بنام خدا

برنامه نویسی به زبانC

.یک زبان برنامه نویسی قدرتمند است که مزایای بسیاری، نسبت سایر زبان‌ها دارد

فهرستی از برخی مزایای زبان برنامه نویسی

زبانی چندمنظوره است که می‌توان برای کاربردهای مختلف از آن استفاده کرد

زبانی بسیار کارآمد است که می‌توان کدهایی را در

زبانی قابل حمل است، به این معنی که کدی را که در زبان آن توسعه داد توسعه داده شده است

می‌توان به راحتی کامپایل و بر روی سیستم عامل‌های مختلف اجرا کرد

زبان برنامه نویسی زبانی کاملاً جاافتاده با جامعه کاربری بزرگ و فعال از توسعه‌دهندگان به حساب می‌آید که به طور مداوم درحال بهبود و ایجاد ابزارها و کتابخانه‌های جدید برای آن هستند

استفاده از زبان Cو کاربردهای کلیدی آن

زبان برنامه نویسی سی یکی از قدیمی‌ترین و اساسی‌ترین زبان‌های برنامه نویسی محسوب می‌شود و در C زبانی سریع و قابل حمل، گسترده است این زبان، .سراسر جهان به طور گسترده مورد استفاده قرار می‌گیرد زبانی سطح میانی به حساب می‌آید که هم مزایای زبان سطح پایین و هم قابلیت‌های زبان‌های سطح بالا را .دارا است

تاریخچه

تحولات اولیه

PDP-7 بر روی کاملاً مرتبط است، که در ابتدا با زبان اسمبلی با توسعه سیستم عامل یونیکس C منشأ پیاده‌سازی شده و چندین ایده از همکاران را دربر گرفته‌است. سرانجام، آنها و کن تامپسون توسط دنیس ریچی

یونیکس نیز به زبان PDP-11 منتقل کنند. نسخه اصلی PDP-11 را به تصمیم گرفتند سیستم عامل اسمبلی تولید شده‌است.

را برای ایجاد برنامه‌های کاربردی برای سیستم عامل جدید می‌خواست. در تامپسون یک زبان برنامه‌نویسی بسازد، اما خیلی زود این ایده را رها کرد. در عوض، او یک نسخه برش Fortran ابتدا، او سعی کرد تا کامپایلر BCPL که اخیراً توسعه یافته بود، ایجاد کرد. توضیحات رسمی BCPLخورده از زبان برنامه‌نویسی سیستم‌های را مشابه اما کمی ساده‌تر تولید Bرا اصلاح و کم‌حرف‌تر و (syntax) در آن زمان موجود نبود و تامپسون نحو نمی‌توانست از ویژگی‌های B نوشته شدند زیرا خیلی کند بود و B می‌کند با این حال، کمترین ابزارها نهایتاً در بهره ببرد مانند آدرس پذیری بایت PDP-11.

و برخی C شد. کامپایلر C کرد، که منجر به ایجاد زبان جدید B شروع به بهبود در سال ۱۹۷۲، دنیس ریچی از برنامه‌های کاربردی ساخته شده با آن در نسخه ۲ یونیکس گنجانده شده‌است. در نسخه ۴ یونیکس، که در C دوباره اجرا شد. در این زمان، زبان C نوامبر ۱۹۷۳ منتشر شد، هسته یونیکس به‌طور گسترده‌ای در ویژگی‌های قدرتمندی مانند انواع ساختار را به دست آورده بود.

K&R C

را منتشر کردند. این کتاب C در سال ۱۹۷۸، برایان کرنیگان و دنیس ریچی چاپ اول کتاب زبان برنامه‌نویسی شناخته شده بود، سالها به عنوان مشخصات غیررسمی زبان مورد C برای برنامه نویسان K&R که به عنوان شناخته می‌شود. چاپ دوم کتاب "K&R C" که توصیف می‌کند معمولاً با عنوان C استفاده قرار گرفت. نسخه است که در زیر شرح داده شده‌است ANSI C شامل استاندارد بعدی.

