایان اگر
۱۰ $Ave \leftarrow Sum/n$
۱۱ $i \leftarrow 0$، $counter \leftarrow 0$
۱۲ اگر $i < n$ آنگاه
۱۳ | اگر $A[i] > Ave$ آنگاه
۱۴ | | $counter \leftarrow counter + 1$
۱۵ | پایان اگر
۱۶ | $i \leftarrow i + 1$
۱۷ | به مرحلهٔ ۱۲ برو
۱۸ پایان اگر
۱۹ $counter$ را چاپ کن
۲۰ پایان

برنامه:

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int i, n, counter;
6      float A[200], Sum, Ave;
7      cout<<"Enter An Integer
           (<=200):";
8      cin>>n;
9      for (i=0; i<n;i++)
10     {
11         cout<<"\n Enter Next
               number:";
12         cin>>A[i];
13         Sum+=A[i];
14     }
15     Ave=Sum/n;
16     counter=0;
17     for (i=0; i<n; i++)
18     {
19         if (A[i]>Ave)
20             counter++;
21     }
22     cout<<"# of grades above
           average:"<<counter;
23     return 0;
24 }
```

آرایه‌ها

آرایه‌های دوبعدی

- آرایه‌های دوبعدی، روش تعریف و استفاده مشابه آرایه‌های یک بعدی است. تنها تفاوت آنها این است که در تعریف آرایه، باید تعداد سطر و ستون‌های آنها مشخص شود.
- مثال: برای تعریف آرایهٔ دوبعدی Matrix از نوع اعشاری و با ۳ سطر و ۷ ستون، دستور زیر استفاده می‌شود:

```
float Matrix [3][7];
```

- برای دسترسی به هر درایه باید شماره سطر و ستون آنها بیان شود. برای ارجاع به درایهٔ سطر اول و ستون چهارم از Matrix [0][3] استفاده می‌کنیم.

http://pws.yazd.ac.ir/farshi/ محمّد فرشی- دانشگاه یزد

آرایه‌ها

آرایه‌های دوبعدی

- آرایه‌های دوبعدی، روش تعریف و استفاده مشابه آرایه‌های یک بعدی است. تنها تفاوت آنها این است که در تعریف آرایه، باید تعداد سطر و ستون‌های آنها مشخص شود.
- **مثال: برای تعریف آرایهٔ دوبعدی Matrix از نوع اعشاری و با ۳ سطر و ۷ ستون، دستور زیر استفاده می‌شود:**

```
float Matrix [3][7];
```

- برای دسترسی به هر درایه باید شماره سطر و ستون آنها بیان شود. برای ارجاع به درایهٔ سطر اول و ستون چهارم از Matrix [0][3] استفاده می‌کنیم.

آرایه‌ها

آرایه‌های دوبعدی

- آرایه‌های دوبعدی، روش تعریف و استفاده مشابه آرایه‌های یک بعدی است. تنها تفاوت آنها این است که در تعریف آرایه، باید تعداد سطر و ستون‌های آنها مشخص شود.
- **مثال: برای تعریف آرایهٔ دوبعدی Matrix از نوع اعشاری و با ۳ سطر و ۷ ستون، دستور زیر استفاده می‌شود:**

```
float Matrix [3][7];
```

- برای دسترسی به هر درایه باید شماره سطر و ستون آنها بیان شود. برای ارجاع به درایهٔ سطر اول و ستون چهارم از Matrix [0][3] استفاده می‌کنیم.


باآرزوی موفقیت
وبهروزی
mfarshi@yazd.ac.ir