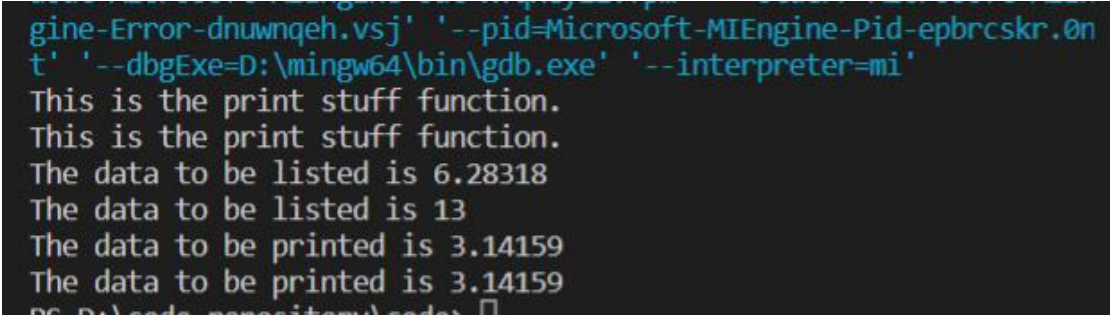
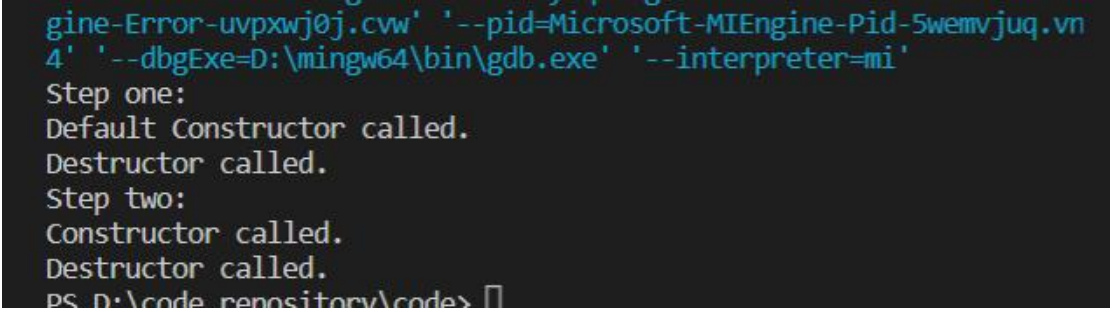


# 计算机学院 高级语言程序设计 课程实验报告

实验题目：对象数组，指针，动态内存		学号：202200130048
日期：2023. 3. 31	班级： 6	姓名： 陈静雯
Email：1205037094@qq. com		
<p>实验步骤与内容：</p> <ol style="list-style-type: none"><li>1. 练习以下课程内容，做好结果截图与分析</li><li>2. 分析实验 7 附件.zip 中的 c6test.cpp</li><li>3. vector 练习</li><li>4. string 练习</li><li>5. 分析对象数组项目 fig07_15to17 以及 fig07_22to24</li></ol>		
<p>结论分析与体会：</p> <p>1. (1)</p>  <p>分析：先定义函数指针，然后 (*functionpointer) (PI) 调用 printstuff 函数，再把指针指向 printmessage 函数，(TWO_PI) (13.0) 调用两次，最后指向 printfloat，(PI) 调用。函数指针指向程序代码存储区，实现函数回调</p> <p>1. (2)</p>  <p>分析：Point *ptr1 = new Point; 构造没有参数，调用第一个构造函数；delete</p>		

即释放指针指向的内存，即析构动态对象 point，输出 Destructor called. 第二步，new Point(1,2) 有参数传递，调用第二个构造函数，输出 Constructor called. 然后 delete

1. (3)

```
dout=Microsoft-MIEngine-Out-1kofhcsr.3vn' '--stderr=Microsoft-MIEngine-Error-iq3x5naf.n4d' '--pid=Microsoft-MIEngine-Pid-c2oeq1pi.z00' '--dbgExe=D:\mingw64\bin\gdb.exe' '--interpreter=mi'
Default Constructor called.
Default Constructor called.
Deleting...
Destructor called.
Destructor called.
PS D:\code_repository\code> |
```

分析：Point \*ptr = new Point[2] 没有参数传递，调用第一个构造函数，动态对象数组，创建两个 point 对象，输出两次 Default Constructor called. ptr.move; 改变 point 里 x y 的值，最后 delete 数组，调用两次析构函数，输出两次 Destructor called.

