بنام خدا

آرمان ارجمندیان

شماره دانشجویی : 01221033720001

تحقیق جلسه دوم

انواع داده متغیر

byte:

این نوع داده‌ی عددی فضایی معادل ۸ بیت را اشغال می‌کند و اعداد صحیح مثبت بین ۰ تا ۲۵۵± را در خود جای می‌دهد.

sbyte:

این نوع داده‌ی عددی فضایی معادل ۸ بیت را اشغال می‌کند و اعداد صحیح مثبت و منفی بین ۱۲۸- تا ۱۲۷ را در خود جای می‌دهد.

short:این نوع داده عددی فضایی معادل ۱۶ بیت را اشغال کرده و اعداد صحیح مثبت و منفی در بازه‌ی بین ۳۲۷۶۸- تا ۳۲۷۶۷ را در خود ذخیره می‌کند.

ushort:

این نوع داده عددی فضایی معادل ۱۶ بیت را اشغال کرده و اعداد صحیح مثبت در بازه‌ی بین ۰ تا ۶۵۵۳۵± را در خود ذخیره می‌کند.

int:

این نوع داده عددی فضایی معادل ۳۲ بیت را اشغال کرده و اعداد صحیح مثبت و منفی در بازه‌ی بین ۲۱۴۷۴۸۳۶۴۸- تا ۲۱۴۷۴۸۳۶۴۸+ (حدود مثبت منفی ۲۰۰۰۰۰۰۰۰ دو میلیارد) را در خود ذخیره می‌کند.

uint:

این نوع داده عددی فضایی معادل ۳۲ بیت را اشغال کرده و اعداد صحیح مثبت در بازه‌ی بین ۰ تا ۴۲۹۴۹۶۷۲۹۵± (حدود ۴۰۰۰۰۰۰۰۰۰ چهار میلیارد) را در خود ذخیره می‌کند.

long:این نوع داده عددی فضایی معادل ۶۴ بیت را اشغال کرده و اعداد صحیح مثبت و منفی در بازه‌ی بین ۹۲۲۳۳۷۲۰۳۶۸۵۴۷۷۵۸۰۸- تا ۹۲۲۳۳۷۲۰۳۶۸۵۴۷۷۵۸۰۷+ را در خود ذخیره می‌کند.

ulong:

این نوع داده عددی فضایی معادل ۶۴ بیت را اشغال کرده و اعداد صحیح مثبت در بازه‌ی بین ۰ تا ۱۸۴۴۶۷۴۴۰۷۳۷۰۹۵۵۱۶۱۵ را در خود ذخیره می‌کند.

float:

این نوع داده‌ی عددی فضایی معادل ۳۲ بیت را اشغال کرده و اعداد اعشاری مثبت و منفی در بازه‌ی بین ۳۸e۳.۴۰۲۸۲۳ -تا ۳۸e۳.۴۰۲۸۲۳ -را درون خود ذخیره می‌کند.

double:

این نوع داده‌ی عددی فضایی معادل ۶۴ بیت را اشغال کرده و اعداد اعشاری مثبت و منفی در بازه‌ی بین ۳۰۸e۷۹۷۶۹۳۱۳۴۸۶۲۳۲.۱ -تا ۳۰۸e۱.۷۹۷۶۹۳۱۳۴۸۶۲۳۲ +را درون خود ذخیره می‌کند. همچنین با استفاده از این نوع داده مقدار اعشاری به صورت اتوماتیک رند می‌شود.

decimal:

این نوع داده‌ی عددی فضایی معادل ۱۲۸ بیت را اشغال کرده و اعداد اعشاری مثبت و منفی در بازه‌ی بین ۲e-۱۰ × ۱.۰ ± تا ۲۸e۱۰ × ۷.۹ ± را درون خود ذخیره می‌کند.

تفاوت بین نوع داده‌ی float و double وdecimal

توجه :داده‌های عددی float و double معمولا برای اندازه گیری مقادیری که دقت در آنها معیار نیست، مورد استفاده قرار می‌گیرند. مثلا فاصله، مسافت و ... اما داده‌ی عددی decimal برای حالتی که دقت عددی مدنظر می‌باشد بکار گرفته خواهد شد مثل واحد پول، محاسبات حسابداری و...

char:

این نوع داده‌ی رشته‌ای فضایی معادل ۱۶ بیت را اشغال کرده و تمام کاراکترهای یونیکد را درون خود ذخیره می‌کند.

string:

این نوع داده‌ی رشته‌ای مجموعه‌ای از کاراکترها را در خود ذخیره می‌کند و متناسب با آنها فضایی را اشغال خواهد کرد.

bool:

این نوع داده‌ی باینری فضایی معادل ۸ بیت را اشغال کرده و معمولا برای عبارتهای درست و غلط یا ۰ و ۱ مورد استفاده قرار می‌گیرد.

