اصول برنامه‌نویسی کامپیوتر

اشاره‌گرها و تخصیص حافظهٔ پویا

محمّد فرشی

آزمایشگاه الگوریتم های ترکیباتی و هندسی - دانشکدهٔ علوم ریاضی - دانشگاه یزد

۱۴۰۳-۱

اشاره‌گرها

مقدمه

- زبان C++ به عنوان قدرتمندترین زبان برنامه‌نویسی سیستم
- اشاره‌گرها، یکی از مهمترین این قابلیت‌ها است.
- امکان دسترسی به تک تک بایت‌های حافظه اصلی کامپیوتر
- اهمیت اشاره‌گرها: مقایسهٔ دانستن آدرس شخص با دانستن محل منزل به صورت تجربی

اشاره‌گرها

مقدمه

- زبان C++ به عنوان قدرتمندترین زبان برنامه‌نویسی سیستم
- اشاره‌گرها، یکی از مهمترین این قابلیت‌ها است.
- امکان دسترسی به تک تک بایت‌های حافظه اصلی کامپیوتر
- اهمیت اشاره‌گرها: مقایسهٔ دانستن آدرس شخص با دانستن محل منزل به صورت تجربی

اشاره‌گرها

مقدمه

- زبان C++ به عنوان قدرتمندترین زبان برنامه‌نویسی سیستم
- اشاره‌گرها، یکی از مهمترین این قابلیت‌ها است.
- امکان دسترسی به تک تک بایت‌های حافظه اصلی کامپیوتر
- اهمیت اشاره‌گرها: مقایسهٔ دانستن آدرس شخص با دانستن محل منزل به صورت تجربی

اشاره‌گرها

مقدمه

- زبان C++ به عنوان قدرتمندترین زبان برنامه‌نویسی سیستم
- اشاره‌گرها، یکی از مهمترین این قابلیت‌ها است.
- امکان دسترسی به تک تک بایت‌های حافظه اصلی کامپیوتر
- اهمیت اشاره‌گرها: مقایسهٔ دانستن آدرس شخص با دانستن محل منزل به صورت تجربی

اشاره‌گرها

اشاره‌گر چیست؟

- حافظهٔ اصلی کامپیوتر را می‌توان به صورت یک آرایه در نظر گرفت.
- هر بایت در حافظه کامپیوتر یک اندیس بین ۰ تا اندازه حافظهٔ اصلی کامپیوتر دارد.
- سرعت دسترسی به بایت‌ها در محل‌های مختلف حافظه اصلی یکسان است و ارتباط و وابستگی به محل آن بایت در حافظهٔ اصلی کامپیوتر ندارد.
- دلیل نام‌گذاری حافظه اصلی به نام RAM نیز دقیقاً همین ساختار دسترسی به حافظهٔ اصلی کامپیوتر است.
- اندیس هر بایت در حافظهٔ اصلی را اصطلاحاً آدرس آن بایت می‌نامیم.
- اشاره‌گر به یک متغیر: آدرس اولین بایت حافظهٔ آن متغیر.
- تفاوت کامپیوترهای ۳۲ بیتی و ۶۴ بیتی.

اشاره‌گرها

اشاره‌گر چیست؟

- حافظهٔ اصلی کامپیوتر را می‌توان به صورت یک آرایه در نظر گرفت.
- هر بایت در حافظه کامپیوتر یک اندیس بین ۰ تا اندازه حافظهٔ اصلی کامپیوتر دارد.
- سرعت دسترسی به بایت‌ها در محل‌های مختلف حافظه اصلی یکسان است و ارتباط و وابستگی به محل آن بایت در حافظهٔ اصلی کامپیوتر ندارد.
- دلیل نام‌گذاری حافظه اصلی به نام RAM نیز دقیقاً همین ساختار دسترسی به حافظهٔ اصلی کامپیوتر است.
- اندیس هر بایت در حافظهٔ اصلی را اصطلاحاً آدرس آن بایت می‌نامیم.
- اشاره‌گر به یک متغیر: آدرس اولین بایت حافظهٔ آن متغیر.
- تفاوت کامپیوترهای ۳۲ بیتی و ۶۴ بیتی.

اشاره‌گرها

اشاره‌گر چیست؟

- حافظهٔ اصلی کامپیوتر را می‌توان به صورت یک آرایه در نظر گرفت.
- هر بایت در حافظه کامپیوتر یک اندیس بین ۰ تا اندازه حافظهٔ اصلی کامپیوتر دارد.
- سرعت دسترسی به بایت‌ها در محل‌های مختلف حافظه اصلی یکسان است و ارتباط و وابستگی به محل آن بایت در حافظهٔ اصلی کامپیوتر ندارد.
- دلیل نام‌گذاری حافظه اصلی به نام RAM نیز دقیقاً همین ساختار دسترسی به حافظهٔ اصلی کامپیوتر است.
- اندیس هر بایت در حافظهٔ اصلی را اصطلاحاً آدرس آن بایت می‌نامیم.
- اشاره‌گر به یک متغیر: آدرس اولین بایت حافظهٔ آن متغیر.
- تفاوت کامپیوترهای ۳۲ بیتی و ۶۴ بیتی.

اشاره‌گرها

اشاره‌گر چیست؟

- حافظهٔ اصلی کامپیوتر را می‌توان به صورت یک آرایه در نظر گرفت.
- هر بایت در حافظه کامپیوتر یک اندیس بین ۰ تا اندازه حافظهٔ اصلی کامپیوتر دارد.
- سرعت دسترسی به بایت‌ها در محل‌های مختلف حافظه اصلی یکسان است و ارتباط و وابستگی به محل آن بایت در حافظهٔ اصلی کامپیوتر ندارد.
- دلیل نام‌گذاری حافظه اصلی به نام RAM نیز دقیقاً همین ساختار دسترسی به حافظهٔ اصلی کامپیوتر است.
- اندیس هر بایت در حافظهٔ اصلی را اصطلاحاً آدرس آن بایت می‌نامیم.
- اشاره‌گر به یک متغیر: آدرس اولین بایت حافظهٔ آن متغیر.
- تفاوت کامپیوترهای ۳۲ بیتی و ۶۴ بیتی.

اشاره‌گرها

اشاره‌گر چیست؟

- حافظهٔ اصلی کامپیوتر را می‌توان به صورت یک آرایه در نظر گرفت.
- هر بایت در حافظه کامپیوتر یک اندیس بین ۰ تا اندازه حافظهٔ اصلی کامپیوتر دارد.
- سرعت دسترسی به بایت‌ها در محل‌های مختلف حافظه اصلی یکسان است و ارتباط و وابستگی به محل آن بایت در حافظهٔ اصلی کامپیوتر ندارد.
- دلیل نام‌گذاری حافظه اصلی به نام RAM نیز دقیقاً همین ساختار دسترسی به حافظهٔ اصلی کامپیوتر است.
- اندیس هر بایت در حافظهٔ اصلی را اصطلاحاً آدرس آن بایت می‌نامیم.
- اشاره‌گر به یک متغیر: آدرس اولین بایت حافظهٔ آن متغیر.
- تفاوت کامپیوترهای ۳۲ بیتی و ۶۴ بیتی.

اشاره‌گرها

اشاره‌گر چیست؟

- حافظهٔ اصلی کامپیوتر را می‌توان به صورت یک آرایه در نظر گرفت.
- هر بایت در حافظه کامپیوتر یک اندیس بین ۰ تا اندازه حافظهٔ اصلی کامپیوتر دارد.
- سرعت دسترسی به بایت‌ها در محل‌های مختلف حافظه اصلی یکسان است و ارتباط و وابستگی به محل آن بایت در حافظهٔ اصلی کامپیوتر ندارد.
- دلیل نام‌گذاری حافظه اصلی به نام RAM نیز دقیقاً همین ساختار دسترسی به حافظهٔ اصلی کامپیوتر است.
- اندیس هر بایت در حافظهٔ اصلی را اصطلاحاً آدرس آن بایت می‌نامیم.
- اشاره‌گر به یک متغیر: آدرس اولین بایت حافظهٔ آن متغیر.
- تفاوت کامپیوترهای ۳۲ بیتی و ۶۴ بیتی.

اشاره‌گرها

اشاره‌گر چیست؟

- حافظهٔ اصلی کامپیوتر را می‌توان به صورت یک آرایه در نظر گرفت.
- هر بایت در حافظه کامپیوتر یک اندیس بین ۰ تا اندازه حافظهٔ اصلی کامپیوتر دارد.
- سرعت دسترسی به بایت‌ها در محل‌های مختلف حافظه اصلی یکسان است و ارتباط و وابستگی به محل آن بایت در حافظهٔ اصلی کامپیوتر ندارد.
- دلیل نام‌گذاری حافظه اصلی به نام RAM نیز دقیقاً همین ساختار دسترسی به حافظهٔ اصلی کامپیوتر است.
- اندیس هر بایت در حافظهٔ اصلی را اصطلاحاً آدرس آن بایت می‌نامیم.
- اشاره‌گر به یک متغیر: آدرس اولین بایت حافظهٔ آن متغیر.
- تفاوت کامپیوترهای ۳۲ بیتی و ۶۴ بیتی.

اشاره‌گرها

انواع اشاره‌گرها

- یا اصولاً نیاز به انواع اشاره‌گرها وجود دارد یا خیر؟ از یک دید، جواب منفی است، زیرا اشاره‌گر به هر متغیر، یک آدرس یا اندیس در حافظه کامپیوتر است و لذا تفاوتی بین آدرس انواع مختلف متغیرها وجود ندارد.
- اما، در زبان C++ ، انواع مختلف اشاره‌گرها وجود دارد.
- زیرا، صرفاً داشتن آدرس محل شروع ذخیره‌سازی متغیر، کافی نیست.
- لازم است تعداد بایت‌هایی که آن متغیر اشغال می‌کند نیز مشخص باشد تا امکان استفاده از اشاره‌گر به متغیر باشد.
- اشاره‌گر به یک متغیر، از طرفی یک مقدار است که محل شروع ذخیره‌سازی متغیر در حافظه است و از طرف دیگر، یک نوع دارد که مشخص می‌کند، فضای استفاده شده توسط آن متغیر، چند بایت است.
- به ازای هر نوع متغیر، یک نوع متغیری اشاره‌گر به آن متغیر هم وجود دارد.

اشاره‌گرها

انواع اشاره‌گرها

- یا اصولاً نیاز به انواع اشاره‌گرها وجود دارد یا خیر؟ از یک دید، جواب منفی است، زیرا اشاره‌گر به هر متغیر، یک آدرس یا اندیس در حافظه کامپیوتر است و لذا تفاوتی بین آدرس انواع مختلف متغیرها وجود ندارد.
- اما، در زبان C++ ، انواع مختلف اشاره‌گرها وجود دارد.
- زیرا، صرفاً داشتن آدرس محل شروع ذخیره‌سازی متغیر، کافی نیست.
- لازم است تعداد بایت‌هایی که آن متغیر اشغال می‌کند نیز مشخص باشد تا امکان استفاده از اشاره‌گر به متغیر باشد.
- اشاره‌گر به یک متغیر، از طرفی یک مقدار است که محل شروع ذخیره‌سازی متغیر در حافظه است و از طرف دیگر، یک نوع دارد که مشخص می‌کند، فضای استفاده شده توسط آن متغیر، چند بایت است.
- به ازای هر نوع متغیر، یک نوع متغیری اشاره‌گر به آن متغیر هم وجود دارد.