ANSI C and ISO C

برای طیف گسترده‌ای از رایانه‌های اصلی، مینی رایانه و میکرو رایانه‌ها C در اواخر دهه ۱۹۷۰ و ۱۹۸۰، نسخه پیاده‌سازی شد، زیرا محبوبیت آن به میزان قابل توجهی افزایش یافت IBM PC از جمله.

تشکیل داد تا مشخصات X3J11 کمیته ای با نام (ANSI) در سال ۱۹۸۳، مؤسسه استاندارد ملی آمریکا بر روی اجرای یونیکس؛ با این حال، بخش غیرقابل C براساس استاندارد X3J11 .را ایجاد کند C استاندارد POSIX واگذار شد تا پایه ای برای استاندارد IEEE 1003 به گروه کاری C حمل از کتابخانه یونیکس "C زبان برنامه نویسی" ANSI X3.159-1989 با عنوان C باشد. در سال ۱۹۸۹، استاندارد 1988 گفته می‌شود C89 یا بعضی اوقات C ، استانداردANSI C تصویب شد. این نسخه از زبان اغلب به عنوان.

(ISO) با تغییرات قالب بندی) توسط سازمان بین‌المللی استانداردسازی) ANSI Cدر سال ۱۹۹۰، استاندارد نیز نامیده می‌شود؛ بنابراین، اصطلاحات C90 تصویب شد، که گاهی ISO / IEC 9899: 1990 به عنوان به همان زبان برنامه‌نویسی اشاره دارند "C90" و "C89".

C99

در ۱۹۹۹ شد، ISO / IEC 9899: 1999 در اواخر دهه ۱۹۹۰ بازنگری شد و منجر به انتشار C استاندارد گفته می‌شود. از آن زمان سه بار توسط غلط‌های فنی اصلاح شده‌است "C99" که معمولاً به آن.

و int long long از جمله) جدید چندین ویژگی جدید از جمله توابع درون خطی، چندین نوع داده C99 ، آرایه‌های با طول متغیر و اعضای آرایه انعطاف‌پذیر، پشتیبانی (یک نوع مختلط برای نشان دادن اعداد مختلط نقطه شناور، پشتیبانی از ماکرو متغیر را معرفی کرد؛ و پشتیبانی از نظرات تک IEEE 754 بهبود یافته از بسیاری از این موارد قبلاً به عنوان پسوند در چندین .++ C یا BCPL خطی که با // شروع می‌شود، مانند اجرا شده بودند C کامپایلر.

سازگاری پسرو دارد، اما از بعضی جهات سخت‌تر است. به ویژه، بیانیه‌ای که فاقد C90 در اکثر موارد با C99 یک مشخص کننده نوع است، دیگر به‌طور ضمنی فرض شده‌است. یک کلان استاندارد در C99 تعریف شده‌است تا نشان دهد که پشتیبانی Lبا مقدار ۱۹۹۹۰۱ __STDC_VERSION__ C99 اکنون از بسیاری از ویژگی‌های جدید C و سایر کامپایلرهای GCC , Solaris Studio .دسترس است را C99 و قسمت‌هایی از C89 ، استانداردMicrosoft Visual C++ در C پشتیبانی می‌کنند. کامپایلر لازم است، پیاده‌سازی می‌کند C ++ 11 که برای سازگاری با.

C11

تا انتشار رسمی "C1X" ، به‌طور غیررسمی به نامC در سال ۲۰۰۷، کار بر روی تجدید نظر در مورد استاندارد دستورالعمل‌هایی را برای محدود کردن استفاده از C آن در تاریخ ۰۸-۱۲-۲۰۱۱ آغاز شد. کمیته استاندارد ویژگی‌های جدید که توسط آزمایش‌های موجود آزمایش نشده‌اند، اتخاذ کرده‌است.