1. (4)

```
P' '--dbgExe=D:\mingw64\bin\gdb.exe' '--interpreter=mi
0 1 2 3 4 5 6 7
10 11 12 13 14 15 16 17
20 21 22 23 24 25 26 27
30 31 32 33 34 35 36 37
40 41 42 43 44 45 46 47
50 51 52 53 54 55 56 57
60 61 62 63 64 65 66 67
70 71 72 73 74 75 76 77
80 81 82 83 84 85 86 87

100 101 102 103 104 105 106 107
110 111 112 113 114 115 116 117
120 121 122 123 124 125 126 127
130 131 132 133 134 135 136 137
140 141 142 143 144 145 146 147
150 151 152 153 154 155 156 157
160 161 162 163 164 165 166 167
170 171 172 173 174 175 176 177
180 181 182 183 184 185 186 187

200 201 202 203 204 205 206 207
210 211 212 213 214 215 216 217
220 221 222 223 224 225 226 227
230 231 232 233 234 235 236 237
240 241 242 243 244 245 246 247
250 251 252 253 254 255 256 257
260 261 262 263 264 265 266 267
270 271 272 273 274 275 276 277
280 281 282 283 284 285 286 287
```

```

300 301 302 303 304 305 306 307
310 311 312 313 314 315 316 317
320 321 322 323 324 325 326 327
330 331 332 333 334 335 336 337
340 341 342 343 344 345 346 347
350 351 352 353 354 355 356 357
360 361 362 363 364 365 366 367
370 371 372 373 374 375 376 377
380 381 382 383 384 385 386 387

400 401 402 403 404 405 406 407
410 411 412 413 414 415 416 417
420 421 422 423 424 425 426 427
430 431 432 433 434 435 436 437
440 441 442 443 444 445 446 447
450 451 452 453 454 455 456 457
460 461 462 463 464 465 466 467
470 471 472 473 474 475 476 477
480 481 482 483 484 485 486 487

500 501 502 503 504 505 506 507
510 511 512 513 514 515 516 517
520 521 522 523 524 525 526 527
530 531 532 533 534 535 536 537
540 541 542 543 544 545 546 547
550 551 552 553 554 555 556 557
560 561 562 563 564 565 566 567
570 571 572 573 574 575 576 577
580 581 582 583 584 585 586 587

600 601 602 603 604 605 606 607
610 611 612 613 614 615 616 617
620 621 622 623 624 625 626 627
630 631 632 633 634 635 636 637
640 641 642 643 644 645 646 647
650 651 652 653 654 655 656 657
660 661 662 663 664 665 666 667
670 671 672 673 674 675 676 677
680 681 682 683 684 685 686 687

```

分析：int (\*cp)[9][8] = new int[7][9][8]; 相当于创建一个三维数组，每个元素值为数组下标 i (0-6) j (0-8) k (0-7) 分别为百十个位所组成的数，第一个三重循环给数组赋值，第二个三重循环输出，数组个数为除最左边一维外的下标乘积即 72 个，总共有 7 个这样的数组，相当于三维数组

## 2. (1)

解释：Object \*a=new Object，动态创建一个 object 对象，再创建一个指针 a 指向这个对象的地址；Object &b(\*new Object(\*a))，把 a 指向的对象复制构造给一个新的对象，&b 即引用这个新对象，引用就是与被引用的本质是同一个

## 2. (2)



```

=D:\mingw64\bin\gdb.exe' '--interpreter=mi'
in Constructor. this:0x771e80, id:1
in Copy constructor:Object(Object &ob) this:0x771a50, ob:0x771e80, id:2
In exec:after(a,b),count=2
exec:a id:1
exec:b id:2
in Constructor. this:0x61fdc4, id:3
In exec:after c, count=3
Remove:0x771e80, id:1
after delete a, count=2
Remove:0x61fdc4, id:3
PS D:\code repository\code> & 'c:\Users\c''j''w\vscode\extensions\ms-vscode.cpptools-1