اشاره‌گرها

انواع اشاره‌گرها

- یا اصولاً نیاز به انواع اشاره‌گرها وجود دارد یا خیر؟ از یک دید، جواب منفی است، زیرا اشاره‌گر به هر متغیر، یک آدرس یا اندیس در حافظه کامپیوتر است و لذا تفاوتی بین آدرس انواع مختلف متغیرها وجود ندارد.
- اما، در زبان C++ ، انواع مختلف اشاره‌گرها وجود دارد.
- زیرا، صرفاً داشتن آدرس محل شروع ذخیره‌سازی متغیر، کافی نیست.
- لازم است تعداد بایت‌هایی که آن متغیر اشغال می‌کند نیز مشخص باشد تا امکان استفاده از اشاره‌گر به متغیر باشد.
- اشاره‌گر به یک متغیر، از طرفی یک مقدار است که محل شروع ذخیره‌سازی متغیر در حافظه است و از طرف دیگر، یک نوع دارد که مشخص می‌کند، فضای استفاده شده توسط آن متغیر، چند بایت است.
- به ازای هر نوع متغیر، یک نوع متغیری اشاره‌گر به آن متغیر هم وجود دارد.

اشاره‌گرها

انواع اشاره‌گرها

- یا اصولاً نیاز به انواع اشاره‌گرها وجود دارد یا خیر؟ از یک دید، جواب منفی است، زیرا اشاره‌گر به هر متغیر، یک آدرس یا اندیس در حافظه کامپیوتر است و لذا تفاوتی بین آدرس انواع مختلف متغیرها وجود ندارد.
- اما، در زبان C++ ، انواع مختلف اشاره‌گرها وجود دارد.
- زیرا، صرفاً داشتن آدرس محل شروع ذخیره‌سازی متغیر، کافی نیست.
- لازم است تعداد بایت‌هایی که آن متغیر اشغال می‌کند نیز مشخص باشد تا امکان استفاده از اشاره‌گر به متغیر باشد.
- اشاره‌گر به یک متغیر، از طرفی یک مقدار است که محل شروع ذخیره‌سازی متغیر در حافظه است و از طرف دیگر، یک نوع دارد که مشخص می‌کند، فضای استفاده شده توسط آن متغیر، چند بایت است.
- به ازای هر نوع متغیر، یک نوع متغیری اشاره‌گر به آن متغیر هم وجود دارد.

اشاره‌گرها

انواع اشاره‌گرها

- یا اصولاً نیاز به انواع اشاره‌گرها وجود دارد یا خیر؟ از یک دید، جواب منفی است، زیرا اشاره‌گر به هر متغیر، یک آدرس یا اندیس در حافظه کامپیوتر است و لذا تفاوتی بین آدرس انواع مختلف متغیرها وجود ندارد.
- اما، در زبان C++ ، انواع مختلف اشاره‌گرها وجود دارد.
- زیرا، صرفاً داشتن آدرس محل شروع ذخیره‌سازی متغیر، کافی نیست.
- لازم است تعداد بایت‌هایی که آن متغیر اشغال می‌کند نیز مشخص باشد تا امکان استفاده از اشاره‌گر به متغیر باشد.
- اشاره‌گر به یک متغیر، از طرفی یک مقدار است که محل شروع ذخیره‌سازی متغیر در حافظه است و از طرف دیگر، یک نوع دارد که مشخص می‌کند، فضای استفاده شده توسط آن متغیر، چند بایت است.
- به ازای هر نوع متغیر، یک نوع متغیری اشاره‌گر به آن متغیر هم وجود دارد.

اشاره‌گرها

انواع اشاره‌گرها

- یا اصولاً نیاز به انواع اشاره‌گرها وجود دارد یا خیر؟ از یک دید، جواب منفی است، زیرا اشاره‌گر به هر متغیر، یک آدرس یا اندیس در حافظه کامپیوتر است و لذا تفاوتی بین آدرس انواع مختلف متغیرها وجود ندارد.
- اما، در زبان C++ ، انواع مختلف اشاره‌گرها وجود دارد.
- زیرا، صرفاً داشتن آدرس محل شروع ذخیره‌سازی متغیر، کافی نیست.
- لازم است تعداد بایت‌هایی که آن متغیر اشغال می‌کند نیز مشخص باشد تا امکان استفاده از اشاره‌گر به متغیر باشد.
- اشاره‌گر به یک متغیر، از طرفی یک مقدار است که محل شروع ذخیره‌سازی متغیر در حافظه است و از طرف دیگر، یک نوع دارد که مشخص می‌کند، فضای استفاده شده توسط آن متغیر، چند بایت است.
- به ازای هر نوع متغیر، یک نوع متغیری اشاره‌گر به آن متغیر هم وجود دارد.

اشاره‌گرها

تعریف متغیر اشاره‌گری:

- کافی است در تعریف هر متغیر، یک علامت ستاره قبل از نام منغیر (و بعد از نوع آن)اضافه کنیم تا نوع اشاره‌گری مربوط به آن متغیر ایجاد شود.
- دستور int * A; متغیری به نام A ایجاد می‌کند که می‌تواند اشاره‌گر به یک متغیر از نوع int را نگهداری کند.
- این کار برای هر نوع از پیش تعریف شده در C++ و یا هر نوع متغیری که توسط برنامه‌نویس ایجاد شود، امکان‌پذیر است و محدودیتی وجود ندارد.

اشاره‌گرها

تعریف متغیر اشاره‌گری:

- کافی است در تعریف هر متغیر، یک علامت ستاره قبل از نام منغیر (و بعد از نوع آن)اضافه کنیم تا نوع اشاره‌گری مربوط به آن متغیر ایجاد شود.
- دستور int * A; متغیری به نام A ایجاد می‌کند که می‌تواند اشاره‌گر به یک متغیر از نوع int را نگهداری کند.
- این کار برای هر نوع از پیش تعریف شده در C++ و یا هر نوع متغیری که توسط برنامه‌نویس ایجاد شود، امکان‌پذیر است و محدودیتی وجود ندارد.

اشاره‌گرها

تعریف متغیر اشاره‌گری:

- کافی است در تعریف هر متغیر، یک علامت ستاره قبل از نام منغیر (و بعد از نوع آن)اضافه کنیم تا نوع اشاره‌گری مربوط به آن متغیر ایجاد شود.
- دستور int * A; متغیری به نام A ایجاد می‌کند که می‌تواند اشاره‌گر به یک متغیر از نوع int را نگهداری کند.
- این کار برای هر نوع از پیش تعریف شده در C++ و یا هر نوع متغیری که توسط برنامه‌نویس ایجاد شود، امکان‌پذیر است و محدودیتی وجود ندارد.

اشاره‌گرها

عملیات روی اشاره‌گرها:

- محل ذخیره‌سازی متغیر در حافظه توسط برنامه‌نویس، قابل تعیین نیست.
- روش بدست آوردن آدرس یا اشاره‌گر به یک متغیر چیست؟ عملگر &
- به عنوان مثال، اگر x یک متغیر باشد، عبارت &x، اشاره‌گر به متغیر x را به برنامه‌نویس می‌دهد.
- به عنوان مثال، برای ذخیره کردن اشاره‌گر به متغیر x در متغیر A که در بالا تعریف شده است، کافی است از دستور A=&x; استفاده کنیم.
- با داشتن اشاره‌گر به یک متغیر، چگونه می‌توان به آن متغیر دسترسی داشت؟ عملگر *
- دستور A=5; *x=5; کاری مشابه دستور انجام می‌دهد.
- انجام عملیات محاسباتی و همچنین مقایسه‌ای نیز روی اشاره‌گرها ممکن است، هرچند نتیجه برخی ممکن است معنی خاصی نداشته باشد.

اشاره‌گرها

عملیات روی اشاره‌گرها:

- محل ذخیره‌سازی متغیر در حافظه توسط برنامه‌نویس، قابل تعیین نیست.
- روش بدست آوردن آدرس یا اشاره‌گر به یک متغیر چیست؟ عملگر &
- به عنوان مثال، اگر x یک متغیر باشد، عبارت &x، اشاره‌گر به متغیر x را به برنامه‌نویس می‌دهد.
- به عنوان مثال، برای ذخیره کردن اشاره‌گر به متغیر x در متغیر A که در بالا تعریف شده است، کافی است از دستور A=&x; استفاده کنیم.
- با داشتن اشاره‌گر به یک متغیر، چگونه می‌توان به آن متغیر دسترسی داشت؟ عملگر *
- دستور A=5; *x=5; کاری مشابه دستور انجام می‌دهد.
- انجام عملیات محاسباتی و همچنین مقایسه‌ای نیز روی اشاره‌گرها ممکن است، هرچند نتیجه برخی ممکن است معنی خاصی نداشته باشد.

اشاره‌گرها

عملیات روی اشاره‌گرها:

- محل ذخیره‌سازی متغیر در حافظه توسط برنامه‌نویس، قابل تعیین نیست.
- روش بدست آوردن آدرس یا اشاره‌گر به یک متغیر چیست؟ عملگر &
- به عنوان مثال، اگر x یک متغیر باشد، عبارت &x، اشاره‌گر به متغیر x را به برنامه‌نویس می‌دهد.
- به عنوان مثال، برای ذخیره کردن اشاره‌گر به متغیر x در متغیر A که در بالا تعریف شده است، کافی است از دستور A=&x; استفاده کنیم.
- با داشتن اشاره‌گر به یک متغیر، چگونه می‌توان به آن متغیر دسترسی داشت؟ عملگر *
- دستور A=5; *کاری مشابه دستور x=5; انجام می‌دهد.
- انجام عملیات محاسباتی و همچنین مقایسه‌ای نیز روی اشاره‌گرها ممکن است، هرچند نتیجه برخی ممکن است معنی خاصی نداشته باشد.

اشاره‌گرها

عملیات روی اشاره‌گرها:

- محل ذخیره‌سازی متغیر در حافظه توسط برنامه‌نویس، قابل تعیین نیست.
- روش بدست آوردن آدرس یا اشاره‌گر به یک متغیر چیست؟ عملگر &
- به عنوان مثال، اگر x یک متغیر باشد، عبارت &x، اشاره‌گر به متغیر x را به برنامه‌نویس می‌دهد.
- به عنوان مثال، برای ذخیره کردن اشاره‌گر به متغیر x در متغیر A که در بالا تعریف شده است، کافی است از دستور A=&x; استفاده کنیم.
- با داشتن اشاره‌گر به یک متغیر، چگونه می‌توان به آن متغیر دسترسی داشت؟ عملگر *
- دستور A=5; *x=5; کاری مشابه دستور x=5; انجام می‌دهد.
- انجام عملیات محاسباتی و همچنین مقایسه‌ای نیز روی اشاره‌گرها ممکن است، هرچند نتیجه برخی ممکن است معنی خاصی نداشته باشد.

