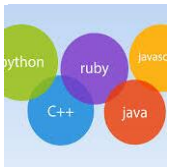
c/c new與malloc的區別及使用時注意的問題

2018.07.27 程式語言

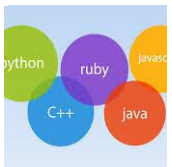
－＞和c＋＋, Advanced C Language, c, C Language, c＋＋: : , c＋＋氣泡排序, c＋＋漢諾塔, c＋＋純虛擬函式, malloc, new, 記憶體

近期文章



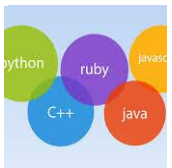
Spark入門（一）用SparkShell初嘗Spark滋味

2019.12.08



Spark入門（二）如何用Idea運行我們的Spark項目

2019.12.08



Spark入門（三）Spark經典的單詞統計

2019.12.08

目錄 關閉

Malloc：

定義上:malloc memory allocation 動態記憶體分配 是c中的一個函式

使用方法:

```
extern void *malloc(unsigned int num_bytes)
```

num_bytes記憶體塊位元組長度。

記憶體塊大小確定:malloc是通過我們計算然後得到一塊新記憶體,然後指定資料型別並且記憶體值也是隨機的。

使用時:需要引入標頭檔案庫函式 stdlib.h 或是 malloc.h(malloc.h與alloc.h一致)。

Spark入門（四）Spark的map、flatMap、mapToPair

2019.12.08

Spark入門（五）Spark的reduce和reduceByKey

2019.12.08

Spark入門（六）Spark的combineByKey、sortBykey

2019.12.08

Spark入門（七）Spark的intersection、subtract、union和distinct

2019.12.08

Spark實戰尋找5億次訪問中，訪問次數最多的人

2019.12.08

Spark實戰搭建我們的Spark分佈式架構

2019.12.08

【機器學習】深度學習開發環境搭建

2019.12.08

記憶體分配位置:堆中動態分配的記憶體。

具體分配過程:由程式向作業系統申請,作業系統遍歷空閒結點連結串列,將第一個大於申請空間的堆結點分配給程式,然後將空閒結點連結串列中此節點刪掉。

成功分配:返回值為指向被分配記憶體的指標。

失敗分配:返回值為空NULL。

返回型別:void* (未確定型別的指標)。

void*型別可以通過強制型別轉換轉換為任意其他型別(因為使用者儲存資料型別未知,由使用者決定資料型別)。

記憶體塊釋放:free()函式 將記憶體還給程式或作業系統。

注意:malloc與free都屬於c/c 標準庫函式,在使用時應該配對 申請之後不釋放就會有可能發生記憶體洩漏。

使用free時需要檢查指標是否為空。

New:

new 是c 中的運算子(其地位等同於“ ”,“=”)。

在使用:new時不止分配記憶體,還會進行初始化,執行相應建構函式,初始化時需要指定資料型別。

記憶體分配位置:自由儲存區為物件分配記憶體。

使用時:無需引入標頭檔案,new是保留字。

new 和 delete 是配對使用的。

在使用delete時注意將指標置為0,否則會形成懸垂指標(指標所指記憶體已被釋放,仍指向該記憶體),造成錯誤。

new 可以看做是malloc 加 建構函式的執行,就是new更高階一些。

new的幾種用法:

```
int *p=new int; //在自由儲存區開闢一個int變數
int *p=new int[10];//在自由儲存區開闢一個int陣列，有10個元素
int *p=new int(10);//在自由儲存區開闢一個int變數,並初始化為10
```

釋放記憶體時無需檢查是否為空。

如果p等於NULL, 則delete p不作任何事情。由於之後可以得到測試, 並且大多數的測試方法論都強制顯式測試每個分支點, 因此你不應該加上多餘的 if 測試。

錯誤的:

```
 if (p != NULL)
delete p;
正確的: delete p;
```

面試時可能會問到的問題:
有了malloc/free為什麼還要new/delete?

1)它們都可用於申請動態記憶體和釋放記憶體。

2)malloc是庫函式只能作用於內部資料型別,對於非內部資料動態物件而言,就不能完成物件的初始化與銷燬,即執行建構函式與解構函式,而new 與 delete此類運算子就能夠在編譯器的控制許可權內完成,物件的初始化與銷燬任務,即執行建構函式與解構函式。

為什麼C 不把malloc/free淘汰出局呢

既然new/delete的功能完全覆蓋了malloc/free,為什麼C 不把malloc/free淘汰出局呢?這是因為C 程式經常要呼叫C函式,而C程式只能用malloc/free管理動態記憶體。

我們不要企圖用malloc/free來完成動態物件的記憶體管理,應該用new/delete。由於內部資料型別的“物件”沒有構造與析構的過程,對它們而言malloc/free和new/delete是等價的。

注意:如果用free釋放“new建立的動態物件”,那麼該物件因無法執行解構函式而可能導致程式出錯。

如果用delete釋放“malloc申請的動態記憶體”,結果也會導致程式出錯,但是該程式的可讀性很差。所以new/delete必須配對使用,malloc/free也一樣。

區分零值指標和NULL指標

零值指標,是值是0的指標,可以是任何一種指標型別,可以是通用變體型別void*也可以是char*,int*等等。

空指標,其實空指標只是一種程式設計概念,就如一個容器可能有空和非空兩種基本狀態,而在非空時可能裡面儲存了一個數值是0,因此空指標是人為認為的指標不提供任何地址訊息。

malloc 與 new 對於記憶體洩漏問題

記憶體洩漏對於malloc或者new都可以檢查出來的,區別在於new可以指明是那個檔案的那一行,而malloc沒有這些資訊。

ps:全文由博主爬貼訪博總結的, 一切源於網路, 希望大家多多提意見。

写评论

Back to Top