اضافه می‌کند و کتابخانه، از جمله ماکرو نوع عمومی، ساختارهای C چندین ویژگی جدید به C11 استاندارد ناشناس، پشتیبانی بهبود یافته یونیکد، عملیات اتمی، چند رشته‌ای و عملکردهای محدود شده را بررسی C++ را به اختیاری تبدیل کرده و سازگاری با C99 می‌کند. همچنین برخی از بخش‌های موجود در کتابخانه تعریف شده‌است تا نشان Lبه عنوان ۲۰۱۱۱۲ __STDC_VERSION__ را بهبود می‌بخشد. کلان استاندارد در دسترس است C11 دهد که پشتیبانی.

C18

است. این معرفی هیچ ویژگی C که در ژوئن سال ۲۰۱۸ منتشر شده استاندارد فعلی زبان برنامه‌نویسی C18 را ارائه می‌دهد. کلان استاندارد C11 جدید زبان نداشت، فقط اصلاحات فنی و شفاف‌سازی در مورد نقص .تعریف شده‌است Lبا عنوان ۲۰۱۷۱۰ __STDC_VERSION__

نشانگرها]انواع داده‌ها

، امکان استفاده از اشاره‌گرهاست. اشاره‌گرها کارایی، قدرت C یکی از مهم‌ترین قابلیت‌های زبان برنامه را بیشتر می‌نمایند. علاوه بر فراهم آوردن امکان نوشتن کدهای برنامهٔ کوتاه‌تر، غنی‌تر و انعطاف‌پذیری و کاراتر، گاهی مواردی پیش می‌آید که انجام محاسبات مورد نیاز برنامه تنها توسط اشاره گرها امکان‌پذیر زیاد استفاده می‌گردد. اگر چه در ظاهر کار با اشاره‌گرها C می‌باشد. به همین دلیل، از این توانایی در برنامه‌های مشکل است و درک برنامه‌ای که از آنها استفاده شده‌است ساده نیست، اما در حقیقت اگر از آنها به درستی .استفاده گردد، وضوح و سادگی برنامه افزایش می‌یابد

اشاره‌گر چیست؟

روش مستقیم دسترسی به حافظه برای ذخیره‌سازی مقداری در آن، یا بازیابی محتوای ذخیره شده، استفاده از نام متغیر است. همین که متغیری معرفی شد، سیستم بر اساس نوع اعلام شده، تعداد بایت لازم را به آن اختصاص می‌دهد و آدرس متغیر، شماره اولین بایت از مجموعهٔ اختصاص یافته‌است. پس از آن به راحتی از طریق نام متغیر به محل موردنظر دسترسی یافته، محتوای آن پردازش می‌گردد. اما گاهی لازم می‌شود به جای نام، آدرس متغیر در اختیار برنامه‌نویس قرار گیرد تا از طریق آن دستیابی به محل مربوط صورت گیرد. در زبان C ، به راحتی می‌توان آدرس یک متغیر را در اختیار داشت و مقدارش را در حافظه ذخیره نمود تا در زمان نیاز، توسط آن به صورت غیرمستقیم به محل موردنظر دست یافت. برای این منظور از متغیر اشاره‌گر استفاده .می‌گردد. متغیر اشاره‌گر، متغیری است که محتوای آن آدرس یک متغیر دیگر است

آرایه‌ها

بسیاری از موارد پیش می‌آید که برنامه‌نویس نیاز به استفاده از تعداد زیادی متغیر پیدا می‌کند؛ مثلاً اگر را برای استفاده ذخیره کنیم، به صد متغیر نیاز داریم. تعریف این بخواهیم جملات ۱ تا ۱۰۰ سری فیبوناچی صد متغیر به صورت مستقل و با نام‌های جداگانه کاری سخت و طاقت‌فرساست و البته معقول نیست. به همین .قابلیتی پیش‌بینی شده تا بتواند تعداد دلخواهی متغیر از یک نوع را به راحتی ایجاد کرد C خاطر در زبان

می‌گویند. (Array) به این متغیرها که در حافظه پشت سر هم قرار می‌گیرند و همگی از یک نوع هستند آرایه آرایه‌ها کاربردهای بسیار زیادی دارند و همانند دنباله‌ها در ریاضی عمل می‌کنند. همان‌طور که در ریاضی برای

