Programování v jazyce C a C++

Richter[1]
Petyovský[2]

1. března 2015

[1]Ing. Richter Miloslav, Ph.D., UAMT FEKT VUT Brno
[2]Ing. Petyovský Petr, UAMT FEKT VUT Brno

C++
Stručná charakteristika

- Nesdílí normu s C, má vlastní (snaha o maximální shodu)
- Rozšiřuje C o neobjektové a objektové vlastnosti,
- Rozšiřuje C o pokročilé programátorské techniky (šablony, výjimky, ...)
- Koncovky zdrojových souborů jsou cpp. Hlavičkových h nebo hpp.

Přechod od C k C++
Složitější programy v C přináší odlišný programovací styl a ukazují nedostatky jazyka

- Výstup a vstup nejsou typově orientované (printf, scanf) - objektový mechanizmus streamů
- Alokace paměti není typově orientovaná - nový mechanizmus manipulace s pamětí (new, delete - umožňují konstrukci a zánik objektů)
- Není možné mít více funkcí stejného jména (labs, fabs, iabs...) - v C++ je možné (umožňuje využití šablon, streamů ...)
- Při spolupráci více autorů nesmí dojít ke kolizi jmen proměnných / funkcí (funkce min, max, pocet jsou často využívané, název se nabízí) - řeší mechanizmus prostorů jmen

Přechod od C k C++
Složitější programy v C přináší odlišný programovací styl a ukazují nedostatky jazyka

- Předávání parametrů je možné pouze hodnotou, pro návrat hodnoty je nutné využít konstrukce s ukazatelem. V některých situacích není použití předání hodnotou použit - nově mechanizmus reference
- Tvorba konstantních hodnot pomocí define není ideální (určení typu, horší realizace pro složené datové typy) - const je již i v C
- Tvorba maker pomocí define není ideální (chová se různě pro různé datové typy či jejich kombinace - není typová kontrola) - inline je již i v C

Přechod od C k C++
Složitější programy v C přináší odlišný programovací styl a ukazují nedostatky jazyka

- Přechod ke složeným proměnným
- Ve stuktuře jsou ukazatele na funkce, které manipulují s jejími daty
- Inicializace proměnných
- Vracení zdrojů (paměť, soubory) - konec života proměnné
- Nechtěná/nesprávný změna proměnných
- Využití operátorů pro složené datové typy
- Řešení chyb
- Znovupoužitelnost kódu
- Tvorba rozhraní

Přechod od C k C++
Přechod ke složeným proměnným - struktury

- Problém bývá často popsán souborem proměnných
- Nutnost předávat do funkcí všechny tyto proměnné (složité)
- Snaha o zjednodušení předávání parametrů - využití struktur
- Struktura obsahuje logicky související data

- Složený datový typ je základem objektového programování (C++)
- Vede ke zjednodušení kódu

Přechod od C k C++
Přechod ke složeným proměnným - struktury

- Napište funkci pro sčítání dvou (libovolných) 2D polí pomocí jednoduchých proměnných. Výsledek předejte pomocí nové proměnné. Ukažte volání funkce z funkce main, včetně uložení výsledku do proměnné. Předpokládejte dynamické pole, algoritmy nepište.
- Napište tutéž hlavičku pomocí složené proměnné. Napište definici složené proměnné a ukažte volání z funkce main.
- Srovnejte složitost obou způsobů realizace.
- Rozdělte do zdrojových a hlavičkových souborů.

Přechod od C k C++
Přechod ke složeným proměnným - struktury

- Funkce pro sčítání pomocí jednoduchých proměnných musí mít tři trojice parametrů. Rozměry a ukazatel na data. Dvě proměnné stačí předat "jednoduše", třetí trojice musí být předána tak, aby pomocí ní šel vrátit výsledek (realizace pomocí ukazatelů)

- Struktura bude obsahovat rozměry a ukazatel na data. Hlavička bude mít tři nebo dva parametry. V případě dvou parametrů bude struktura návratovou hodnotou. V případě tří parametrů musí být struktura předána tak, aby v ní hodnoty zůstaly i po opuštění funkce.

- Srovnejte složitost obou způsobů realizace.

- Rozdělte do zdrojových a hlavičkových souborů.

Přechod od C k C++
Ukazatele na funkce

- V programu se může vyskytnout situace, kdy je nutné za běhu programu vyhledat pro složený datový typ nejvhodnější funkci pro manipulaci.
- Funkce se může lišit například na datovém typu nebo na charakteru uložených dat.
- Než hledat vhodnou funkci, je vhodné aby byla přítomna prímo ve stuktuře
- Využití ukazatelů na funkce

- Napište funkci na tisk rozměrů 2D pole a vložte ji do struktury pole.
- Zavolejte tuto funkci aby vytiskla rozměry (aby měla co tisknout, musí mít strukturu jako parametr)

Přechod od C k C++
přetěžování funkcí

- V C může být pouze jedna funkce daného jména
- V C++ může být funkcí se stejným jménem více
- Funkce se musí lišit typem nebo počtem parametrů
- Překladač musí jenoznačně určit, kterou funkci zavolat

- Napište tři funkce na inicializaci struktury 2D pole (všechny pojmenujte Init):
- Jako "prázdnou"
- Pomocí rozměrů x,y - hodnoty vynuluje
- Na základě jiné proměnné typu pole (jako kopii) - originál předejte pomocí ukazatele