Object

: این نوع داده بر اساس مقداری که برابر آن قرار می‌گیرد نوع عددی، رشته‌ای یا باینری را می‌تواند در خود ذخیره کند. به عبارتی نوع داده object تمام مقادیر و عبارتها را می‌تواند در خود ذخیره کند.

جدول زیر انواع داده رایج در برنامه نویسی C را همراه اندازه آنها نشان می دهد.

نوع داده	اندازه (برحسب بایت)	قالب بندی
int	حداقل ۲، معمولا ۴	%d, %i
char	۱	%c
float	۴	%f
double	۸	%lf
short int	معمولا ۲	%hd
unsigned int	حداقل ۲، معمولا ۴	%u
long int	حداقل ۴، معمولا ۸	%ld, %li
long long int	حداقل ۸	%lld, %lli
unsigned long int	حداقل ۴	%lu
unsigned long long int	حداقل ۸	%llu
signed char	۱	%c
unsigned char	۱	%c
long double	حداقل ۱۰، معمولا ۱۲ تا ۱۶	%Lf

حافظه هیپ(Heap)

ناحیهٔ هیپ (Heap) به طور رایج در ابتدای بخش‌های `.bss` و `.data` قرار گرفته است و به اندازه‌های آدرس بزرگتر قابل رشد است. ناحیهٔ هیپ توسط توابع malloc, calloc, realloc و free مدیریت می‌شود که ممکن است توسط سیستم‌های `brk` و `sbrk` جهت تنظیم اندازه مورد استفاده قرار گیرد. ناحیه هیپ توسط تمامی نخ‌ها، کتابخانه‌های مشترک و ماژول‌های بارگذاری شده در یک فرآیند به اشتراک گذاشته می‌شود.

به طور کلی حافظه Heap بخشی از حافظه کامپیوتر شما است که به صورت خودکار برای شما مدیریت نمی‌شود، و به صورت محکم و مطمئن توسط پردازنده مرکزی مدیریت نمی‌شود. آن بیشتر به عنوان یک ناحیه شناور بسیار بزرگی از حافظه است. برای اختصاص دادن حافظه در ناحیه هیپ شما باید از توابع `malloc()`, `calloc()` که توابعی از C هستند استفاده کنید. یکبار که شما حافظه ای را در ناحیه هیپ اختصاص دهید، جهت آزاد سازی آن باید خود مسئول باشید و با استفاده از تابع free() این کار را به صورت دستی جهت آزاد سازی حافظه اختصاص یافته شده انجام دهید. اگر شما در این کار موفق نباشید، برنامه شما در وضعیت نَشت حافظه (Memory Leak) قرار خواهد گرفت. این بدین معنی است که حافظه اختصاص یافته شده در هیپ هنوز خارج از دسترس قرار گرفته و مورد استفاده قرار نخواهد گرفت. این وضعیت همانند گرفتگی رَگ در بدن انسان است و حافظه نشت شده جهت عملیات در دسترس نخواهد بود. خوشبختانه ابزارهایی برای کمک کردن به شما در این زمینه موجود هستند که یکی از آن‌ها Valgrindنام دارد و شما می‌توانید در زمان اشکال زدائی از آن جهت تشخیص نواحی نشت دهنده حافظه استفاده کنید.

بر خلاف حافظه اِستک (Stack) حافظه هیپ محدودیتی در اندازه متغیرها ندارد (جدا از محدودیت آشکار فیزیکی در کامپیوتر شما). حافظه هیپ در خواندن کمی کُند تر از نوشتن نسبت به حافظه اِستک است، زیرا جهت دسترسی به آن‌ها در حافظه هیپ باید از اشاره گر استفاده شود. بر خلاف حافظه اِستک، متغیرهایی که در حافظه هیپ ساخته می‌شوند توسط هر تابعی در هر بخشی از برنامه شما در دسترس بوده و اساسا متغیرهای تعریف شده در هیپ در دامنه سراسری قرار دارند.

