

[文章](#) [问答](#) [论坛](#) [东西](#) [休息室](#) [?](#)

Search for articles, questions,

手表



就地多维 C++ 数组



迈克尔·乔达基斯

2019 年 6 月 16 日 [警察](#)

评价我: 4.94/5 (12 票)

就地创建动态多维数组

介绍

通常，您需要一个 C++ 多维数组。如果它是静态的，这很容易，例如：

C++

复制代码

```
int a[5][10];
```

但是，当长度可变的时候，就做不到了，`int a[n1][n2]`；通常我们使用 `std::vector`，但这需要手动构建：

C++

复制代码

```
int n1 = 5;
int n2 = 6;
std::vector<std::vector<int>> v;
v.resize(n1);
for(auto& vv : v)
    vv.resize(n2);
```

这会浪费时间，因为有多多个内存分配。如果能分配一次就更好了。但是，这不起作用：

C++

复制代码

```
std::vector<char> v(1000);
int** n = (int**)v.data();
n[0][0] = 5; // boom
```

它崩溃了，我们必须使用 `for` 循环进行初始化。

代码

C++ 11 和可变参数模板来帮助我们：

C++

缩小▲ 复制代码

```

template <typename T = char> size_t CreateArrayAtMemory(void*, size_t bs)
{
    return bs*sizeof(T);
}

template <typename T = char, typename ... Args>
size_t CreateArrayAtMemory(void* p, size_t bs, Args ... args)
{
    size_t R = 0;
    size_t PS = sizeof(void*);
    char* P = (char*)p;
    char* P0 = (char*)p;

    size_t BytesForAllPointers = bs*PS;
    R = BytesForAllPointers;

    char* pos = P0 + BytesForAllPointers;
    for (size_t i = 0; i < bs; i++)
    {
        char** pp = (char**)P;
        if (p)
            *pp = pos;
        size_t RLD = CreateArrayAtMemory<T>(p ? pos : nullptr, args ...);
        P += PS;
        R += RLD;
        pos += RLD;
    }
    return R;
}

```

这个递归函数接受许多参数（声明数组的维度）并修改传递给它的内存指针，因此将其转换为多维数组是安全的。如果 `nullptr` 传递，则返回分配所需的字节。

例如，一维数组：

C++

复制代码

```

int j = 0x21;
size_t n1 = CreateArrayAtMemory(nullptr, 2);
vector<char> a1(n1);
char* f1 = (char*)a1.data();
CreateArrayAtMemory(f1, 2);
for (int i1 = 0; i1 < 2 ; i1++)
{
    f1[i1] = j++;
}
for (int i1 = 0; i1 < 2 ; i1++)
{
    std::cout << (int)f1[i1] << " ";
}
std::cout << std::endl << n1 << " bytes used " << std::endl;

```

预期输出：

复制代码

```

33 34
2 bytes used

```

现在是 `short`s 的二维数组 2x3：

C++

复制代码

```

int j = 0x21;
size_t n2 = CreateArrayAtMemory<short>(nullptr, 2, 3);
vector<char> a2(n2);
short** f2 = (short**)a2.data();

```

```

CreateArrayAtMemory(f2,2,3);
for (int i1 = 0; i1 < 2; i1++)
{
    for (int i2 = 0; i2 < 3; i2++)
    {
        f2[i1][i2] = j++;
    }
}
// Dump
for (int i1 = 0; i1 < 2; i1++)
{
    for (int i2 = 0; i2 < 3; i2++)
    {
        std::cout << (int)f2[i1][i2] << " ";
    }
}
std::cout << std::endl << n2 << " bytes used " << std::endl;
std::cout << std::endl;

```

复制代码

```

33 34 35 36 37 38
20 bytes used

```

2x3x4x5x6 的一些更大的东西:

