C 可变参数 C 命令行参数

C 内存管理

本章将讲解 C 中的动态内存管理。C 语言为内存的分配和管理提供了几个函数。这些函数可以在 **<stdlib.h>** 头文件中找到。

序号	函数和描述
1	**void *calloc(int num, int size);** 在内存中动态地分配 num 个长度为 size 的连续空间，并将每一个字节都初始化为 0。所以它的结果是分配了 num*size 个字节长度的内存空间，并且每个字节的值都是0。
2	**void free(void *address);** 该函数释放 address 所指向的内存块,释放的是动态分配的内存空间。
3	**void *malloc(int num);** 在堆区分配一块指定大小的内存空间，用来存放数据。这块内存空间在函数执行完成后不会被初始化，它们的值是未知的。
4	**void *realloc(void *address, int newsize);** 该函数重新分配内存，把内存扩展到 **newsize**。

注意：void * 类型表示未确定类型的指针。C、C++ 规定 void * 类型可以通过类型转换强制转换为任何其它类型的指针。

动态分配内存

编程时，如果您预先知道数组的大小，那么定义数组时就比较容易。例如，一个存储人名的数组，它最多容纳 100 个字符，所以您可以定义数组，如下所示：

```
char name[100];
```

但是，如果您预先不知道需要存储的文本长度，例如您向存储有关一个主题的详细描述。在这里，我们需要定义一个指针，该指针指向未定义所需内存大小的字符，后续再根据需求来分配内存，如下所示：

实例

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main()
{
char name[100];
char *description;
strcpy(name, "Zara Ali");
```

```
/* 动态分配内存 */
description = (char *)malloc( 200 * sizeof(char) );
if( description == NULL )
{
fprintf(stderr, "Error - unable to allocate required memory\n");
}
else
{
strcpy( description, "Zara ali a DPS student in class 10th");
}
printf("Name = %s\n", name );
printf("Description: %s\n", description );
}
```

当上面的代码被编译和执行时，它会产生下列结果：

```
Name = Zara Ali
Description: Zara ali a DPS student in class 10th
```

上面的程序也可以使用 **calloc()** 来编写，只需要把 malloc 替换为 calloc 即可，如下所示：

```
calloc(200, sizeof(char));
```

当动态分配内存时，您有完全控制权，可以传递任何大小的值。而那些预先定义了大小的数组，一旦定义则无法改变大小。

重新调整内存的大小和释放内存

当程序退出时，操作系统会自动释放所有分配给程序的内存，但是，建议您在不需要内存时，都应该调用函数 **free()** 来释放内存。

或者，您可以通过调用函数 **realloc()** 来增加或减少已分配的内存块的大小。让我们使用 realloc() 和 free() 函数，再次查看上面的实例：

实例

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main()
{
char name[100];
char *description;
strcpy(name, "Zara Ali");
/* 动态分配内存 */
description = (char *)malloc( 30 * sizeof(char) );
if( description == NULL )
{
fprintf(stderr, "Error - unable to allocate required memory\n");
}
else
{
```

```
strcpy( description, "Zara ali a DPS student.");
}
/* 假设您想要存储更大的描述信息 */
description = realloc( description, 100 * sizeof(char) );
if( description == NULL )
{
fprintf(stderr, "Error - unable to allocate required memory\n");
}
else
{
strcat( description, "She is in class 10th");
}
printf("Name = %s\n", name );
printf("Description: %s\n", description );
/* 使用 free() 函数释放内存 */
free(description);
}
```

当上面的代码被编译和执行时，它会产生下列结果：

```
Name = Zara Ali
Description: Zara ali a DPS student.She is in class 10th
```

您可以尝试一下不重新分配额外的内存，strcat() 函数会生成一个错误，因为存储 description 时可用的内存不足。

← C 可变参数 C 命令行参数 →

2 篇笔记

写笔记

对于 void 指针，GNU 认为 **`void *`** 和 **`char *`** 一样，所以以下写法是正确的:

```
description = malloc( 200 * sizeof(char) );
```

但按照 ANSI(American National Standards Institute) 标准，需要对 void 指针进行强制转换，如下:

```
description = (char *)malloc( 200 * sizeof(char) );
```

同时，按照 ANSI(American National Standards Institute) 标准，不能对 void 指针进行算法操作:

```
void * pvoid;
pvoid++; //ANSI：错误
pvoid += 1; //ANSI：错误
// ANSI标准之所以这样认定，是因为它坚持：进行算法操作的指针必须是确定知道其指向数据类型大小的。

int *pint;
pint++; //ANSI：正确
```

更多内容参考：C 语言中 void* 详解及应用

长颈鹿 7个月前 (09-01)

对于我们手动分配的内存，在 C 语言中是不用强制转换类型的。

```
description = malloc( 200 * sizeof(char) ); // C 语言正确。
description = malloc( 200 * sizeof(char) ); // C++ 错误
```

但是 C++ 是强制要求的，不然会报错。

Blithe 2个月前 (01-24)