```

```

=D:\mingw64\bin\gdb.exe' '--interpreter=mi'
in Constructor. this:0xd61e80, id:1
in Copy constructor:Object(Object &ob) this:0xd61a50, ob:0xd61e80, id:2
In exec:after(a,b),count=2
exec:a id:1
exec:b id:2
in Constructor. this:0x61fdc4, id:3
In exec:after c, count=3
Remove:0xd61e80, id:1
after delete a, count=2
Remove:0x61fdc4, id:3
return main, after exec, count:1
PS D:\code repository\code> & 'c:\Users\c''j''w\vscode\extensions\ms-vscode.cpptools-1.11.5\bin\gdb.exe' '--interpreter=mi'

```

```

=D:\mingw64\bin\gdb.exe' '--interpreter=mi'
in Constructor. this:0xf81e80, id:1
in Copy constructor:Object(Object &ob) this:0xf81a50, ob:0xf81e80, id:2
in Copy constructor:Object(Object &ob) this:0x61fdcc, ob:0xf81a50, id:3
In exec:after(a,b),count=3
exec:a id:1
exec:b id:3
in Constructor. this:0x61fdc8, id:4
In exec:after c, count=4
Remove:0xf81e80, id:1
after delete a, count=3
Remove:0x61fdc8, id:4
Remove:0x61fdcc, id:3
return main, after exec, count:1
PS D:\code repository\code>

```

分析: \*b(new Object(\*a)) 先把 a 地址里的内容复制构造给新的 object 对象, 再让 b 指向 new 返回的地址, 运行结果与原程序一样; b(\*new Object(\*a)) 同样先把 a 地址里的内容复制构造给一个新对象, \*new 返回新对象的地址里的内容复制构造给 b, 运行结果比原程序多了一步复制构造和析构, 因为比前两个程序构造时多构造了一个对象, 最后也多析构了一个对象

## 2. (3)

解释: &b(\*new Object(\*a)) 中的 new object 的这个对象没有被 delete, 因为是中间传递的对象, exec 函数结束释放的时候, 被编译器忽略了, 三种情况都没有在函数中明确有一个变量是指这个中间对象的, 所以需要手动 delete, 可以创建一个指针指向这个地址, 再 delete

## 3. (1)

代码:

```

#include <iostream>
#include <vector>

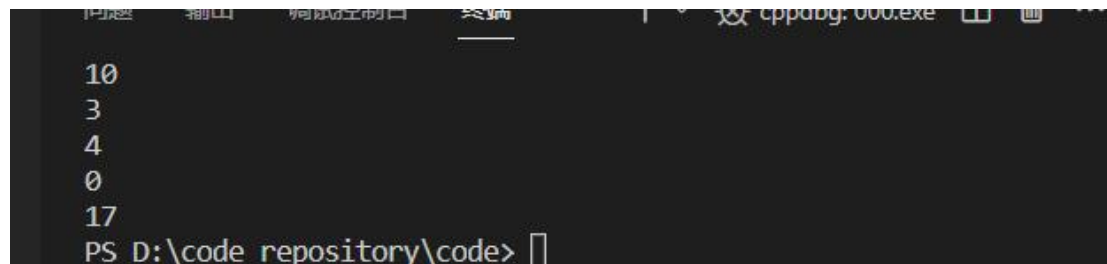
```

```

using namespace std;
int main() {
    vector<int>arr;
    int a;
    cin>>a;
    while(a!=0) {
        arr.push_back(a);
        cin>>a;
    }
    int sum = 0;
    for(int i=0;i<arr.size();i++) {
        sum += arr[i];
    }
    cout<<sum<<endl;
}