اشاره‌گرها

عملیات روی اشاره‌گرها:

- محل ذخیره‌سازی متغیر در حافظه توسط برنامه‌نویس، قابل تعیین نیست.
- روش بدست آوردن آدرس یا اشاره‌گر به یک متغیر چیست؟ عملگر &
- به عنوان مثال، اگر x یک متغیر باشد، عبارت &x، اشاره‌گر به متغیر x را به برنامه‌نویس می‌دهد.
- به عنوان مثال، برای ذخیره کردن اشاره‌گر به متغیر x در متغیر A که در بالا تعریف شده است، کافی است از دستور A=&x; استفاده کنیم.
- با داشتن اشاره‌گر به یک متغیر، چگونه می‌توان به آن متغیر دسترسی داشت؟ عملگر *
- دستور A=5; *x=5; کاری مشابه دستور انجام می‌دهد.
- انجام عملیات محاسباتی و همچنین مقایسه‌ای نیز روی اشاره‌گرها ممکن است، هرچند نتیجه برخی ممکن است معنی خاصی نداشته باشد.

اشاره‌گرها

عملیات روی اشاره‌گرها:

- محل ذخیره‌سازی متغیر در حافظه توسط برنامه‌نویس، قابل تعیین نیست.
- روش بدست آوردن آدرس یا اشاره‌گر به یک متغیر چیست؟ عملگر &
- به عنوان مثال، اگر x یک متغیر باشد، عبارت &x، اشاره‌گر به متغیر x را به برنامه‌نویس می‌دهد.
- به عنوان مثال، برای ذخیره کردن اشاره‌گر به متغیر x در متغیر A که در بالا تعریف شده است، کافی است از دستور A=&x; استفاده کنیم.
- با داشتن اشاره‌گر به یک متغیر، چگونه می‌توان به آن متغیر دسترسی داشت؟ عملگر *
- دستور A=5; *x=5; کاری مشابه دستور انجام می‌دهد.
- انجام عملیات محاسباتی و همچنین مقایسه‌ای نیز روی اشاره‌گرها ممکن است، هرچند نتیجه برخی ممکن است معنی خاصی نداشته باشد.

اشاره‌گرها

عملیات روی اشاره‌گرها:

- محل ذخیره‌سازی متغیر در حافظه توسط برنامه‌نویس، قابل تعیین نیست.
- روش بدست آوردن آدرس یا اشاره‌گر به یک متغیر چیست؟ عملگر &
- به عنوان مثال، اگر x یک متغیر باشد، عبارت &x، اشاره‌گر به متغیر x را به برنامه‌نویس می‌دهد.
- به عنوان مثال، برای ذخیره کردن اشاره‌گر به متغیر x در متغیر A که در بالا تعریف شده است، کافی است از دستور A=&x; استفاده کنیم.
- با داشتن اشاره‌گر به یک متغیر، چگونه می‌توان به آن متغیر دسترسی داشت؟ عملگر *
- دستور A=5; *x=5; کاری مشابه دستور انجام می‌دهد.
- انجام عملیات محاسباتی و همچنین مقایسه‌ای نیز روی اشاره‌گرها ممکن است، هرچند نتیجه برخی ممکن است معنی خاصی نداشته باشد.

اشاره‌گرها

رابطهٔ آرایه با اشاره‌گرها:

- با توجه به این که آرایه در یک فضای به هم پیوسته از حافظه ذخیره می‌شود؛ لذا آدرس محل شروع ذخیره‌سازی آرایه در حافظه، در حقیقت همان اشاره‌گر به درایه اول آرایه است.
- مثال: اگر آرایهٔ double Array[100]; تعریف کنیم، محل شروع ذخیره‌سازی آرایه، با دستور &A[0] مشخص می‌شود.
- روش ساده‌تر: نام هر آرایه (با هر تعداد بعد)، حاوی اشاره‌گر به آن آرایه است.
- فراخوانی در خصوص آرایه‌ها، در عمل، فراخوانی با ارجاع است.

اشاره‌گرها

رابطهٔ آرایه با اشاره‌گرها:

- با توجه به این که آرایه در یک فضای به هم پیوسته از حافظه ذخیره می‌شود؛ لذا آدرس محل شروع ذخیره‌سازی آرایه در حافظه، در حقیقت همان اشاره‌گر به درایه اول آرایه است.
- مثال: اگر آرایهٔ double Array[100]; تعریف کنیم، محل شروع ذخیره‌سازی آرایه، با دستور &A[0] مشخص می‌شود.
- روش ساده‌تر: نام هر آرایه (با هر تعداد بعد)، حاوی اشاره‌گر به آن آرایه است.
- فراخوانی در خصوص آرایه‌ها، در عمل، فراخوانی با ارجاع است.

اشاره‌گرها

رابطهٔ آرایه با اشاره‌گرها:

- با توجه به این که آرایه در یک فضای به هم پیوسته از حافظه ذخیره می‌شود؛ لذا آدرس محل شروع ذخیره‌سازی آرایه در حافظه، در حقیقت همان اشاره‌گر به درایه اول آرایه است.
- مثال: اگر آرایهٔ double Array[100]; تعریف کنیم، محل شروع ذخیره‌سازی آرایه، با دستور &A[0] مشخص می‌شود.
- روش ساده‌تر: نام هر آرایه (با هر تعداد بعد)، حاوی اشاره‌گر به آن آرایه است.
- فراخوانی در خصوص آرایه‌ها، در عمل، فراخوانی با ارجاع است.

اشاره‌گرها

رابطهٔ آرایه با اشاره‌گرها:

- با توجه به این که آرایه در یک فضای به هم پیوسته از حافظه ذخیره می‌شود؛ لذا آدرس محل شروع ذخیره‌سازی آرایه در حافظه، در حقیقت همان اشاره‌گر به درایه اول آرایه است.
- مثال: اگر آرایهٔ double Array[100]; تعریف کنیم، محل شروع ذخیره‌سازی آرایه، با دستور &A[0] مشخص می‌شود.
- روش ساده‌تر: نام هر آرایه (با هر تعداد بعد)، حاوی اشاره‌گر به آن آرایه است.
- فراخوانی در خصوص آرایه‌ها، در عمل، فراخوانی با ارجاع است.

اشاره‌گرها

```
1  #include <iostream>
2  using namespace std;
3  int  main(void)
4  {
5       double A[10];
6        int  i ;
7       for  (i=0;i<10;i++)
8       {
9         cout<<"\nEnter a Number:";
10        cin>>A[i];
11      }
12      cout<<endl;
13      for  (i=0;i<10;i++)
14      {
15        cout<<A[i]<<",";
16      }
17      return  0;
18 }
```

```
1  #include <iostream>
2  using namespace std;
3  int  main(void)
4  {
5       double A[10];
6        int  i ;
7       double *Aptr;
8       Aptr=A;
9       for  (i=0;i<10;i++)
10      {
11        cout<<"\nEnter a Number:";
12        cin>>*Aptr;
13        Aptr++;
14      }
15      cout<<endl;
16      Aptr=A;
17      for  (i=0;i<10;i++)
18      {
19        cout<<*Aptr<<",";
20        Aptr++;
21      }
22      return  0;
23 }
```

اشاره‌گرها

کاربردهای اشاره‌گرها:

- فراخوانی با ارجاع
- تخصیص حافظهٔ پویا
- پیاده‌سازی برخی ساختمان داده‌ها
- برنامه‌نویسی سیستم

اشاره‌گرها

کاربردهای اشاره‌گرها:

- فراخوانی با ارجاع
- تخصیص حافظهٔ پویا
- پیاده‌سازی برخی ساختمان داده‌ها
- برنامه‌نویسی سیستم

اشاره‌گرها

کاربردهای اشاره‌گرها:

- فراخوانی با ارجاع
- تخصیص حافظهٔ پویا
- پیاده‌سازی برخی ساختمان داده‌ها
- برنامه‌نویسی سیستم

اشاره‌گرها

کاربردهای اشاره‌گرها:

- فراخوانی با ارجاع
- تخصیص حافظهٔ پویا
- پیاده‌سازی برخی ساختمان داده‌ها
- برنامه‌نویسی سیستم

اشاره‌گرها

```
1  #include <iostream>
2  using namespace std;
3  void swap(int x,  int  y)
4  {
5      int  temp;
6      temp = x; /* save the value at  address x */
7      x = y;      /* put y  into  x  */
8      y = temp; /* put  x  into  y  */
9      return;
10 }
11 int  main ()
12 {
13     // local  variable  declaration:
14     int  a = 100;
15     int  b = 200;
16     cout << "Before swap, value of a  :"  << a << endl;
17     cout << "Before swap, value of b  :"  << b << endl;
18     /* calling  a function  to  swap the values using variable reference. */
19     swap(a, b);
20     cout << "After  swap, value of a  :"  << a << endl;
21     cout << "After  swap, value of b  :"  << b << endl;
22     return  0;
23 }
```

اشاره‌گرها

فراخوانی با ارجاع با استفاده از اشاره‌گرها

```
1  void swap(int *x,  int  *y)
2  {
3      int  temp;
4      temp = *x; /* save the value at address x */
5      *x = *y;      /* put y into x */
6      *y = temp; /* put x into y */
7      return;
8  }
```

فراخوانی با ارجاع با استفاده از متغیرهای مرجع

```
1  void swap(int &x,  int  &y)
2  {
3      int  temp;
4      temp = x; /* save the value at address x */
5      x = y;      /* put y into x */
6      y = temp; /* put x into y */
7      return;
8  }
```

تخصیص حافظهٔ پویا

تخصیص حافظهٔ پویا:

- هر برنامه‌ای برای پردازش داده‌ها نوشته می‌شود و لذا نیاز به فضایی برای ذخیره‌سازی داده‌ها در حافظهٔ کامپیوتر و پردازش آنها
- مثال: می‌خواهیم برنامه‌ای بنویسیم که ابتدا عدد n و سپس n عدد را از ورودی بگیرد و روی آن محاسباتی انجام دهد.
- برای این منظور آرایه‌ای برای نگهداری اعداد وارد شده تعریف می‌کنیم. اما اندازهٔ آرایه را چقدر در نظر بگیریم؟
- تنها راه برای حل این مشکل این است که در زمان اجرای برنامه، و بعد از وارد کردن عدد n توسط کاربر، مقدار حافظهٔ لازم به آرایه تخصیص داده شود.
- اختصاص حافظه در زمان اجرای برنامه را اصطلاحاً تخصیص حافظهٔ پویا گویند.

