به نام خدا

موضوع تحقیق:

تفاوت bool–float–string–int و... در حافظه

انواع حافظه در رم (Stack)و(Heap)

کدام متغیر ها reference type هستند و کدام یک value type

تهیه کننده:

محمد نسیمی فر

رشته تحصیلی:

مهندسی فناوری اطلاعات

درس:

برنامه نویسی سمت سرویس گیرنده

استاد:

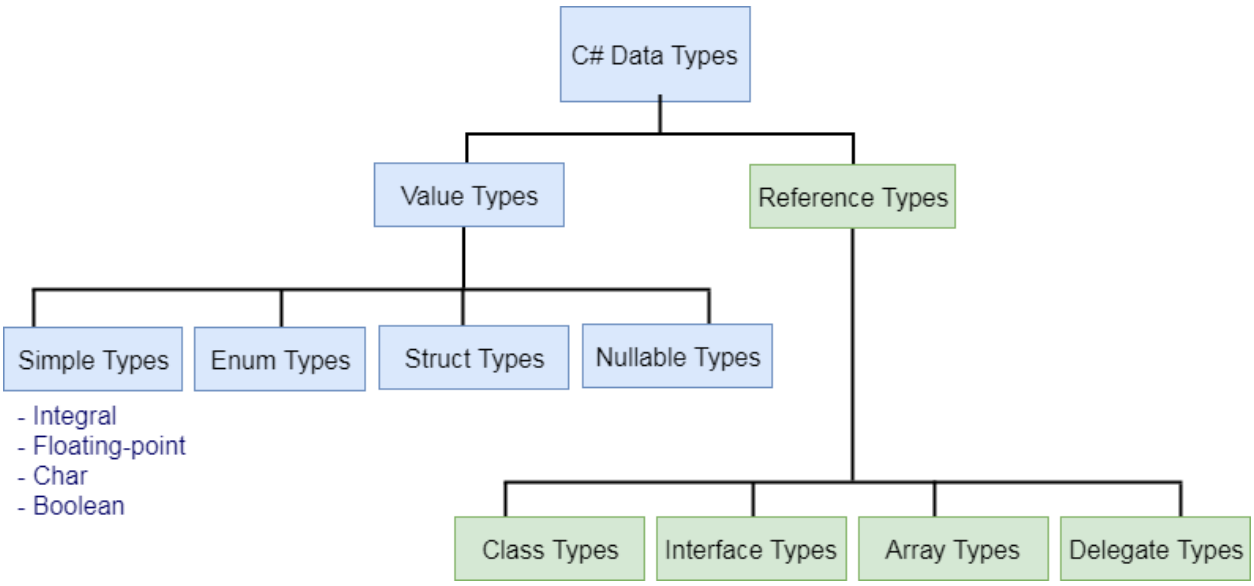
جناب آقای میثاق یاریان

تحقیق شماره ۲

تفاوت int–string–float–bool و... در حافظه

در زبان c# می توان متغیر را بدون مقدار تعریف کرد و بعدا از آن استفاده کرده و آن را مقدار دهی گردد. همچنین می‌توان همزمان با تعیین Data type، مقدار متغیر را نیز مشخص کرد.

به طور کلی نوع داده در سی شارپ به دو دسته تقسیم می شود. این دو دسته عبارتند از داده های مشخص و داده های ارجاعی. داده های مشخص عبارتند از داده های ساده ای مثل int، Float، bool و char، داده های enum، struct، داده های NULL. داده های ارجاعی نیز شامل داده های Class، Interface، Delegate و array می‌باشد.



Byte

این نوع داده اعداد صحیح بین ۰ تا ۲۵۵ را در خود ذخیره کرده و ۸ بیت فضا در حافظه را اشغال می‌کند.

Sbyte

این نوع داده می‌تواند اعداد بین -۱۲۸ تا +۱۲۸ را در ۸ بیت حافظه ذخیره کند.

Short

این نوع داده اعداد صحیح بین -۳۲۷٦۸ تا +۳۲۷٦۸ را در خود ذخیره کرده و ١٦ بیت فضا در حافظه را اشغال می‌کند.