```

运行结果：

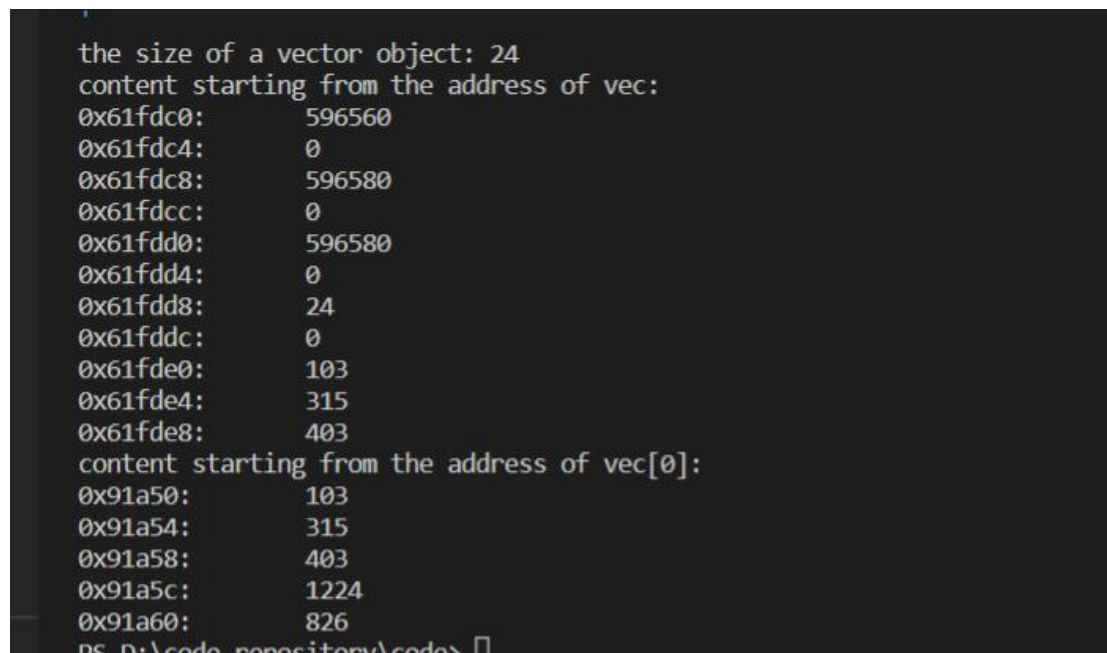


```

10
3
4
0
17
PS D:\code repository\code>

```

### 3. (2)



```

the size of a vector object: 24
content starting from the address of vec:
0x61fdc0: 596560
0x61fdc4: 0
0x61fdc8: 596580
0x61fdcc: 0
0x61fdd0: 596580
0x61fdd4: 0
0x61fdd8: 24
0x61fddc: 0
0x61fde0: 103
0x61fde4: 315
0x61fde8: 403
content starting from the address of vec[0]:
0x91a50: 103
0x91a54: 315
0x91a58: 403
0x91a5c: 1224
0x91a60: 826
PS D:\code repository\code>

```

分析：vector 数组对象名不表示数组首地址，第一个 for 循环输出从 vec 数组对象名指向的地址开始的地址，第二个 for 循环输出从数组的首地址开始的，即各个元素存放的地址，在 C++中，vector 的对象存放在栈区，元素存放在堆区，

且变量名地址中存放的是堆区元素的首地址；而 vec 的 size 与元素个数无关，把元素个数减少一个，它返回的 size 还是 24。

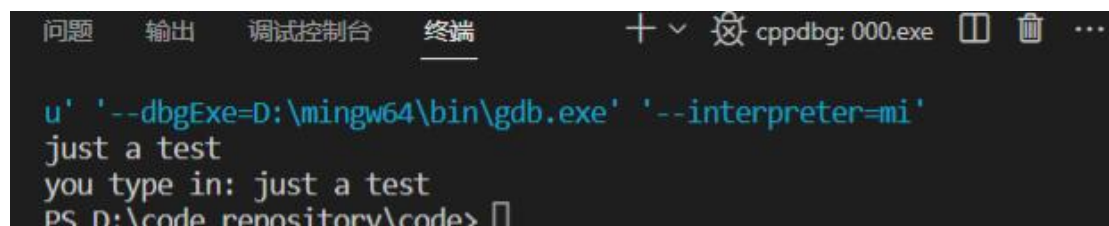
（同一段代码在 vs 中运行结果不一样？Vs 中 size 是 32，而且堆中存放的数据也不太一样，是跟软件和编译器有关吗）

4.