تخصیص حافظهٔ پویا

تخصیص حافظهٔ پویا:

- هر برنامه‌ای برای پردازش داده‌ها نوشته می‌شود و لذا نیاز به فضایی برای ذخیره‌سازی داده‌ها در حافظهٔ کامپیوتر و پردازش آنها
- مثال: می‌خواهیم برنامه‌ای بنویسیم که ابتدا عدد n و سپس n عدد را از ورودی بگیرد و روی آن محاسباتی انجام دهد.
- برای این منظور آرایه‌ای برای نگهداری اعداد وارد شده تعریف می‌کنیم. اما اندازهٔ آرایه را چقدر در نظر بگیریم؟
- تنها راه برای حل این مشکل این است که در زمان اجرای برنامه، و بعد از وارد کردن عدد n توسط کاربر، مقدار حافظهٔ لازم به آرایه تخصیص داده شود.
- اختصاص حافظه در زمان اجرای برنامه را اصطلاحاً تخصیص حافظهٔ پویا گویند.

تخصیص حافظهٔ پویا

تخصیص حافظهٔ پویا:

- هر برنامه‌ای برای پردازش داده‌ها نوشته می‌شود و لذا نیاز به فضایی برای ذخیره‌سازی داده‌ها در حافظهٔ کامپیوتر و پردازش آنها
- مثال: می‌خواهیم برنامه‌ای بنویسیم که ابتدا عدد n و سپس n عدد را از ورودی بگیرد و روی آن محاسباتی انجام دهد.
- برای این منظور آرایه‌ای برای نگهداری اعداد وارد شده تعریف می‌کنیم. اما اندازهٔ آرایه را چقدر در نظر بگیریم؟
- تنها راه برای حل این مشکل این است که در زمان اجرای برنامه، و بعد از وارد کردن عدد n توسط کاربر، مقدار حافظهٔ لازم به آرایه تخصیص داده شود.
- اختصاص حافظه در زمان اجرای برنامه را اصطلاحاً تخصیص حافظهٔ پویا گویند.

تخصیص حافظهٔ پویا

تخصیص حافظهٔ پویا:

- هر برنامه‌ای برای پردازش داده‌ها نوشته می‌شود و لذا نیاز به فضایی برای ذخیره‌سازی داده‌ها در حافظهٔ کامپیوتر و پردازش آنها
- مثال: می‌خواهیم برنامه‌ای بنویسیم که ابتدا عدد n و سپس n عدد را از ورودی بگیرد و روی آن محاسباتی انجام دهد.
- برای این منظور آرایه‌ای برای نگهداری اعداد وارد شده تعریف می‌کنیم. اما اندازهٔ آرایه را چقدر در نظر بگیریم؟
- تنها راه برای حل این مشکل این است که در زمان اجرای برنامه، و بعد از وارد کردن عدد n توسط کاربر، مقدار حافظهٔ لازم به آرایه تخصیص داده شود.
- اختصاص حافظه در زمان اجرای برنامه را اصطلاحاً تخصیص حافظهٔ پویا گویند.

تخصیص حافظهٔ پویا

تخصیص حافظهٔ پویا:

- هر برنامه‌ای برای پردازش داده‌ها نوشته می‌شود و لذا نیاز به فضایی برای ذخیره‌سازی داده‌ها در حافظهٔ کامپیوتر و پردازش آنها
- مثال: می‌خواهیم برنامه‌ای بنویسیم که ابتدا عدد n و سپس n عدد را از ورودی بگیرد و روی آن محاسباتی انجام دهد.
- برای این منظور آرایه‌ای برای نگهداری اعداد وارد شده تعریف می‌کنیم. اما اندازهٔ آرایه را چقدر در نظر بگیریم؟
- تنها راه برای حل این مشکل این است که در زمان اجرای برنامه، و بعد از وارد کردن عدد n توسط کاربر، مقدار حافظهٔ لازم به آرایه تخصیص داده شود.
- اختصاص حافظه در زمان اجرای برنامه را اصطلاحاً تخصیص حافظهٔ پویا گویند.

تخصیص حافظهٔ پویا

دستور تخصیص حافظهٔ پویا:

- اختصاص حافظه در C++ با دستور new انجام می‌شود.
- گرامر استفاده از دستور new به صورت new type; است که در آن منظور از type، نوع متغیری است که برای آن حافظه نیاز داریم.
- مقدار بازگشتی دستور new، اشاره‌گر به حافظه‌ای است که اختصاص داده شده است.
- double *A;
- A= new double;
- تخصیص حافظه برای آرایهٔ ۱۰ تایی: A= new double [10];
- تخصیص حافظه برای آرایهٔ n تایی: A= new double [n];
- اگر عملیات تخصیص حافظهٔ پویا موفقیت آمیز نباشد، آنگاه دستور new مقدار nullptr یا NULL را برمی‌گرداند.

تخصیص حافظهٔ پویا

دستور تخصیص حافظهٔ پویا:

- اختصاص حافظه در C++ با دستور new انجام می‌شود.
- گرامر استفاده از دستور new به صورت ;new type است که در آن منظور از type، نوع متغیری است که برای آن حافظه نیاز داریم.
- مقدار بازگشتی دستور new، اشاره‌گر به حافظه‌ای است که اختصاص داده شده است.
- double *A;
- A= new double;
- تخصیص حافظه برای آرایهٔ ۱۰ تایی: A= new double [10];
- تخصیص حافظه برای آرایهٔ n تایی: A= new double [n];
- اگر عملیات تخصیص حافظهٔ پویا موفقیت آمیز نباشد، آنگاه دستور new مقدار nullptr یا NULL را برمی‌گرداند.

تخصیص حافظهٔ پویا

دستور تخصیص حافظهٔ پویا:

- اختصاص حافظه در C++ با دستور new انجام می‌شود.
- گرامر استفاده از دستور new به صورت ;new type است که در آن منظور از type، نوع متغیری است که برای آن حافظه نیاز داریم.
- مقدار بازگشتی دستور new، اشاره‌گر به حافظه‌ای است که اختصاص داده شده است.
- double *A;
- A= new double;
- تخصیص حافظه برای آرایهٔ ۱۰ تایی: A= new double [10];
- تخصیص حافظه برای آرایهٔ n تایی: A= new double [n];
- اگر عملیات تخصیص حافظهٔ پویا موفقیت آمیز نباشد، آنگاه دستور new مقدار nullptr یا NULL را برمی‌گرداند.

تخصیص حافظهٔ پویا

دستور تخصیص حافظهٔ پویا:

- اختصاص حافظه در C++ با دستور new انجام می‌شود.
- گرامر استفاده از دستور new به صورت ;new type است که در آن منظور از type، نوع متغیری است که برای آن حافظه نیاز داریم.
- مقدار بازگشتی دستور new، اشاره‌گر به حافظه‌ای است که اختصاص داده شده است.
- double *A;
- A= new double;
- تخصیص حافظه برای آرایهٔ ۱۰ تایی: A= new double [10];
- تخصیص حافظه برای آرایهٔ n تایی: A= new double [n];
- اگر عملیات تخصیص حافظهٔ پویا موفقیت آمیز نباشد، آنگاه دستور new مقدار nullptr یا NULL را برمی‌گرداند.

تخصیص حافظهٔ پویا

دستور تخصیص حافظهٔ پویا:

- اختصاص حافظه در C++ با دستور new انجام می‌شود.
- گرامر استفاده از دستور new به صورت ;new type است که در آن منظور از type، نوع متغیری است که برای آن حافظه نیاز داریم.
- مقدار بازگشتی دستور new، اشاره‌گر به حافظه‌ای است که اختصاص داده شده است.

```
double *A;
A= new double;
```

- تخصیص حافظه برای آرایهٔ ۱۰ تایی: ;A= new double [10]
- تخصیص حافظه برای آرایهٔ n تایی: ;A= new double [n]
- اگر عملیات تخصیص حافظهٔ پویا موفقیت آمیز نباشد، آنگاه دستور new مقدار nullptr یا NULL را برمی‌گرداند.

تخصیص حافظهٔ پویا

دستور تخصیص حافظهٔ پویا:

- اختصاص حافظه در C++ با دستور new انجام می‌شود.
- گرامر استفاده از دستور new به صورت new type; است که در آن منظور از type، نوع متغیری است که برای آن حافظه نیاز داریم.
- مقدار بازگشتی دستور new، اشاره‌گر به حافظه‌ای است که اختصاص داده شده است.

```
double *A;
A= new double;
```

- تخصیص حافظه برای آرایهٔ ۱۰ تایی: A= new double [10];
- تخصیص حافظه برای آرایهٔ n تایی: A= new double [n];
- اگر عملیات تخصیص حافظهٔ پویا موفقیت آمیز نباشد، آنگاه دستور new مقدار nullptr یا NULL را برمی‌گرداند.

تخصیص حافظهٔ پویا

دستور تخصیص حافظهٔ پویا:

- اختصاص حافظه در C++ با دستور new انجام می‌شود.
- گرامر استفاده از دستور new به صورت new type; است که در آن منظور از type، نوع متغیری است که برای آن حافظه نیاز داریم.
- مقدار بازگشتی دستور new، اشاره‌گر به حافظه‌ای است که اختصاص داده شده است.
- double *A;
- A= new double;
- تخصیص حافظه برای آرایهٔ ۱۰ تایی: A= new double [10];
- تخصیص حافظه برای آرایهٔ n تایی: A= new double [n];
- اگر عملیات تخصیص حافظهٔ پویا موفقیت آمیز نباشد، آنگاه دستور new مقدار nullptr یا NULL را برمی‌گرداند.

تخصیص حافظهٔ پویا

دستور تخصیص حافظهٔ پویا:

- اختصاص حافظه در C++ با دستور new انجام می‌شود.
- گرامر استفاده از دستور new به صورت new type; است که در آن منظور از type، نوع متغیری است که برای آن حافظه نیاز داریم.
- مقدار بازگشتی دستور new، اشاره‌گر به حافظه‌ای است که اختصاص داده شده است.
- double *A;
- A= new double;
- تخصیص حافظه برای آرایهٔ ۱۰ تایی: A= new double [10];
- تخصیص حافظه برای آرایهٔ *n* تایی: A= new double [n];
- اگر عملیات تخصیص حافظهٔ پویا موفقیت آمیز نباشد، آنگاه دستور new مقدار nullptr یا NULL را برمی‌گرداند.