نیز برای مشخص کردن یک متغیر خاص C مشخص کردن یک جمله از دنباله از اندیس استفاده می‌کنیم، در .از اندیس استفاده می‌شود

همان‌طور که مشاهده می‌کنید خانهٔ پنجم شمارهٔ ۴ دارد و این بدین خاطر است که خانه‌ها از شمارهٔ ۰ تا شماره‌گذاری می‌شود. پس باید به این مورد دقت کرد. مورد دیگر این که (تعداد خانه‌هاست n که) n-1 .شمارهٔ خانه در [] قرار می‌گیرد. درون علامت آکلاد می‌توان عبارت نیز قرار داد

خانه‌های آرایه پشت سر هم قرار می‌گیرند. اندازهٔ هر خانه به همان‌طور که گفته شد در حافظهٔ رایانه ، در کل ۲۰ بایت از array دو بایت اشغال کند آرایهٔ int اندازهٔ نوع تعریف شده‌است؛ مثلاً در مثال قبل اگر حافظه اشغال خواهد کرد. این که خانه‌ها پشت سر هم قرار می‌گیرند ویژگی کارایی است که در بحث اشاره‌گرها .به کار می‌آید

مدیریت حافظه

و مواردی است یکی از مهمترین کارکردهای یک زبان برنامه‌نویسی، فراهم آوردن امکاناتی برای مدیریت حافظه :سه روش مشخص برای اختصاص حافظه برای اشیاء ارائه می‌دهد C .که در حافظه ذخیره می‌شوند

- **تخصیص حافظه استاتیک** فضایی برای جسم در زمان کامپایل در دودویی فراهم می‌شود. این اشیاء تا .حدودی (یا طول عمر) دارند تا زمانی که باینری که شامل آنها است در حافظه بارگذاری شود
- **تخصیص خودکار حافظه** اشیاء موقتی را می‌توان در پشته ذخیره کرد و پس از خارج شدن از بلوکی که : .در آن اعلام شده‌است، این فضای به‌طور خودکار آزاد و قابل استفاده مجدد می‌شود
  - malloc بلوک‌های حافظه با اندازه دلخواه را می‌توان در زمان اجرا با استفاده از توابع کتابخانه مانند از منطقه ای از حافظه به نام پشته درخواست کرد. این بلوک‌ها تا زمانی که با :**تخصیص حافظه پویا** استفاده مجدد از عملکرد مجدد کتابخانه یا مجدداً آزاد شوند، برای استفاده مجدد آزاد می‌شوند
- **تخصیص حافظه خودکار** ذخیره شوند، این فضای اختصاص داده اشیا موقتی می‌توانند بر روی پشته : شده به اشیا به صورت اتوماتیک پس از خارج شدن از بلاکی که اشیا در آنها تعریف شده‌اند، آزاد و دوباره .قابل استفاده خواهد بود
  - malloc() اندازه‌های اختیاری از بلاک‌های حافظه می‌توانند توسط توابع کتابخانه‌ای همانند تابع : در هنگام اجرای برنامه درخواست بشود. این **تخصیص حافظه پویا** از ناحیه‌ای از رم موسوم به هیپ به سیستم بازگردانده نشوند در حافظه باقی می‌مانند. free() بلاک‌های حافظه تا زمانی‌که توسط تابع malloc() برای تخصیص حافظه به صورت پویا (دینامیک) باید آدرس بلوک حافظه‌ای که توسط تابع .ذخیره کنیم گرفته می‌شود را در یک اشاره گر