代码：

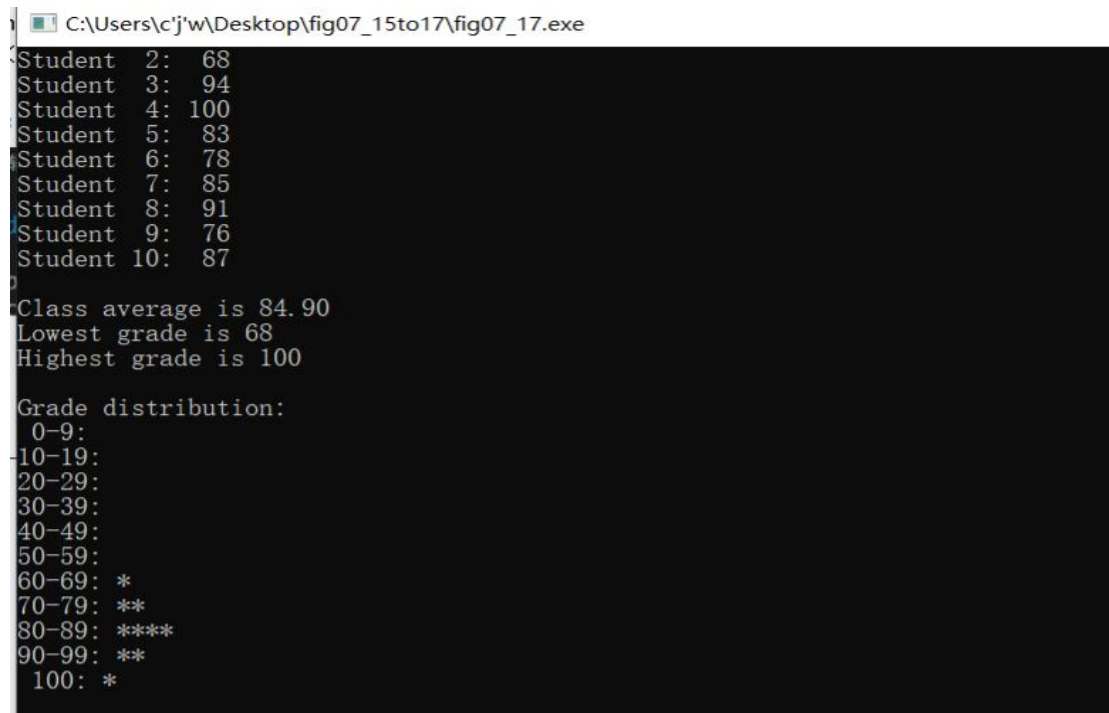
```
#include <iostream>
#include <string>
using namespace std;
int main() {
    string str;
    getline(cin, str);
    string str1 = "you type in: " + str;
    cout<<str1;
}
```

运行结果：



5.

fig07\_15to17



```
C:\Users\c'j'w\Desktop\fig07_15to17\fig07_17.exe
Student 2: 68
Student 3: 94
Student 4: 100
Student 5: 83
Student 6: 78
Student 7: 85
Student 8: 91
Student 9: 76
Student 10: 87
Class average is 84.90
Lowest grade is 68
Highest grade is 100
Grade distribution:
0-9:
10-19:
20-29:
30-39:
40-49:
50-59:
60-69: *
70-79: **
80-89: ****
90-99: **
100: *
```

fig07\_22to24

C:\Users\cjjw\Desktop\fig07\_22to24\07\_24.exe

```
Student 1: 0x70fd88
Student 2: 0x70fd94
Student 3: 0x70fda0
Student 4: 0x70fdac
Student 5: 0x70fdb8
Student 6: 0x70fdc4
Student 7: 0x70fdd0
Student 8: 0x70fddc
Student 9: 0x70fde8
Student 10: 0x70fdf4
```

```
Lowest grade in the grade book is 65
Highest grade in the grade book is 100
```

```
Grade distribution:
```

```
0-9:
10-19:
20-29:
30-39:
40-49:
50-59:
60-69: ***
70-79: *****
80-89: *****
90-99: *****
100: ***
```