تخصیص حافظهٔ پویا

```
1  #include <iostream>
2  using namespace std;
3  int  main(void)
4  {
5       double *A;
6        int  n;
7       cout<<"Enter n:";
8       cin>>n;
9       A= new double [n];
10       if (A== nullptr)
11       {
12            cout<<"Memory Allocation Failed. We terminate the
                    program here!";
13            return  -1;
14       }
15       cout<<"Memory Allocation Succeeded. Continue the program
                    ...\n";
16       for( int  i=0;i<n;i++)
17       {
18            cout<<"Enter next number:";
19            cin>>A[i];
20       }
21          cout<<endl;
22          for  ( int  i=0;i<n;i++)
23       {
24            cout<<A[i]<<",";
25       }
26       //We need to free the allocated  memory here
27       return  0;
28  }
```

تخصیص حافظهٔ پویا

آزاد کردن حافظه:

- تخصیص حافظه و آزاد کردن حافظه در متغیرهای معمول به صورت خودکار انجام می‌شود.
- طول عمر متغیر: بازه زمانی که آن متغیر دارای حافظه‌ای در حافظهٔ اصلی کامپیوتر است.
- در پایان عمر متغیر: حافظهٔ در نظر گرفته شده برای متغیر اصطلاحاً آزاد می‌شود.
- در تخصیص حافظهٔ پویا: باید حافظهٔ تخصیص داده، پس از اتمام نیاز به آن، با دستوری توسط برنامه‌نویس آزاد شود.
- `delete pointer;`
- آزاد کردن حافظهٔ آرایه: `delete [] A;`
- در صورت آزاد نکردن حافظهٔ تخصیص داده شده: اشکال خاصی در اجرای برنامه ایجاد نمی‌شود، ولی ممکن است اشکالاتی ایجاد کند.

تخصیص حافظهٔ پویا

آزاد کردن حافظه:

- تخصیص حافظه و آزاد کردن حافظه در متغیرهای معمول به صورت خودکار انجام می‌شود.
- **طول عمر متغیر: بازه زمانی که آن متغیر دارای حافظه‌ای در حافظهٔ اصلی کامپیوتر است.**
- در پایان عمر متغیر: حافظهٔ در نظر گرفته شده برای متغیر اصطلاحاً آزاد می‌شود.
- در تخصیص حافظهٔ پویا: باید حافظهٔ تخصیص داده، پس از اتمام نیاز به آن، با دستوری توسط برنامه‌نویس آزاد شود.
- `delete pointer;`
- آزاد کردن حافظهٔ آرایه: `delete [] A;`
- در صورت آزاد نکردن حافظهٔ تخصیص داده شده: اشکال خاصی در اجرای برنامه ایجاد نمی‌شود، ولی ممکن است اشکالاتی ایجاد کند.

تخصیص حافظهٔ پویا

آزاد کردن حافظه:

- تخصیص حافظه و آزاد کردن حافظه در متغیرهای معمول به صورت خودکار انجام می‌شود.
- **طول عمر متغیر:** بازه زمانی که آن متغیر دارای حافظه‌ای در حافظهٔ اصلی کامپیوتر است.
- در پایان عمر متغیر: حافظهٔ در نظر گرفته شده برای متغیر اصطلاحاً آزاد می‌شود.
- در تخصیص حافظهٔ پویا: باید حافظهٔ تخصیص داده، پس از اتمام نیاز به آن، با دستوری توسط برنامه‌نویس آزاد شود.
- `delete pointer;`
- آزاد کردن حافظهٔ آرایه: `delete [] A;`
- در صورت آزاد نکردن حافظهٔ تخصیص داده شده: اشکال خاصی در اجرای برنامه ایجاد نمی‌شود، ولی ممکن است اشکالاتی ایجاد کند.

تخصیص حافظهٔ پویا

آزاد کردن حافظه:

- تخصیص حافظه و آزاد کردن حافظه در متغیرهای معمول به صورت خودکار انجام می‌شود.
- **طول عمر متغیر:** بازه زمانی که آن متغیر دارای حافظه‌ای در حافظهٔ اصلی کامپیوتر است.
- در پایان عمر متغیر: حافظهٔ در نظر گرفته شده برای متغیر اصطلاحاً آزاد می‌شود.
- در تخصیص حافظهٔ پویا: باید حافظهٔ تخصیص داده، پس از اتمام نیاز به آن، با دستوری توسط برنامه‌نویس آزاد شود.
  - delete pointer;
  - آزاد کردن حافظهٔ آرایه: delete [] A;
  - در صورت آزاد نکردن حافظهٔ تخصیص داده شده: اشکال خاصی در اجرای برنامه ایجاد نمی‌شود، ولی ممکن است اشکالاتی ایجاد کند.

تخصیص حافظهٔ پویا

آزاد کردن حافظه:

- تخصیص حافظه و آزاد کردن حافظه در متغیرهای معمول به صورت خودکار انجام می‌شود.
- **طول عمر متغیر:** بازه زمانی که آن متغیر دارای حافظه‌ای در حافظهٔ اصلی کامپیوتر است.
- در پایان عمر متغیر: حافظهٔ در نظر گرفته شده برای متغیر اصطلاحاً آزاد می‌شود.
- در تخصیص حافظهٔ پویا: باید حافظهٔ تخصیص داده، پس از اتمام نیاز به آن، با دستوری توسط برنامه‌نویس آزاد شود.
- `delete pointer;`
- آزاد کردن حافظهٔ آرایه: `delete [] A;`
- در صورت آزاد نکردن حافظهٔ تخصیص داده شده: اشکال خاصی در اجرای برنامه ایجاد نمی‌شود، ولی ممکن است اشکالاتی ایجاد کند.

تخصیص حافظهٔ پویا

آزاد کردن حافظه:

- تخصیص حافظه و آزاد کردن حافظه در متغیرهای معمول به صورت خودکار انجام می‌شود.
- طول عمر متغیر: بازه زمانی که آن متغیر دارای حافظه‌ای در حافظهٔ اصلی کامپیوتر است.
- در پایان عمر متغیر: حافظهٔ در نظر گرفته شده برای متغیر اصطلاحاً آزاد می‌شود.
- در تخصیص حافظهٔ پویا: باید حافظهٔ تخصیص داده، پس از اتمام نیاز به آن، با دستوری توسط برنامه‌نویس آزاد شود.
- delete pointer;
- آزاد کردن حافظهٔ آرایه: delete [] A;
- در صورت آزاد نکردن حافظهٔ تخصیص داده شده: اشکال خاصی در اجرای برنامه ایجاد نمی‌شود، ولی ممکن است اشکالاتی ایجاد کند.

تخصیص حافظهٔ پویا

آزاد کردن حافظه:

- تخصیص حافظه و آزاد کردن حافظه در متغیرهای معمول به صورت خودکار انجام می‌شود.
- طول عمر متغیر: بازه زمانی که آن متغیر دارای حافظه‌ای در حافظهٔ اصلی کامپیوتر است.
- در پایان عمر متغیر: حافظهٔ در نظر گرفته شده برای متغیر اصطلاحاً آزاد می‌شود.
- در تخصیص حافظهٔ پویا: باید حافظهٔ تخصیص داده، پس از اتمام نیاز به آن، با دستوری توسط برنامه‌نویس آزاد شود.
- `delete pointer;`
- آزاد کردن حافظهٔ آرایه: `delete [] A;`
- در صورت آزاد نکردن حافظهٔ تخصیص داده شده: اشکال خاصی در اجرای برنامه ایجاد نمی‌شود، ولی ممکن است اشکالاتی ایجاد کند.

تخصیص حافظهٔ پویا

نشت حافظه:

- منظور از نشت آب از لوله؛ خارج شدن آب از لوله است که با این اتفاق؛ آب خارج شده؛ عملاً به دست استفاده کننده نمی‌رسد و بدون استفاده تلف می‌شود.
- منظور از نشت حافظه، از دسترس خارج شدن حافظهٔ کامپیوتر برای استفاده است.
- اصولاً حافظه کامپیوتر، توسط سیستم عامل کامپیوتر مدیریت می‌شود.
- بلا استفاده شدن حافظهٔ آزاد نشده.
- وجود نشت حافظه در برنامه‌های کاربردی؛ می‌تواند کامپیوتر را با کمبود حافظه مواجه کند.
- یک راهکار جایگزین: اشاره‌گرهای هوشمند

تخصیص حافظهٔ پویا

نشت حافظه:

- منظور از نشت آب از لوله؛ خارج شدن آب از لوله است که با این اتفاق؛ آب خارج شده؛ عملاً به دست استفاده کننده نمی‌رسد و بدون استفاده تلف می‌شود.
- منظور از نشت حافظه، از دسترس خارج شدن حافظهٔ کامپیوتر برای استفاده است.
- اصولاً حافظه کامپیوتر، توسط سیستم عامل کامپیوتر مدیریت می‌شود.
- بلا استفاده شدن حافظهٔ آزاد نشده.
- وجود نشت حافظه در برنامه‌های کاربردی: می‌تواند کامپیوتر را با کمبود حافظه مواجه کند.
- یک راهکار جایگزین: اشاره‌گرهای هوشمند

تخصیص حافظهٔ پویا

نشت حافظه:

- منظور از نشت آب از لوله؛ خارج شدن آب از لوله است که با این اتفاق؛ آب خارج شده؛ عملاً به دست استفاده کننده نمی‌رسد و بدون استفاده تلف می‌شود.
- منظور از نشت حافظه، از دسترس خارج شدن حافظهٔ کامپیوتر برای استفاده است.
- اصولاً حافظه کامپیوتر، توسط سیستم عامل کامپیوتر مدیریت می‌شود.
- بلا استفاده شدن حافظهٔ آزاد نشده.
- وجود نشت حافظه در برنامه‌های کاربردی؛ می‌تواند کامپیوتر را با کمبود حافظه مواجه کند.
- یک راهکار جایگزین: اشاره‌گرهای هوشمند

تخصیص حافظهٔ پویا

نشت حافظه:

- منظور از نشت آب از لوله؛ خارج شدن آب از لوله است که با این اتفاق؛ آب خارج شده؛ عملاً به دست استفاده کننده نمی‌رسد و بدون استفاده تلف می‌شود.
- منظور از نشت حافظه، از دسترس خارج شدن حافظهٔ کامپیوتر برای استفاده است.
- اصولاً حافظه کامپیوتر، توسط سیستم عامل کامپیوتر مدیریت می‌شود.
- بلا استفاده شدن حافظهٔ آزاد نشده.
- وجود نشت حافظه در برنامه‌های کاربردی: می‌تواند کامپیوتر را با کمبود حافظه مواجه کند.
- یک راهکار جایگزین: اشاره‌گرهای هوشمند

تخصیص حافظهٔ پویا

نشت حافظه:

- منظور از نشت آب از لوله؛ خارج شدن آب از لوله است که با این اتفاق؛ آب خارج شده؛ عملاً به دست استفاده کننده نمی‌رسد و بدون استفاده تلف می‌شود.
- منظور از نشت حافظه، از دسترس خارج شدن حافظهٔ کامپیوتر برای استفاده است.
- اصولاً حافظه کامپیوتر، توسط سیستم عامل کامپیوتر مدیریت می‌شود.
- بلا استفاده شدن حافظهٔ آزاد نشده.
- وجود نشت حافظه در برنامه‌های کاربردی: می‌تواند کامپیوتر را با کمبود حافظه مواجه کند.
- یک راهکار جایگزین: اشاره‌گرهای هوشمند

تخصیص حافظهٔ پویا

نشت حافظه:

- منظور از نشت آب از لوله؛ خارج شدن آب از لوله است که با این اتفاق؛ آب خارج شده؛ عملاً به دست استفاده کننده نمی‌رسد و بدون استفاده تلف می‌شود.
- منظور از نشت حافظه، از دسترس خارج شدن حافظهٔ کامپیوتر برای استفاده است.
- اصولاً حافظه کامپیوتر، توسط سیستم عامل کامپیوتر مدیریت می‌شود.
- بلا استفاده شدن حافظهٔ آزاد نشده.
- وجود نشت حافظه در برنامه‌های کاربردی: می‌تواند کامپیوتر را با کمبود حافظه مواجه کند.
- یک راهکار جایگزین: اشاره‌گرهای هوشمند

اشاره‌گر به اشاره‌گر

اشاره‌گر به اشاره‌گر:

- با تعریف یک متغیر، محلی در حافظهٔ اصلی کامپیوتر به آن متغیر اختصاص داده می‌شود. این مهم در خصوص متغیرهای اشاره‌گری نیز اتفاق می‌افتد.
- int *x;
- اگر بخواهیم آدرس متغیر x را در متغیر y ذخیره کنیم، متغیر y باید از نوع int * * تعریف شود.
- این روند، می‌تواند به همین صورت تکرار شود و مجدداً، آدرس متغیر y از نوع int * * * خواهد بود.