Int

این نوع داده اعداد صحیح بین -۲۱٤۷٤۸۳٦٤۸ تا +۲۱٤۷٤۸۳٦٤۸ را در خود ذخیره کرده و ۳۲ بیت فضا در حافظه را اشغال می‌کند.

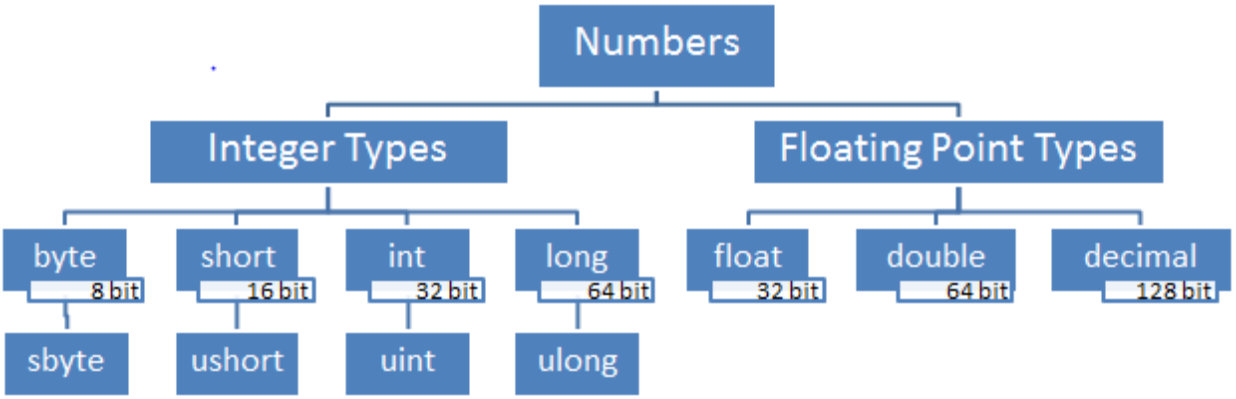
Float

این نوع داده اعداد اعشاری بین – ۳۸e۳٫٤۰۲۸۲۳ تا +۳۸e۳٫٤۰۲۸۲۳ را در خود ذخیره کرده و ۳۲ بیت فضا در حافظه را اشغال می کند. در هنگام کد نویسی نیز باید از پسوند f استفاده شود.

Double

این نوع داده اعداد اعشاری بین -۳۰۸e۱٫۷۹۷٦۹۳۱۳٤۸٦۲۳۲ تا +۳۰۸e۱٫۷۹۷٦۹۳۱۳٤۸٦۲۳۲ را در خود ذخیره کرده و ٦٤ بیت فضا در حافظه را اشغال می کند. در هنگام کد نویسی نیز باید از پسوند d استفاده شود

تفاوت بین Float و Double در این است که Float تا ۷ رقم اعشاری را می تواند در خود ذخیره کند اما Double می تواند ١٤ الی ١٥ رقم اعشاری را در خود ذخیره کند.



انواع حافظه در رم (Stack)و(Heap)

حافظه stack

در بخش user-space حافظه قرار دارد و به صورت خودکار توسط CPU مدیریت می شود. متغیرهای غیر استاتیک، پارامتر های ارسالی به توابع و آدرس های مربوط به return توابع در این حافظه ذخیره می شوند. اندازه حافظه stack ثابت است به همین دلیل به آن static memory گفته می شود.

در این حافظه اطلاعات پشت سر هم و به ترتیب قرار می گیرند به این صورت که آخرین داده ذخیره شده در بالای stack قرار می گیرد و به اصطلاح push می شود، حال اگر قصد برداشتن اطلاعات یا به اصطلاح pop کردن اطلاعات را داشته باشیم آخرین اطلاعات وارد شده در stack را در اختیار داریم. به این الگوریتم LIFO(Last In First Out) می گویند. مثال پر کاربرد در توضیح stack خشاب اسلحه (آخرین گلوله ای که در خشاب قرار داده می شود اولین گلوله ای است که شلیک می شود) و یا بشقاب های روی هم چیده شده (آخرین بشقابی که روی سایر بشقاب ها قرار داده می شود اولین بشقابی است که برداشته می شود) است.