این سه رویکرد در موقعیتهای مختلف مناسب است و دارای تبعات مختلفی است. به عنوان مثال، تخصیص حافظه استاتیک اختصاص کمی به سربار دارد، تخصیص خودکار ممکن است کمی بیشتر از سربار باشد و تخصیص حافظه پویا به‌طور بالقوه می‌تواند مقدار زیادی از سربار را برای تخصیص و جابجایی داشته باشد. ماهیت پایدار اشیاء استاتیک برای حفظ اطلاعات حالت در طول فراخوانی‌های عملکردی مفید است، تخصیص خودکار به راحتی قابل استفاده است اما فضای پشته معمولاً بسیار محدودتر و گذرا از حافظه استاتیک یا فضای پشته است و تخصیص حافظه پویا امکان تخصیص مناسب اشیاء را می‌دهد که اندازه فقط در زمان اجرا شناخته .از هر سه مورد استفاده گسترده‌ای می‌کنند C شده‌است. بیشتر برنامه‌های

در صورت امکان، تخصیص اتوماتیک یا استاتیک معمولاً ساده‌ترین است زیرا ذخیره‌سازی توسط کامپایلر اداره می‌شود، و برنامه‌نویس را از روی خطای بالقوه خطا در اختصاص و آزاد سازی فضای ذخیره‌سازی آزاد می‌کند. با و) این وجود بسیاری از ساختارهای داده می‌توانند در زمان اجرا تغییر کنند و از آنجا که تخصیص استاتیک باید در زمان کامپایل اندازه ثابت داشته باشد، موقعیت‌های بسیاری وجود دارد (C99 تخصیص خودکار قبل از به) ،. آرایه‌های اندازه متغیر نمونه متداول این امر بودندC99 که تخصیص پویا لازم است. قبل از استاندارد برخلاف تخصیص (مراجعه کنید malloc عنوان مثال از آرایه‌های اختصاص داده شده پویا به مقاله در مورد خودکار، که می‌تواند در زمان اجرا با عواقب کنترل نشده از کار بیفتد، عملکردهای تخصیص پویا هنگام ذخیره‌سازی مورد نیاز، نشانه (به صورت مقدار اشاره گر تهی) را برمی‌گردانند. نمی‌توان اختصاص داد (قبل از اینکه برنامه حتی بتواند اجرای آن را شروع کند، معمولاً توسط لینک دهنده یا لودر تشخیص داده می‌شود)

مگر در مواردی که مشخص شده باشد، اشیاء استاتیک حاوی مقادیر نشانگر صفر یا تهی هنگام شروع برنامه هستند. اشیاء اختصاص داده شده به صورت خودکار و پویا فقط درصورتی تنظیم می‌شوند که مقدار اولیه صریحاً مشخص شود. در غیر این صورت آنها در ابتدا مقادیر مشخص نشده‌ای دارند (به‌طور معمول، هر الگوی بیتی که در آن ذخیره می‌شود، حتی ممکن است یک مقدار معتبر برای آن نوع نداشته باشد). اگر برنامه سعی کند به یک مقدار ناشناخته دسترسی پیدا کند، نتایج مشخص نیست. بسیاری از کامپایلرهای مدرن سعی در کشف و .هشدار درباره‌ٔ این مشکل دارند، اما هم مثبت‌های کاذب و هم منفی کاذب می‌تواند رخ دهد

مسئله دیگر این است که تخصیص حافظه پشته باید با کاربرد واقعی آن در هر برنامه همگام سازی شود تا در استفاده مجدد از آن تا حد امکان استفاده شود. به عنوان مثال، اگر تنها اشاره گر برای تخصیص حافظه پنهان از محدوده خارج شود یا مقدار آن را قبل از بازنویسی بازنویسی کرده باشد () فراخوانی شود، پس از آن حافظه نشت a برای استفاده مجدد بعدی قابل بازیابی نیست و در اصل برای برنامه از بین می‌رود، پدیده ای معروف به حافظه در مقابل، امکان آزادسازی حافظه وجود دارد اما همچنان به آن مراجعه می‌شود و منجر به نتایج غیرقابل پیش‌بینی می‌شود. به‌طور معمول، علائم در بخشی از برنامه به دور از خطای واقعی ظاهر می‌شوند، و پیگیری مشکل را دشوار می‌کند. (چنین مواردی در زبانهایی که جمع‌آوری زباله‌های اتوماتیک دارند بهبود یافته‌است)