Přechod od C k C++
Reference

- Slouží k získání druhého jména pro existující proměnnou (alias)
- Používá se k předávání proměnných do a z funkcí

- Přeprogramujte funkci init pro vytvoření kopie z ukazatele na referenci - srovnejte

Přechod od C k C++

const

- Slouží k ochraně proměnné před přepsáním
- kontroluje překladač
- Typ a const Typ jsou rozlišitelné typy

- V realizovaných kódech doplňte k předávaným parametrům const tam kde je to možné

Přechod od C k C++
Prostory jmen

- Slouží k oddělení programových celků
- V oddělených celcích je možné používat stejné názvy proměnnýc
- Proměnné či celé prostory je možné zpřístupnit v jiných částech programu

- strukturu a "její" funkce přesuňte do jmenného prostoru

Přechod od C k C++
Práce s (dynamickou) pamětí

- Pro manipulaci s objekty jsou operátory new a delete (pro pole new[] a delete[])
- new slouží pro získání paměti (a volá inicializační funkci - konstuktor)
- delete slouží pro uvolnění paměti (předtím volá "uklízecí" funkci - destruktor)
- původní malloc a free stále existují (zpětná kompatibilita s C) ale nejsou vhodné pro objekty
- opetátor new pracuje přímo s názvem datového typu

- naprogramujte alokaci (a odalokaci) 2D pole pomocí new a delete

Přechod od C k C++
Řešení chybových stavů

- Řešení chybových stavů je odděleno od toku programu
- Chyby se neřeší v místě vzniku, ale v místě, kde je to možné
- Při chybě je vyvolána výjimka, které "přenese" informaci o chybě do místa, kde je možné ji řešit

- Ve funkci alokace použijte pro řešení chyb mechanizmus výjimek
- Při záporném rozměru pole použijte výjimku int
- Při nulovém rozměru pole použijte výjimku typu řetězec
- nedostatek paměti ošetřete standardní výjimkou

Přechod od C k C++
vstupy a výstupy - streamy

- Vstupy a výstypy jsou stále v knihovnách
- Díky přetížení operátorů ¡¡ a ¿¿ je správný operátor volán pro daný typ překladačem
- Je možné psát pro vlastní objekty

- Napište funkci, která vytiskne strukturu (čtvercové pole) pomocí operátorů streamů

Přechod od C k C++
inline

- Obdoba maker
- je vhodné pro krátké funkce
-

Přechod od C k C++
Nekontrolovaný přístup k proměnné

- Přičtěte / odečtěte od y-ového rozměru dvojku mezi inicializací a ukončením života - co se stane? Jaký je problém?
- Tato manipulace není v C kontrolovatelná.
- V C++ je možné proměnné "skrýt" před uživatelem, který má možnost s nimi manipulovat pouze pomocí funkcí, které při změně hodnot provedou příslušné kontroly, aby se objekt nedostal do "špatného" stavu

Přechod od C k C++
Inicializace proměnných

- V C neprobíhá automaticky
- Proveďte inicializaci proměnných (do "nedefinovaného" či "nulového" stavu) pro minulý příklad
- Napište funkci, která provede inicializaci proměnné (již pouze pro struct)
- Jak se inicializační funkce zachová při opakovaném volání
- Správnost inicializace záleží v C na přístupu programátora
- V C++ je možné (pro složené datové typy) napsat speciální funkci, kterou při vzniku automaticky zavolá překladač - nazývá se konstruktor

Přechod od C k C++
Ukončení života proměnných

- Proveďte odalokaci proměnné 2D pole
- jak se zachová na neinicializovanou proměnnou
- jak se zachová při opakovaném volání
- správné použití záleží v C na přístupu programátora
- V C++ je možné (pro složené datové typy) napsat speciální funkci, kterou při zániku automaticky zavolá překladač - nazývá se destruktor

Přechod od C k C++
Operátory

- Pro strukturu není možné použít operátory (například pro typ complex, zlomek, 3D vektor/bod ....)
- V C++ je možné pro složené datové typy operátory (do)definovat

Přechod od C k C++
Znovupoužitelnost totožného kódu pro různé typy - šablony

- napište 2D pole pro typ float, double, long double
- v C je nutné napsat (zkopírovat a upravit typ) každý zvlášť
- v případě chyby je nutné opravit pro všechny typy
- V C++ je možné použít tzv. šablony - tvorba stejného kódu pro různé typy

Přechod od C k C++
Znovupoužitelnost stejného základního kódu - dědění

- je-li v C++ základní kód stejný a je nutné ho mírně modifikovat pro různé typy úloh, lze použít mechanizmus dědění, kdy se vytvoří společný základ a ten je základem (base) pro další typy, které jsou mírně upraveny či rozšířeny

Přechod od C k C++
Tvorba rozhraní - polymorfizmus

- polymorfizmus
- máme-li různé typy (struktury), které mají různá data ale se kterými se provádí stejné operace (ale specifický kód pro daná data) je možné nadefinovat společné rozhraní a následně je možné pracovat s různými typy "společně".
- Správnou funkci dohledá za chodu programu sám program podle aktuálního typu proměnné.