از آنجا که در حافظه stack نیازی به پیدا کردن فضای خالی در حافظه نیست و محل قرارگیری اطلاعات مشخص است (بالای حافظه) بنابراین این حافظه سریع تر از حافظه heap است.

پارامتر ها و اطلاعات مربوط به توابع برای اجرا و کنترل آن ها در این حافظه ذخیره می شوند. تابعی که در بالای stack قرار دارد تابعی است که در حال اجراست و بعد از اتمام کار تابع یا بروز خطا در

اجرای تابع، حافظه اختصاص داده شده به تابع از stack حذف می شود و حافظه اشغال شده آزاد می شود. زمانی که یک thread تعریف می شود در stack قرار می گیرد.

خطایی که ممکن است در اثر استفاده نادرست از حافظه stack رخ دهد stack overflow است. از جمله دلایل stack overflow یا سر ریز می توان به استفاده از متغیرهای محلی حجیم که منجر به کاهش فضای آزاد در stack و تخریب یا corrupt شدن بخشی از memory اشاره کرد.

حافظه Heap

حافظه Heap در قست user-space حافظه مجازی قرار دارد و به صورت دستی توسط برنامه نویس مدیریت می شود. Heap مربوط به زمان اجرا (runtime) است و فضای اشغال شده در heap با اتمام کار تابع آزاد نمی شوند و تا زمانی که Garbage Collector این فضا را آزاد کند یا توسط برنامه نویس داده ها از حافظه heap پاک نشوند در این فضا باقی می ماند. اندازه حافظه heap متغیر است به همین دلیل به آن dynamic memory گفته می شود.

در این نوع از حافظه برای ذخیره مقادیر ابتدا محاسبه ای توسط سیستم عامل صورت می گیرد تا اولین فضای حافظه ای که اندازه آن متناسب با اندازه ای که مورد نیاز ماست را پیدا کند، در صورت وجود این میزان از حافظه درخواستی آن را به صورت رزرو شده درمی آورد تا بقیه برنامه ها به این فضا دسترسی نداشته باشند، سپس آدرس ابتدای این فضای محاسبه شده به صورت یک اشاره گر (pointer) در اختیارمان قرار می دهد (یا به اصلاح allocating).

متغیر ها به صورت پیش فرض در این حافظه قرار نمی گیرند و اگر قصد ذخیره متغیر ها در این حافظه را داشته باشیم باید به صورت دستی این اقدام انجام شود. متغیر هایی که در heap ذخیره می شوند به طور خودکار حذف نمی شوند و باید توسط برنامه نویس و به صورت دستی حذف شوند. به طور کلی مدیریت حافظه heap به صورت دستی توسط برنامه نویس انجام می شود. آرایه های داینامیک در heap ذخیره می شوند.

در صورتی که داده های ما از تعدا block های پشت سر هم در حافظه بیشتر باشد یا در صورت تغییر حجم داده ها در زمان های مختلف (تغییر سایز داده ها امکان پذیر است)، سیستم عامل داده ها را به صورت تکه تکه در block های حافظه ذخیره خواهد کرد.

به دلیل محاسبات برای یافتن آدرس شروع حافظه و در اختیار گرفتن pointer حافظه heap نسبت به stack کندتر است. همچنین اگر داده ها به صورت پشت سر هم در block های حافظه قرار نگرفته باشند (این احتمال بسیار زیاد است) موجب کندی در بازیابی اصلاعات خواهد شد.

وقتی که نمونه‌ای از یک کلاس ایجاد می‌کنیم این مقدار در Heap ذخیره می‌شود و وقتی که کار آن به پایان می رسد garbage collector حافظه را آزاد می کند و اگر موفق به این کار نشود، برنامه نویس باید به صورت دستی حافظه heap را آزاد کند، در غیر این صورت Memory leak اتفاق می افتد که به معنی in use نگه داشتن فضای حافظه برای اشیایی است که دیگر از آن ها در برنامه استفاده نمی شود و garbage collector قادر به آزاد سازی فضایی که آن ها اشغال کرده اند نیست.

به طور کلی Value Type ها (primitive type) فضای زیادی اشغال نمی کنند و در stack ذخیره می شوند. برای دسترسی به متغیر های Value Type، مقدار آن به صورت مستقیم از حافظه stack خوانده می شود، مثلا زمانی که متغیری تعریف می کنیم آن متغیر به همراه مقدار آن در stack قرار می گیرد.