C++

缩小▲ 复制代码

```

int j = 0x21;
size_t n5 = CreateArrayAtMemory<int>(a5, 2,3,4,5,6);
vector<char> a5(n5);
int***** f5 = (int*****)a5.data();
CreateArrayAtMemory(f5,2,3,4,5,6);
for (int i1 = 0; i1 < 2; i1++)
{
    for (int i2 = 0; i2 < 3; i2++)
    {
        for (int i3 = 0; i3 < 4; i3++)
        {
            for (int i4 = 0; i4 < 5; i4++)
            {
                for (int i5 = 0; i5 < 6; i5++)
                {
                    f5[i1][i2][i3][i4][i5] = j++;
                }
            }
        }
    }
}

// Dump
for (int i1 = 0; i1 < 2; i1++)
{
    for (int i2 = 0; i2 < 3; i2++)
    {
        for (int i3 = 0; i3 < 4; i3++)
        {
            for (int i4 = 0; i4 < 5; i4++)
            {
                for (int i5 = 0; i5 < 6; i5++)
                {
                    std::cout << (int)f5[i1][i2][i3][i4][i5] << " ";
                }
            }
        }
    }
}

```

```
std::cout << std::endl << n5 << " bytes used " << std::endl;
std::cout << std::endl;
```

缩小▲ 复制代码

```
33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61
62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90
91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114
115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136
137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158
159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180
181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202
203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224
225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246
247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268
269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290
291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312
313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334
335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356
357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378
379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400
401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422
423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444
445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466
467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488
489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510
511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532
533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554
555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576
577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598
599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620
621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642
643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664
665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686
687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708
709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730
731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752

3488 bytes used
```

玩得开心!

历史

- 2019 年 6 月 16 日 : 首次发布

执照


本文以及任何相关的源代码和文件均根据[The Code Project Open License \(CPOl\)](#)获得许可

分享

关于作者



迈克尔·乔达基斯

软件开发人员
希腊 

手表
该会员

我正在使用 C++、PHP、Java、Windows、iOS、Android 和 Web (HTML/Javascript/CSS)。

我拥有数字信号处理和人工智能博士学位，专攻专业音频和人工智能应用。

我的主页：<https://www.turbo-play.com>

评论和讨论

添加评论或问题



电子邮件提醒

Search Comments



第一 页上一页 下一页

它不是多维 C++ 数组。 

armagedescu 5-May-20 20:23

测试示例中的多个错误。还有一个建议。 

Stefan_Lang 5-Jul-19 20:09

它仅适用于整数。其他原生类型呢？ 

Wym2 19-Jun-19 2:49

回复：它仅适用于整数。其他原生类型呢？ 

Michael Chourdakis 19-Jun-19 4:26

回复：它仅适用于整数。其他原生类型呢？ 

Wym2 20-Jun-19 1:52

回复：它仅适用于整数。其他原生类型呢？ 

Michael Chourdakis 20-Jun-19 2:09

回复：它仅适用于整数。其他原生类型呢？ 

Wym2 20-Jun-19 5:25

回复：它仅适用于整数。其他原生类型呢？ 

Stefan_Lang 5-Jul-19 0:30

回复：它仅适用于整数。其他原生类型呢？ 

Stefan_Lang 5-Jul-19 19:57

回复：它仅适用于整数。其他原生类型呢？ 

Wym2 6-Jul-19 0:55

示例中有错误

Member 14187956 18-Jun-19 10:52

Re: 例子中有错误

Michael Chourdakis 18-Jun-19 16:13

Re: 例子中有错误

Member 14187956 18-Jun-19 20:06

Re: 例子中有错误

Stefan_Lang 5-Jul-19 19:32

Re: 例子中有错误

Stefan_Lang 5-Jul-19 19:51

不明显的方式

Сергій Ярошко 17-Jun-19 15:25

回复：不明显的方式

Michael Chourdakis 17-Jun-19 17:40

回复：不明显的方式

Stefan_Lang 8-Jul-19 22:23

刷新

1

- 一般
- 新闻
- 建议
- 问题
- 错误
- 答案
- 笑话
- 赞美
- 咆哮
- 管理员

使用Ctrl+Left/Right 切换消息， Ctrl+Up/Down 切换主题， Ctrl+Shift+Left/Right 切换页面。