حافظه اِستک(Stack)

ناحیهٔ اِستک (Stack) شامل برنامه اِستک، با ساختار **LIFO**کوتاه شده عبارت **Last In First Out**(آخرین ورودی از همه زودتر خارج می‌شود) به طور رایج در بالاترین بخش از حافظه قرار می‌گیرد. یک (اشاره گر پشته) در بالاترین قسمت اِستک قرار می‌گیرد. زمانی که تابعی فراخوانی می‌شود این تابع به همراه تمامی متغیرهای محلی خودش در داخل حافظه اِستک قرار می‌گیرد و با فراخوانی یک تابع جدید تابع جاری بر روی تابع قبلی قرار می‌گیرد و کار به همین صورت درباره دیگر توابع ادامه پیدا می‌کند.

مزیت استفاده از حافظه اِستک در ذخیره متغیرها است، چرا که حافظه به صورت خودکار برای شما مدیریت می‌شود. شما نیازی برای اختصاص دادن حافظه به صورت دستی ندارید، یا نیازی به آزاد سازی حافظه ندارید. به طور کلی دلیل آن نیز این است که حافظه اِستک به اندازه کافی توسط پردازنده مرکزی بهینه و سازماندهی می‌شود. بنابراین خواندن و نوشتن در حافظه اِستک بسیار سریع است.

کلید درک حافظه اِستک در این است که زمانی که تابع خارج می‌شود، تمامی متغیرهای موجود در آن همراه با آن خارج و به پایان زندگی خود میرسند. بنابراین متغیرهای موجود در حافظه اِستک به طور طبیعی به صورت محلی هستند. این مرتبط با مفهوم دامنه متغیرها است که قبلا از آن یاد شده است، یا همان متغیرهای محلی در مقابل متغیرهای سراسری .

یک اشکال رایج در برنامه نویسی C تلاش برای دسترسی به یک متغیر که در حافظه اِستک برای یک تابع درونی ساخته شده است می‌باشد. یعنی از یک مکان در برنامه شما به خارج از تابع (یعنی زمانی که آن تابع خارج شده باشد) رجوع می‌کند.

یکی دیگر از ویژگی‌های حافظه اِستک که بهتر است به یاد داشته باشید این است که، محدودیت اندازه (نسبت به نوع سیستم عامل متفاوت) است. این مورد در حافظه هیپ صدق نمی‌کند.

خلاصه ای از حافظه اِستک(Stack)

- حافظه اِستک متناسب با ورود و خروج توابع و متغیرهای درونی آن‌ها افزایش و کاهش می‌یابد
- نیازی برای مدیریت دستی حافظه برای شما وجود ندارد، حافظه به طور خودکار برای متغیرها اختصاص و در زمان نیاز به صورت خودکر آزاد می‌شود
- در اِستک اندازه محدود است
- متغیرهای اِستک تنها در زمان اجرای تابع ساخته می‌شوند
- مزایا و معایب حافظه اِستک و هیپ

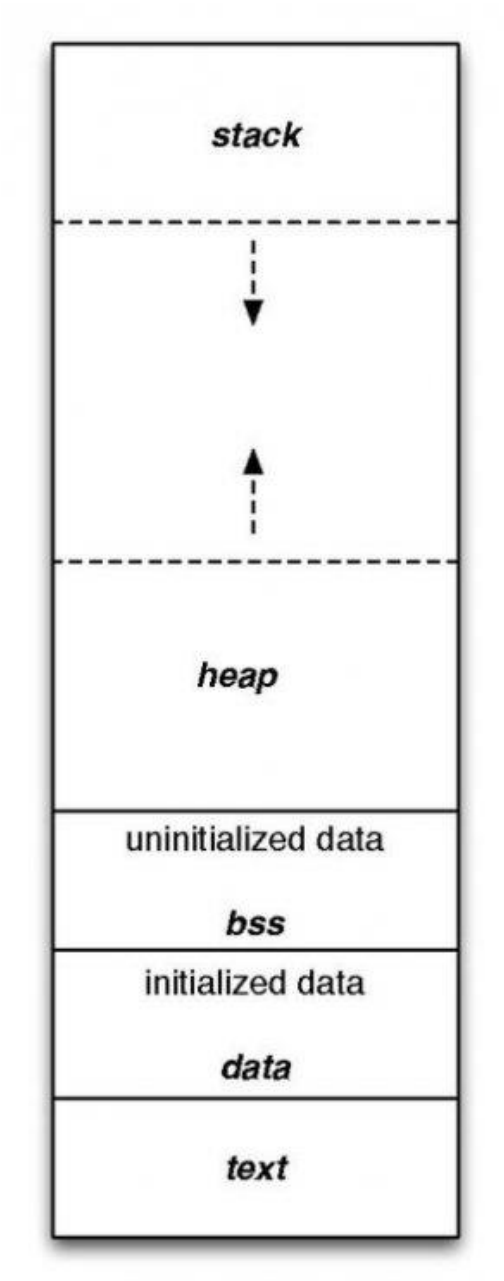
حافظه اِستک(Stack)