برای دسترسی به متغیرهای Reference Type، ابتدا با مراجعه Stack و دریافت آدرس متغیر در Heap به شئ مربوط به متغیر دسترسی خواهیم داشت. Reference Type ها در حافظه heap نگهداری می شوند. زمانی که یک شی از کلاس ایجاد می کنیم ابتدا متغیری که شی به آن assign شده است با مقدار null در حافظه stack قرار می گیرد، سپس شی در heap ذخیره شده و پس از ذخیره سازی در heap آدرس شی در stack جایگزین null می شود.

همچنین reference type ها به dynamic memory و value type ها به static memory نیاز دارند. در صورت نیاز به dynamic memory، باید امکان دسترسی به heap فراهم باشد و اگر نیازمند static memory باشیم، stack محل ذخیره سازی خواهد بود.

کدام متغیر ها reference type هستند و کدام یک value type

Value Type و Reference Type

سه شارپ داده نوع ها را بسته به چگونگی ذخیره مقادیرشان در حافظه به دو دسته تقسیم میکند :

Value Type

Reference Type

Value Type

به داده نوعی Value Type گفته میشود که یک مقدار را در فضای حافظه ی خود ذخیره کند. و این به این معناست که متغیر هایی که از نوع این داده نوع تعریف میشوند به طور مستقیم دارای مقداری در خود هستند.

وقتی یک متغیر از نوع Value Type را به عنوان آرگومان برای یک متد ارسال می کنید ، سیستم یک کپی جداگانه از آن متغیر را ایجاد کرده و آن را برای متد ارسال میکند. بنابراین اگر تغییری در مقدار آن متغیر در متد مربوطه ایجاد شود تاثیری بر مقدار اصلی آن ندارد.

Reference Type

برخلاف Value Type ها ، Reference Type ها مقادیرشان را به صورت مستقیم در خود ذخیره نمی کنند. در عوض آنها آدرس مکانی از حافظه را که مقدار در آن قرار گرفته است، در خود ذخیره میکنند. به عبارت دیگر Reference Type ها شامل یک اشاره گر هستند که به مکانی دیگر از حافظه اشاره میکند که داده یا مقدار در آن ذخیره شده است.

داده نوع های زیر همگی Reference Type هستند :

String

تمام آرایه ، حتی اگر مقادیر آنها از نوع value type باشد

Class

Delegates

ارسال با ارجاع

وقتی یک متغیر Reference Type را به عنوان آرگومان از یک متد به متد دیگری میفرستید دیگر کپی ایی از آن ساخته نمیشود. در عوض آدرس آن متغیر به متد مربطه ارسال میشود.

مقدار null

Reference Type ها موقعی که هنوز مقداردهی نشده اند به صورت پیشفرض دارای مقدار null هستند. برای مثال یک متغیر رشته ای (و یا هر متغیر از نوع Reference Type) اگر مقدار دهی نشوند دارای مقدار null هستند و عملا به هیچ جایی اشاره نمی کنند .

یک متغیر از نوع Value Type نمیتواند null باشد به این دلیل که آنها یک مقدار را میپذیرند نه یک آدرس را . با این حال متغیر های Value Type قبل از اینکه استفاده شوند باید مقدار دهی شوند. در صورتی که سعی کنید از یک متغیر محلی از نوع Value Type بدون اینکه آن را مقدار دهی کنید استفاده کنید با خطای زمان کامپایل مواجه میشوید.

- Value Type ها مقادیر را در فضای حافظه ی خود ذخیره میکنند در حالی که Reference Type ها آدرس یک مقدار را در فضای خود ذخیره میکنند.
- داده نوع های اولیه و داده نوع struct از نوع Value Type هستند.اشیای class ها ، string ، array و delegate از نوع Reference Type هستند.
- Value Type ها به صورت پیفرض با مقدار ارسال میشوند در حالی که Reference Type ها با ارجاع ارسال میشوند.
- Reference Type ها و Value Type ها بسته به حوزه و مکان قرار گیری آنها در Stack و Heap ذخیره میشوند.