Výhody jazyka C

Univerzálnost a flexibilita

Jazyk C je střední úrovně, což znamená, že kombinuje vlastnosti jak vysokoúrovňových, tak nízkoúrovňových jazyků. To mu umožňuje být použit jak pro nízkoúrovňové programování, například psaní skriptů pro ovladače a jádra operačního systému, tak pro vyšší úrovně abstrakce, například psaní skriptů pro softwarové aplikace atd.

Strukturovaný přístup

Strukturovaný charakter jazyka C umožňuje programátorům rozdělit složité programy na jednodušší části nazývané funkce. Tento přístup zlepšuje čitelnost kódu a umožňuje efektivnější správu projektů.

Přímý přístup k hardwaru a výkonnost

Díky svému přímému přístupu k hardwarovým rozhraním a možnostem nízkoúrovňového programování je jazyk C ideální volbou pro vývoj aplikací a ovladačů pro vestavěné systémy. K dispozici jsou také optimalizované kompilátory, které umožňují dosáhnout vysokého výkonu.

Citlivost na velikost písmen

Citlivost na velikost písmen znamená, že jazyk C rozlišuje mezi malými a velkými písmeny. To může být klíčové pro identifikátory a proměnné v kódu, kde malé a velké písmena mají různý význam.

Univerzální použití

Jazyk C je obecný programovací jazyk, který lze použít pro širokou škálu aplikací, včetně podnikových aplikací, her, grafiky a vědeckých výpočtů.

Bohatá knihovna a dynamická alokace paměti

K dispozici je rozsáhlá knihovna funkcí v jazyce C, která usnadňuje vývoj různých typů aplikací. Dynamická alokace paměti umožňuje programům efektivně využívat a uvolňovat paměť během běhu.

Efektivita výpočtů

Jazyk C je známý svou schopností rychle implementovat algoritmy a manipulovat s datovými strukturami. To umožňuje rychlé a efektivní výpočty, což je důležité pro výkonné aplikace jako MATLAB a Mathematica.

Nevýhody a omezení jazyka C

Hlavní nevýhody

- Jazyk C nemá garbage collector.
- Jazyk C nepodporuje žádný druh polymorfismu.
- Jazyk C nemá výjimky.
- Jazyk C nepodporuje objektově orientované programování příliš dobře.
- Jazyk C poskytuje pouze omezenou podporu pro zabránění kolizím názvů v prostoru jmen.

Jazyk C nemá garbage collector

Mnoho moderních programovacích jazyků detekuje a automaticky uvolňuje nepřístupná data za vás. Jazyk C to nedělá, takže programátor musí trávit hodně času staráním se o to, kdy a kým budou data alokována funkcí `malloc` uvolněna funkcí `free`. To nejenže vytváří mnoho možností pro chybu, ale také znamená, že určité druhy datových struktur, ve kterých je jedna složka datové struktury ukazována nepředvídatelným počtem dalších složek, je obtížné psát v jazyce C, protože je těžké určit, kdy je bezpečné uvolnit složku. Jazyky s garbage collector em těmto problémům vyhýbají za malý výkonový kompromis. I když existuje garbage collector pro jazyky C/C++, není 100% přenositelný a nemusí fungovat tak dobře jako vestavěný garbage collector.

Jazyk C nepodporuje žádný druh polymorfismu

Polymorfismus je schopnost funkce pracovat s více než jedním typem dat. Nejbližší, co jazyk C může udělat, jsou parametrizované makra, těžké používání `void *` a ukazatelů na funkce jako v `qsort`, nebo různé nešikovné triky, kde se kód automaticky generuje s typovými názvy vyplněnými základní šablonou. Většina moderních programovacích jazyků má nějakou formu podpory polymorfismu, což vám umožňuje psát například obecnou rutinu pro řazení bez použití odchylek od typového systému, jakými jsou `void *`.

Jazyk C nemá výjimky

Výjimky jsou mechanismus pro provádění nestandardních návratů z funkce, když něco selže, které jsou zachyceny pomocí "obslužného programu výjimek", který je často oddělený od kódu zpracovávajícího normální návratové hodnoty a který často může být použit k zachycení výjimek z různých zdrojů. Místo toho vyžaduje jazyk C, aby autoři funkcí vymysleli a zdokumentovali ad-hoc protokol pro označení špatných výsledků pro každou funkci, kterou píšou, a vyžaduje od uživatelů funkcí, aby si pamatovali testovat špatné návratové hodnoty. Většina programátorů je příliš líná, aby to dělala pořád, což vede k nedetekovaným běhovým chybám. Většina moderních programovacích jazyků tento problém řeší.

Jazyk C nepodporuje objektově orientované programování příliš dobře

"Objektově orientovaný"je módní termín s mnoha možnými významy, ale minimálně znamená, že kromě podpory polymorfismu (viz výše), váš jazyk by měl podporovat silnou zapouzdřenost a dědičnost. V jazyce C můžete většinu těchto věcí napodobit, pokud se opravdu snažíte (například pomocí ukazatelů na funkce), ale vždy je možné manipulovat s interními částmi věcí jen kvůli neomezené kontrole, kterou vám jazyk C poskytuje nad prostředím. To se může rychle stát nebezpečným v rozsáhlých softwarových projektech.

Jazyk C poskytuje pouze omezenou podporu pro zabránění kolizím názvů v prostoru jmen

V rozsáhlém programu v jazyce C je nemožné zajistit, že funkce `eat_leftovers`, exportovaná z `leftovers.c`, nebude konfliktní s vaší funkcí `eat_leftovers` v `cannibalism.c`. Mediální řešení je použití delších názvů: `leftovers_eat_leftovers` vs `cannibalism_eat_leftovers`, a lze také hrát hry s ukazateli na funkce a globálními strukturami proměnných, aby se umožnilo něco jako `leftovers.eat_leftovers` vs `cannibalism.eat_leftovers`. Většina moderních programovacích jazyků poskytuje explicitní mechanismus balíčku nebo prostoru jmen, aby umožnila programátorovi ovládat, kdo vidí jaké názvy a kde.

Co opravuje jazyk C++?

Na výše uvedeném seznamu jazyk C++ opravuje všechno kromě chybějícího garbage collectoru.