- دسترسی بسیار سریع به متغیرها
- نیازی برای باز پس گیری حافظه اختصاص یافته شده ندارید
- فضا در زمان مورد نیاز به اندازه کافی توسط پردازنده مرکزی مدیریت می‌شود، حافظه ای نشت نخواهد کرد
- متغیرها فقط محلی هستند
- محدودیت در حافظه اِستک بسته به نوع سیستم عامل متفاوت است
- متغیرها نمی‌توانند تغییر اندازه دهند

حافظه هیپ(Heap)

- متغیرها به صورت سراسری قابل دسترس هستند
- محدودیتی در اندازه حافظه وجود ندارد

- تضمینی برای حافظه مصرفی وجود ندارد، ممکن است حافظه در زمان‌های خاص از برنامه نشت کرده و حافظه اختصاص یافته شده برای استفاده در عملیات دیگر آزاد نخواهد شد
- شما باید حافظه را مدیریت کنید، شما باید مسئولیت آزاد سازی حافظه های اختصاص یافته شده به متغیرها را بر عهده بگیرید
- اندازه متغیرها می‌تواند توسط تابع `realloc()` تغییر یابد



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.

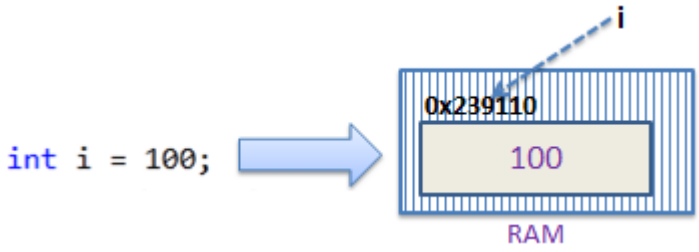
نکته :تمام Value Type ها از فضای نام `System.ValueType` مشتق میشوند که آن فضای نامی هم در فضای نام `System.Object` قرار دارد.

برای مثال متغیری از نوع `int` را در نظر بگیرید :

```
int   i =   100  ;
```

سیستم مقدار عدد صحیح 100 را در فضای حافظه ای که برای متغیر `"i"` تخصیص داده شده است ، ذخیره می کند.

تصویر زیر نحوه ی ذخیره سازی مقدار 100 را در حافظه به آدرس `(0x239110)` برای متغیر `"i"` نشان میدهد:



همه ی داده نوع هایی که در زیر آورده شده است از نوع value type هستند:

bool	byte	char	decimal	double
enum	float	int	long	sbyte
short	sbyte	unit	ulong	ushort

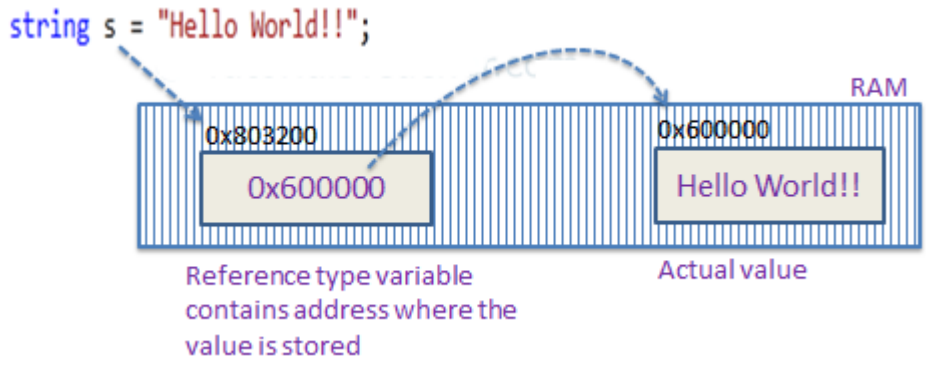
Reference Type

برخلاف Value Type ها ، Reference Typeها 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.

برای این حالت یک متغیر رشته ای را میتوان مثال برد:

```
string s = "Hello Worlds";
```

تصویر زیر چگونگی تخصیص حافظه را برای متغیر رشته ای بالا نشان میدهد:



داده نوع های زیر همگی Reference Type هستند:

- String
- تمام آرایه ، حتی اگر مقادیر آنها از نوع value type باشد
- Class
- Delegates