اشاره‌گر به اشاره‌گر

اشاره‌گر به اشاره‌گر:

- با تعریف یک متغیر، محلی در حافظهٔ اصلی کامپیوتر به آن متغیر اختصاص داده می‌شود. این مهم در خصوص متغیرهای اشاره‌گری نیز اتفاق می‌افتد.
- `int **x;`
- اگر بخواهیم آدرس متغیر x را در متغیر y ذخیره کنیم، متغیر y باید از نوع `int * *` تعریف شود.
- این روند، می‌تواند به همین صورت تکرار شود و مجدداً، آدرس متغیر y از نوع `int * * *` خواهد بود.

اشاره‌گر به اشاره‌گر

اشاره‌گر به اشاره‌گر:

- با تعریف یک متغیر، محلی در حافظهٔ اصلی کامپیوتر به آن متغیر اختصاص داده می‌شود. این مهم در خصوص متغیرهای اشاره‌گری نیز اتفاق می‌افتد.
- `int *x;`
- اگر بخواهیم آدرس متغیر x را در متغیر y ذخیره کنیم، متغیر y باید از نوع `int * *` تعریف شود.
- این روند، می‌تواند به همین صورت تکرار شود و مجدداً، آدرس متغیر y از نوع `int * * *` خواهد بود.

اشاره‌گر به اشاره‌گر

اشاره‌گر به اشاره‌گر:

- با تعریف یک متغیر، محلی در حافظهٔ اصلی کامپیوتر به آن متغیر اختصاص داده می‌شود. این مهم در خصوص متغیرهای اشاره‌گری نیز اتفاق می‌افتد.
- int *x;
- اگر بخواهیم آدرس متغیر x را در متغیر y ذخیره کنیم، متغیر y باید از نوع int * * تعریف شود.
- این روند، می‌تواند به همین صورت تکرار شود و مجدداً، آدرس متغیر y از نوع int * * * خواهد بود.

اشاره‌گر به اشاره‌گر

```
1  #include <iostream>
2  using namespace std;
3   int  main()
4  {
5     int  variable  = 169;
6      int* pointer1;
7      int** pointer2;
8      pointer1  = &variable;
9      pointer2  = &pointer1;
10     cout<<"Value of variable :"<< variable  <<"\n";
11     cout<<"Value of variable using single  pointer :"  <<*pointer1<<"\n";
12     cout<<"Value of variable using double pointer:"<<**pointer2<<"\n";
13    return  0;
14 }
```

۳۲ بیتی یا ۶۴ بیتی؟

۳۲ بیتی یا ۶۴ بیتی؟

- تفاوت کامپیوترهای ۳۲ بیتی و ۶۴ بیتی؟ (فضای آدرس‌دهی)
- کامپیوتری که من استفاده می‌کنم ۳۲ بیتی است یا ۶۴ بیتی؟
- آیا می‌توان از طریق برنامه‌نویسی این را تشخیص داد؟
- شما کافی است اندازه یک متغیر اشاره‌گری را بدانید و اگر این اندازه؛ ۴ بایت بود، معنی آن این است که برنامهٔ شما به صورت ۳۲ بیتی اجرا شده است و اگر حاصل ۸ بایت بود، به معنی ۶۴ بیتی بودن برنامه شماست.
- دستور cout<<sizeof(int*);

۳۲ بیتی یا ۶۴ بیتی؟

۳۲ بیتی یا ۶۴ بیتی؟

- تفاوت کامپیوترهای ۳۲ بیتی و ۶۴ بیتی؟ (فضای آدرس‌دهی)
- کامپیوتری که من استفاده می‌کنم ۳۲ بیتی است یا ۶۴ بیتی؟
- آیا می‌توان از طریق برنامه‌نویسی این را تشخیص داد؟
- شما کافی است اندازه یک متغیر اشاره‌گری را بدانید و اگر این اندازه؛ ۴ بایت بود، معنی آن این است که برنامهٔ شما به صورت ۳۲ بیتی اجرا شده است و اگر حاصل ۸ بایت بود، به معنی ۶۴ بیتی بودن برنامه شماست.
- دستور cout<<sizeof(int*);

۳۲ بیتی یا ۶۴ بیتی؟

۳۲ بیتی یا ۶۴ بیتی؟

- تفاوت کامپیوترهای ۳۲ بیتی و ۶۴ بیتی؟ (فضای آدرس‌دهی)
- کامپیوتری که من استفاده می‌کنم ۳۲ بیتی است یا ۶۴ بیتی؟
- آیا می‌توان از طریق برنامه‌نویسی این را تشخیص داد؟
- شما کافی است اندازه یک متغیر اشاره‌گری را بدانید و اگر این اندازه؛ ۴ بایت بود، معنی آن این است که برنامهٔ شما به صورت ۳۲ بیتی اجرا شده است و اگر حاصل ۸ بایت بود، به معنی ۶۴ بیتی بودن برنامه شماست.
- دستور cout<<sizeof(int*);

۳۲ بیتی یا ۶۴ بیتی؟

۳۲ بیتی یا ۶۴ بیتی؟

- تفاوت کامپیوترهای ۳۲ بیتی و ۶۴ بیتی؟ (فضای آدرس‌دهی)
- کامپیوتری که من استفاده می‌کنم ۳۲ بیتی است یا ۶۴ بیتی؟
- آیا می‌توان از طریق برنامه‌نویسی این را تشخیص داد؟
- شما کافی است اندازه یک متغیر اشاره‌گری را بدانید و اگر این اندازه؛ ۴ بایت بود، معنی آن این است که برنامهٔ شما به صورت ۳۲ بیتی اجرا شده است و اگر حاصل ۸ بایت بود، به معنی ۶۴ بیتی بودن برنامه شماست.
- دستور cout<<sizeof(int*);

۳۲ بیتی یا ۶۴ بیتی؟

۳۲ بیتی یا ۶۴ بیتی؟

- تفاوت کامپیوترهای ۳۲ بیتی و ۶۴ بیتی؟ (فضای آدرس‌دهی)
- کامپیوتری که من استفاده می‌کنم ۳۲ بیتی است یا ۶۴ بیتی؟
- آیا می‌توان از طریق برنامه‌نویسی این را تشخیص داد؟
- شما کافی است اندازه یک متغیر اشاره‌گری را بدانید و اگر این اندازه؛ ۴ بایت بود، معنی آن این است که برنامهٔ شما به صورت ۳۲ بیتی اجرا شده است و اگر حاصل ۸ بایت بود، به معنی ۶۴ بیتی بودن برنامه شماست.
- دستور `cout<<sizeof(int*);`

اشاره‌گر به توابع

اشاره‌گر به توابع:

- یک برنامه کاربردی، برای اجرا، ابتدا داخل حافظه اصلی کامپیوتر قرار می‌گیرد و سپس اجرا می‌شود.
- لذا، هر دستور در برنامه نیز در محلی از حافظه ذخیره می‌شود و به همین دلیل، اشاره‌گر به دستورات مختلف برنامه نیز منطقی است.
- یک اشاره‌گر به تابع، عملاً محل شروع دستورات تابع در حافظه است.
- با فراخوانی تابع، اجرای برنامه به آن آدرس منتقل می‌شود و دستورات از آن محل به ترتیب، تا رسیدن به پایان تابع، اجرا می‌شود.
- مشابه آرایه، نام هر تابع، حاوی اشاره‌گر به آن تابع است.
- از اشاره‌گر به تابع می‌توان برای فراخوانی تابع استفاده کرد.
- همچنین اشاره‌گر به تابع را می‌توان به عنوان پارامتر، به توابع دیگر نیز منتقل کرد تا در صورت نیاز، با استفاده از آن اشاره‌گر، تابع فراخوانی شود

اشاره‌گر به توابع

اشاره‌گر به توابع:

- یک برنامه کاربردی، برای اجرا، ابتدا داخل حافظه اصلی کامپیوتر قرار می‌گیرد و سپس اجرا می‌شود.
- لذا، هر دستور در برنامه نیز در محلی از حافظه ذخیره می‌شود و به همین دلیل، اشاره‌گر به دستورات مختلف برنامه نیز منطقی است.
- یک اشاره‌گر به تابع، عملاً محل شروع دستورات تابع در حافظه است.
- با فراخوانی تابع، اجرای برنامه به آن آدرس منتقل می‌شود و دستورات از آن محل به ترتیب، تا رسیدن به پایان تابع، اجرا می‌شود.
- مشابه آرایه، نام هر تابع، حاوی اشاره‌گر به آن تابع است.
- از اشاره‌گر به تابع می‌توان برای فراخوانی تابع استفاده کرد.
- همچنین اشاره‌گر به تابع را می‌توان به عنوان پارامتر، به توابع دیگر نیز منتقل کرد تا در صورت نیاز، با استفاده از آن اشاره‌گر، تابع فراخوانی شود

اشاره‌گر به توابع

اشاره‌گر به توابع:

- یک برنامه کاربردی، برای اجرا، ابتدا داخل حافظه اصلی کامپیوتر قرار می‌گیرد و سپس اجرا می‌شود.
- لذا، هر دستور در برنامه نیز در محلی از حافظه ذخیره می‌شود و به همین دلیل، اشاره‌گر به دستورات مختلف برنامه نیز منطقی است.
- یک اشاره‌گر به تابع، عملاً محل شروع دستورات تابع در حافظه است.
- با فراخوانی تابع، اجرای برنامه به آن آدرس منتقل می‌شود و دستورات از آن محل به ترتیب، تا رسیدن به پایان تابع، اجرا می‌شود.
- مشابه آرایه، نام هر تابع، حاوی اشاره‌گر به آن تابع است.
- از اشاره‌گر به تابع می‌توان برای فراخوانی تابع استفاده کرد.
- همچنین اشاره‌گر به تابع را می‌توان به عنوان پارامتر، به توابع دیگر نیز منتقل کرد تا در صورت نیاز، با استفاده از آن اشاره‌گر، تابع فراخوانی شود

اشاره‌گر به توابع

اشاره‌گر به توابع:

- یک برنامه کاربردی، برای اجرا، ابتدا داخل حافظه اصلی کامپیوتر قرار می‌گیرد و سپس اجرا می‌شود.
- لذا، هر دستور در برنامه نیز در محلی از حافظه ذخیره می‌شود و به همین دلیل، اشاره‌گر به دستورات مختلف برنامه نیز منطقی است.
- یک اشاره‌گر به تابع، عملاً محل شروع دستورات تابع در حافظه است.
- با فراخوانی تابع، اجرای برنامه به آن آدرس منتقل می‌شود و دستورات از آن محل به ترتیب، تا رسیدن به پایان تابع، اجرا می‌شود.
- مشابه آرایه، نام هر تابع، حاوی اشاره‌گر به آن تابع است.
- از اشاره‌گر به تابع می‌توان برای فراخوانی تابع استفاده کرد.
- همچنین اشاره‌گر به تابع را می‌توان به عنوان پارامتر، به توابع دیگر نیز منتقل کرد تا در صورت نیاز، با استفاده از آن اشاره‌گر، تابع فراخوانی شود

اشاره‌گر به توابع

اشاره‌گر به توابع:

- یک برنامه کاربردی، برای اجرا، ابتدا داخل حافظه اصلی کامپیوتر قرار می‌گیرد و سپس اجرا می‌شود.
- لذا، هر دستور در برنامه نیز در محلی از حافظه ذخیره می‌شود و به همین دلیل، اشاره‌گر به دستورات مختلف برنامه نیز منطقی است.
- یک اشاره‌گر به تابع، عملاً محل شروع دستورات تابع در حافظه است.
- با فراخوانی تابع، اجرای برنامه به آن آدرس منتقل می‌شود و دستورات از آن محل به ترتیب، تا رسیدن به پایان تابع، اجرا می‌شود.
- مشابه آرایه، نام هر تابع، حاوی اشاره‌گر به آن تابع است.
- از اشاره‌گر به تابع می‌توان برای فراخوانی تابع استفاده کرد.
- همچنین اشاره‌گر به تابع را می‌توان به عنوان پارامتر، به توابع دیگر نیز منتقل کرد تا در صورت نیاز، با استفاده از آن اشاره‌گر، تابع فراخوانی شود

اشاره‌گر به توابع

اشاره‌گر به توابع:

- یک برنامه کاربردی، برای اجرا، ابتدا داخل حافظه اصلی کامپیوتر قرار می‌گیرد و سپس اجرا می‌شود.
- لذا، هر دستور در برنامه نیز در محلی از حافظه ذخیره می‌شود و به همین دلیل، اشاره‌گر به دستورات مختلف برنامه نیز منطقی است.
- یک اشاره‌گر به تابع، عملاً محل شروع دستورات تابع در حافظه است.
- با فراخوانی تابع، اجرای برنامه به آن آدرس منتقل می‌شود و دستورات از آن محل به ترتیب، تا رسیدن به پایان تابع، اجرا می‌شود.
- مشابه آرایه، نام هر تابع، حاوی اشاره‌گر به آن تابع است.
- از اشاره‌گر به تابع می‌توان برای فراخوانی تابع استفاده کرد.
- همچنین اشاره‌گر به تابع را می‌توان به عنوان پارامتر، به توابع دیگر نیز منتقل کرد تا در صورت نیاز، با استفاده از آن اشاره‌گر، تابع فراخوانی شود

اشاره‌گر به توابع

اشاره‌گر به توابع:

- یک برنامه کاربردی، برای اجرا، ابتدا داخل حافظه اصلی کامپیوتر قرار می‌گیرد و سپس اجرا می‌شود.
- لذا، هر دستور در برنامه نیز در محلی از حافظه ذخیره می‌شود و به همین دلیل، اشاره‌گر به دستورات مختلف برنامه نیز منطقی است.
- یک اشاره‌گر به تابع، عملاً محل شروع دستورات تابع در حافظه است.
- با فراخوانی تابع، اجرای برنامه به آن آدرس منتقل می‌شود و دستورات از آن محل به ترتیب، تا رسیدن به پایان تابع، اجرا می‌شود.
- مشابه آرایه، نام هر تابع، حاوی اشاره‌گر به آن تابع است.
- از اشاره‌گر به تابع می‌توان برای فراخوانی تابع استفاده کرد.
- همچنین اشاره‌گر به تابع را می‌توان به عنوان پارامتر، به توابع دیگر نیز منتقل کرد تا در صورت نیاز، با استفاده از آن اشاره‌گر، تابع فراخوانی شود

اشاره‌گر به توابع

اشاره‌گر به توابع: (ادامه)

- برای نگهداری اشاره‌گر به یک تابع، نیاز به متغیر اشاره‌گری داریم.
- مثال: برای نگهداری اشاره‌گر به یک تابع که یک پارامتر ورودی از نوع int دارد و مقدار بازگشتی آن از نوع void است، از دستور void (*foo)(int); استفاده می‌کنیم.
- برای قرار دادن اشاره‌گر به تابع func در این متغیر اشاره‌گری می‌توان از دستور foo=func; و یا foo=&func; استفاده کرد.
- برای فراخوانی تابع با پارامتر ورودی ۲ با استفاده از اشاره‌گر به آن، از دستور foo(2); و یا (*foo)(2); استفاده می‌کنیم.
- اشاره‌گر به توابع، کاربردهای متعددی دارند.

اشاره‌گر به توابع

اشاره‌گر به توابع: (ادامه)

- برای نگهداری اشاره‌گر به یک تابع، نیاز به متغیر اشاره‌گری داریم.
- مثال: برای نگهداری اشاره‌گر به یک تابع که یک پارامتر ورودی از نوع int دارد و مقدار بازگشتی آن از نوع void است، از دستور void (*foo)(int); استفاده می‌کنیم.
- برای قرار دادن اشاره‌گر به تابع func در این متغیر اشاره‌گری می‌توان از دستور foo=func; و یا foo=&func; استفاده کرد.
- برای فراخوانی تابع با پارامتر ورودی ۲ با استفاده از اشاره‌گر به آن، از دستور foo(2); و یا (*foo)(2); استفاده می‌کنیم.
- اشاره‌گر به توابع، کاربردهای متعددی دارند.

اشاره‌گر به توابع

اشاره‌گر به توابع: (ادامه)

- برای نگهداری اشاره‌گر به یک تابع، نیاز به متغیر اشاره‌گری داریم.
- مثال: برای نگهداری اشاره‌گر به یک تابع که یک پارامتر ورودی از نوع int دارد و مقدار بازگشتی آن از نوع void است، از دستور ;(int)(*foo) void استفاده می‌کنیم.
- برای قرار دادن اشاره‌گر به تابع func در این متغیر اشاره‌گری می‌توان از دستور ;foo=func و یا ;foo=&func استفاده کرد.
- برای فراخوانی تابع با پارامتر ورودی ۲ با استفاده از اشاره‌گر به آن، از دستور ;(2)foo و یا ;(2)(*foo) استفاده می‌کنیم.
- اشاره‌گر به توابع، کاربردهای متعددی دارند.

اشاره‌گر به توابع

اشاره‌گر به توابع: (ادامه)

- برای نگهداری اشاره‌گر به یک تابع، نیاز به متغیر اشاره‌گری داریم.
- مثال: برای نگهداری اشاره‌گر به یک تابع که یک پارامتر ورودی از نوع int دارد و مقدار بازگشتی آن از نوع void است، از دستور ;(int) (*foo) void استفاده می‌کنیم.
- برای قرار دادن اشاره‌گر به تابع func در این متغیر اشاره‌گری می‌توان از دستور ;foo=func و یا ;foo=&func استفاده کرد.
- برای فراخوانی تابع با پارامتر ورودی ۲ با استفاده از اشاره‌گر به آن، از دستور ;(2)foo و یا ;(2)(*foo) استفاده می‌کنیم.
- اشاره‌گر به توابع، کاربردهای متعددی دارند.

اشاره‌گر به توابع

اشاره‌گر به توابع: (ادامه)

- برای نگهداری اشاره‌گر به یک تابع، نیاز به متغیر اشاره‌گری داریم.
- مثال: برای نگهداری اشاره‌گر به یک تابع که یک پارامتر ورودی از نوع int دارد و مقدار بازگشتی آن از نوع void است، از دستور void (*foo)(int); استفاده می‌کنیم.
- برای قرار دادن اشاره‌گر به تابع func در این متغیر اشاره‌گری می‌توان از دستور foo=func; و یا foo=&func; استفاده کرد.
- برای فراخوانی تابع با پارامتر ورودی ۲ با استفاده از اشاره‌گر به آن، از دستور foo(2); و یا (*foo)(2); استفاده می‌کنیم.
- اشاره‌گر به توابع، کاربردهای متعددی دارند.

اشاره‌گر به توابع

```
1  #include <iostream>
2  using namespace std;
3  bool ascending(double a, double b)
4  {
5       if (a>=b)
6            return  true;
7       else
8            return  false;
9  }
10
11
12 bool descending(double a, double b)
13 {
14      if (a<=b)
15           return  true;
16      else
17           return  false;
18 }
```

اشاره‌گر به توابع

```
1  void Sort(double A[500], int n, bool (*comparison)(double, double))
2  {
3     int i,j;
4     double B;
5     for(i=0;i<n;i++)
6        for(j=i+1;j<n;j++)
7           if (comparison(A[i],A[j]))
8           {
9              B=A[i];A[i]=A[j];A[j]=B;
10          }
11    return;
12 }
13 void input_Array(double A[500],int n)
14 {
15    int i;
16    for(i=0;i<n;i++)
17    {
18       cout<<"Enter Next Number:";
19       cin>>A[i];
20    }
21    return;
22 }
```

اشاره‌گر به توابع

```
1  void output_Array(double A[500],int n)
2  {
3    int  i ;
4    for ( i=0;i<n;i++)
5    {
6        cout<<A[i ]<<",";
7    }
8    return ;
9  }
10  int  main()
11  {
12   int  i ,n;
13   double A[500];
14   cout<<"Enter Size:";
15   cin>>n;
16   input_Array(A,n);
17   cout<<"\n Ascending ***\n";
18   Sort(A,n,  ascending);
19   output_Array(A,n);
20   cout<<"\n Descending ***\n";
21   Sort(A,n,  descending);
22   output_Array(A,n);
23  }
```

متغیرهای const

متغیرهای const:

- به طور معمول، مقدار هر متغیر در برنامه می‌تواند تغییر کند.
- اما امکان‌پذیر است که امکان تغییر مقدار ذخیره شده در یک متغیر را به برنامه‌نویس نداد.
- `const int x=5;`
- شایع‌ترین کاربرد این کلمه کلیدی: تعریف پارامترهای توابع
- اگر یکی از پارامترهای تابع، یک آرایه باشد، فراخوانی مربوطه با ارجاع است و اگر تغییراتی در تابع روی آرایه داده شود، این تغییرات روی آرایه اصلی نیز اعمال خواهد شد.
- برای ندادن اجازه تغییر، در لیست پارامترهای تابع، جلو تعریف این آرایه، از کلمه کلیدی const استفاده کنید.
- عملکرد: پیام خطا در زمان کامپایل برنامه، در صورت تغییر متغیرهای const.

متغیرهای const

متغیرهای const:

- به طور معمول، مقدار هر متغیر در برنامه می‌تواند تغییر کند.
- اما امکان‌پذیر است که امکان تغییر مقدار ذخیره شده در یک متغیر را به برنامه‌نویس نداد.
- `const int x=5;`
- شایع‌ترین کاربرد این کلمه کلیدی: تعریف پارامترهای توابع
- اگر یکی از پارامترهای تابع، یک آرایه باشد، فراخوانی مربوطه با ارجاع است و اگر تغییراتی در تابع روی آرایه داده شود، این تغییرات روی آرایه اصلی نیز اعمال خواهد شد.
- برای ندادن اجازه تغییر، در لیست پارامترهای تابع، جلو تعریف این آرایه، از کلمه کلیدی const استفاده کنید.
- عملکرد: پیام خطا در زمان کامپایل برنامه، در صورت تغییر متغیرهای const.

متغیرهای const

متغیرهای const:

- به طور معمول، مقدار هر متغیر در برنامه می‌تواند تغییر کند.
- اما امکان‌پذیر است که امکان تغییر مقدار ذخیره شده در یک متغیر را به برنامه‌نویس نداد.
- `const int x=5;`
- شایع‌ترین کاربرد این کلمه کلیدی: تعریف پارامترهای توابع
- اگر یکی از پارامترهای تابع، یک آرایه باشد، فراخوانی مربوطه با ارجاع است و اگر تغییراتی در تابع روی آرایه داده شود، این تغییرات روی آرایه اصلی نیز اعمال خواهد شد.
- برای ندادن اجازه تغییر، در لیست پارامترهای تابع، جلو تعریف این آرایه، از کلمه کلیدی const استفاده کنید.
- عملکرد: پیام خطا در زمان کامپایل برنامه، در صورت تغییر متغیرهای const.

متغیرهای const

متغیرهای const:

- به طور معمول، مقدار هر متغیر در برنامه می‌تواند تغییر کند.
- اما امکان‌پذیر است که امکان تغییر مقدار ذخیره شده در یک متغیر را به برنامه‌نویس نداد.
- `const int x=5;`
- شایع‌ترین کاربرد این کلمه کلیدی: تعریف پارامترهای توابع
- اگر یکی از پارامترهای تابع، یک آرایه باشد، فراخوانی مربوطه با ارجاع است و اگر تغییراتی در تابع روی آرایه داده شود، این تغییرات روی آرایه اصلی نیز اعمال خواهد شد.
- برای ندادن اجازه تغییر، در لیست پارامترهای تابع، جلو تعریف این آرایه، از کلمه کلیدی const استفاده کنید.
- عملکرد: پیام خطا در زمان کامپایل برنامه، در صورت تغییر متغیرهای const.

متغیرهای const

متغیرهای const:

- به طور معمول، مقدار هر متغیر در برنامه می‌تواند تغییر کند.
- اما امکان‌پذیر است که امکان تغییر مقدار ذخیره شده در یک متغیر را به برنامه‌نویس نداد.
- `const int x=5;`
- شایع‌ترین کاربرد این کلمه کلیدی: تعریف پارامترهای توابع
- اگر یکی از پارامترهای تابع، یک آرایه باشد، فراخوانی مربوطه با ارجاع است و اگر تغییراتی در تابع روی آرایه داده شود، این تغییرات روی آرایه اصلی نیز اعمال خواهد شد.
- برای ندادن اجازه تغییر، در لیست پارامترهای تابع، جلو تعریف این آرایه، از کلمه کلیدی const استفاده کنید.
- عملکرد: پیام خطا در زمان کامپایل برنامه، در صورت تغییر متغیرهای const.

متغیرهای const

متغیرهای const:

- به طور معمول، مقدار هر متغیر در برنامه می‌تواند تغییر کند.
- اما امکان‌پذیر است که امکان تغییر مقدار ذخیره شده در یک متغیر را به برنامه‌نویس نداد.
- `const int x=5;`
- شایع‌ترین کاربرد این کلمه کلیدی: تعریف پارامترهای توابع
- اگر یکی از پارامترهای تابع، یک آرایه باشد، فراخوانی مربوطه با ارجاع است و اگر تغییراتی در تابع روی آرایه داده شود، این تغییرات روی آرایه اصلی نیز اعمال خواهد شد.
- برای ندادن اجازه تغییر، در لیست پارامترهای تابع، جلو تعریف این آرایه، از کلمه کلیدی const استفاده کنید.
- عملکرد: پیام خطا در زمان کامپایل برنامه، در صورت تغییر متغیرهای const.

متغیرهای const

متغیرهای const:

- به طور معمول، مقدار هر متغیر در برنامه می‌تواند تغییر کند.
- اما امکان‌پذیر است که امکان تغییر مقدار ذخیره شده در یک متغیر را به برنامه‌نویس نداد.
- `const int x=5;`
- شایع‌ترین کاربرد این کلمه کلیدی: تعریف پارامترهای توابع
- اگر یکی از پارامترهای تابع، یک آرایه باشد، فراخوانی مربوطه با ارجاع است و اگر تغییراتی در تابع روی آرایه داده شود، این تغییرات روی آرایه اصلی نیز اعمال خواهد شد.
- برای ندادن اجازه تغییر، در لیست پارامترهای تابع، جلو تعریف این آرایه، از کلمه کلیدی const استفاده کنید.
- عملکرد: پیام خطا در زمان کامپایل برنامه، در صورت تغییر متغیرهای const.

متغیرهای const

متغیرهای اشاره‌گری const:

- در خصوص متغیرهای اشاره‌گری، سه برداشت از یک متغیر اشاره‌گری ثابت می‌توان داشت.
- برداشت اول: یک اشاره‌گر در آن متغیر قرار می‌گیرد و در ادامه برنامه، امکان تغییر اشاره‌گر ذخیره شده در متغیر اشاره‌گری ثابت نیست. به عنوان مثال، دستور int * const xptr=&x; متغیر xptr چنین عملکردی دارد.
- برداشت دوم این است که متغیر اشاره‌گری به صورتی تعریف شود که امکان تغییر مقدار حافظهٔ آدرسی که در این متغیر اشاره‌گری ذخیره شده است، با استفاده از یا از طریق این اشاره‌گر وجود نداشته باشد.
  مثال: دستور const int * xptr;
- برداشت سوم، مجموع دو برداشت فوق است. برای تعریف متغیر اشاره‌گری با این خاصیت می‌توان از دستور const int * const xptr=&x; استفاده کرد.

متغیرهای const

متغیرهای اشاره‌گری const:

- در خصوص متغیرهای اشاره‌گری، سه برداشت از یک متغیر اشاره‌گری ثابت می‌توان داشت.
- برداشت اول: یک اشاره‌گر در آن متغیر قرار می‌گیرد و در ادامه برنامه، امکان تغییر اشاره‌گر ذخیره شده در متغیر اشاره‌گری ثابت نیست. به عنوان مثال، دستور const int * xptr=&x; متغیر xptr چنین عملکردی دارد.
- برداشت دوم این است که متغیر اشاره‌گری به صورتی تعریف شود که امکان تغییر مقدار حافظهٔ آدرسی که در این متغیر اشاره‌گری ذخیره شده است، با استفاده از یا از طریق این اشاره‌گر وجود نداشته باشد.
  مثال: دستور const int * xptr;
- برداشت سوم، مجموع دو برداشت فوق است. برای تعریف متغیر اشاره‌گری با این خاصیت می‌توان از دستور const int * const xptr=&x; استفاده کرد.

متغیرهای const

متغیرهای اشاره‌گری const:

- در خصوص متغیرهای اشاره‌گری، سه برداشت از یک متغیر اشاره‌گری ثابت می‌توان داشت.
- برداشت اول: یک اشاره‌گر در آن متغیر قرار می‌گیرد و در ادامه برنامه، امکان تغییر اشاره‌گر ذخیره شده در متغیر اشاره‌گری ثابت نیست. به عنوان مثال، دستور const int * xptr=&x; متغیر xptr چنین عملکردی دارد.
- برداشت دوم این است که متغیر اشاره‌گری به صورتی تعریف شود که امکان تغییر مقدار حافظهٔ آدرسی که در این متغیر اشاره‌گری ذخیره شده است، با استفاده از یا از طریق این اشاره‌گر وجود نداشته باشد.
  مثال: دستور const int * xptr;
- برداشت سوم، مجموع دو برداشت فوق است. برای تعریف متغیر اشاره‌گری با این خاصیت می‌توان از دستور const int * const xptr=&x; استفاده کرد.

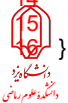
متغیرهای const

متغیرهای اشاره‌گری const:

- در خصوص متغیرهای اشاره‌گری، سه برداشت از یک متغیر اشاره‌گری ثابت می‌توان داشت.
- برداشت اول: یک اشاره‌گر در آن متغیر قرار می‌گیرد و در ادامه برنامه، امکان تغییر اشاره‌گر ذخیره شده در متغیر اشاره‌گری ثابت نیست. به عنوان مثال، دستور const int * xptr=&x; متغیر xptr چنین عملکردی دارد.
- برداشت دوم این است که متغیر اشاره‌گری به صورتی تعریف شود که امکان تغییر مقدار حافظهٔ آدرسی که در این متغیر اشاره‌گری ذخیره شده است، با استفاده از یا از طریق این اشاره‌گر وجود نداشته باشد.
  مثال: دستور const int * xptr;
- برداشت سوم، مجموع دو برداشت فوق است. برای تعریف متغیر اشاره‌گری با این خاصیت می‌توان از دستور const int * const xptr=&x; استفاده کرد.

متغیرهای const

```
1  #include <iostream>
2  using namespace std;
3  int  main(void)
4  {
5     int  x,y,z;
6     const int  *ptr1=&x;
7     int  * const ptr2=&y;
8     const int  * const ptr3=&z;
9     //  *ptr1=5;  //  Error:  assignment of read-only location
10    ptr1=&y;
11    *ptr2=10;
12    //  ptr2=&x;  //  Error:  assignment of read-only variable
13    //  *ptr3=15;  //  Error:  assignment of read-only location
14    //  ptr3=&x;  //  Error:  assignment of read-only variable
15    return  0;
16 }
```




باآرزوی موفقیت
وبهروزی