湖南大學

数据结构

课程实验报告

题　　目：　学生排队（CCF201703-2）

学生姓名　杨兰馨

学生学号　201708020305

专业班级　通信 1703 班

完成日期　2019/3/12

一、 需求分析

问题描述

体育老师小明要将自己班上的学生按顺序排队。他首先让学生按学号从小到大的顺序排成一排，学号小的排在前面，然后进行多次调整。一次调整小明可能让一位同学出队，向前或者向后移动一段距离后再插入队列。

例如，下面给出了一组移动的例子，例子中学生的人数为 8 人。

0）初始队列中学生的学号依次为 1，2，3，4，5，6，7，8；

1）第一次调整，命令为“3 号同学向后移动 2”，表示 3 号同学出队，向后移动 2 名同学的距离，再插入到队列中，新队列中学生的学号依次为 1，2，4，5，3，6，7，8；

2）第二次调整，命令为“8 号同学向前移动 3”，表示 8 号同学出队，向前移动 3 名同学的距离，再插入到队列中，新队列中学生的学号依次为 1，2，4，5，8，3，6，7；

3）第三次调整，命令为“3 号同学向前移动 2”，表示 3 号同学出队，向前移动 2 名同学的距离，再插入到队列中，新队列中学生的学号依次为 1，2，4，3，5，8，6，7。

小明记录了所有调整的过程，请问，最终从前向后所有学生的学号依次是多少？

请特别注意，上述移动过程中所涉及的号码指的是学号，而不是在队伍中的位置。在向后移动时，移动的距离不超过对应同学后面的人数，如果向后移动的距离正好等于对应同学后面的人数则该同学会移动到队列的最后面。在向前移动时，移动的距离不超过对应同学前面的人数，如果向前移动的距离正好等于对应同学前面的人数则该同学会移动到队列的最前面。

1. 问题分析

(1) 该问题要处理的对象是根据输入的人数，形成的由 int 型元素所组成的线性表。输入数据的类型是 int 型，其中数据范围为：人数在[1,1000]之间,命令数（即对线性表进行改变的次数在[1，1000]之间。

(2) 要实现的功能：根据输入的命令（学号，向前/向后移动的位置）来对线性表进行处理：对该学号的位置在线性表中进行相应的移动。

(3) 对操作完成后的线性表按顺序输出。

2. 输入数据

输入的第一行包含一个整数 n，表示学生的数量，学生的学号由 1 到 n 编号。

第二行包含一个整数 m，表示调整的次数。

接下来 m 行，每行两个整数 p，q，如果 q 为正，表示学号为 p 的同学向后移动 q，如果 q 为负，表示学号为 p 的同学向前移动-q。

3. 输出数据

输出一行，包含 n 个整数，相邻两个整数之间由一个空格分隔，表示最

终从前向后所有学生的学号。

4. 测试样例设计

测试样例 1	
设计目的	普通样例
样例输入	8 3 3 2 8 -3 3 -2
说明	最开始线性表中的元素排列是：1 2 3 4 5 6 7 8。 第一条命令3 号向后移动两个位置后变为：1 2 4 5 3 6 7 8； 第二条命令8 号向前移动3 个位置后变为：1 2 4 5 8 3 6 7； 第三条命令3 号向前移动两个位置后变为：1 2 4 3 5 8 6 7。

测试样例 2	
设计目的	存在学生向前移动的个数和他前面人数相等的情况
样例输入	10 3 5 -4 3 -2 8 2
说明	学生向前移动的个数和他前面人数相等，他就变成队列的 第一个结点中存储的元素。

测试样例 3	
设计目的	存在学生向后移动的个数和他前面人数相等的情况
样例输入	15 2 8 7 4 -3
说明	学生向后移动的个数和他前面人数相等，他就变成队列的

	最后一个结点中存储的元素。

测试样例 4	
设计目的	输入移动位置为 0，即 q=0 的情况
样例输入	9 2 5 0 6 -3
说明	移动位置 q=0，队列不发生变化

测试样例 5	
设计目的	输入较大数据
样例输入	1000 2 99 188 999 -998
说明	能对较大数据进行处理

二、概要设计

1.抽象数据类型

为实现上述程序的功能，可以用整数存储用户的输入。将每一数据作为一个结点存储于线性表（链表）中。

抽象数据类型设计：

数据对象： n 个结点

数据关系： 每个结点从头节点开始依次存储根据输入值 n 生成的整数数据（1—>n）。满足线性特征。

基本操作：

1. 定义一个 append 函数，在结点的尾部添加结点，进而实现对链表的存储；
2. 定义一个 func 函数，根据输入的命令对学生进行相应的移位，其中包括的操作有链表结点的插入和删除；
3. 定义一个 print 函数，对链表中的元素进行遍历输出。

ADT IntegerSet {

数据对象：D = { ai | ai 为整数，i = 1,2，…，n，n>=0}

数据关系：R = {< ai-1，ai> | ai-1，ai 属于 D}

基本操作：
LList()//构造函数初始化；
~LList()//析构函数；
void append (int e)//操作功能：在结点的尾部添加结点（新节点的 element 域等于参数 e 的值）
void func（int e,int n,int q）//操作功能：按要求执行增删操作；
Void print()//操作功能：遍历链表并输出链表的 element 的值。
}

2.算法的基本思想

本题应用了链表这种数据结构来解决问题，因此，基于链表的特点，方便进行插入和删除的操作。根据题意，可以将每一次的操作分为三步：

1. 找到 element 域为 p 的结点，然后删除该结点，并记录该结点前一个结点的位置 pos；

2. 根据得到的 pos 值，加上 q 的值，因为 q 本身有正负，正数即为位置增大，进行右移，负数即为位置减小，位置左移，此时得到了新的 pos 值；

3. 找到新的 pos 值对应的结点，在该节点后面插入新的结点，并且定义新的结点的 element 域即为 p 的值。

执行完先删除后插入的操作之后，此时的新链表就是执行了一次操作之后的，然后再重复执行该函数 m 次，每次传递给该函数的参数根据输入的值变化，最后得到的链表中储存的新的线性表即为所求结果。

3.程序的流程

程序由三个模块组成：

（1） 输入模块：无提示语句，第一行输入总数 n，第二行一次输入执行次数 m，接下来 m 行，每行两个整数 p，q 分别对应需要移动的元素以及如何移动；

（2） 处理模块：将学号按 1-n 分别存储在链表中，根据命令，对应元素所在结点进行删除和插入；

（3） 输出模块：依次输出链表中的结点的 element 域；

三、详细设计

1.物理数据类型

由于输入的数据是整数，且规定评测用例，1<=n<=1000,1<=m<=1000.所以数据型用 int。由于根据输入的 n 由学号为 1-n 的n 个学生，学生（学号是依次排列的）满足前后相继的线性特征，所以物理结构数据为线性结构。

定义的数据类型的伪代码：

```
typedef struct node{
  int element;
  struct node *next;
}Node;//定义单链表结构
```

```
class LList
{
  public:
      int listsize;
      Node *head;
      LList()
      {
          head=curr=NULL;
          listsize=0;
      }
      ~LList();
      void append(int e);
      void func(int e,int n,int q);//执行题目所需所有要求;
      void print();
};
```

2.输入和输出的格式

输入的第一行包含一个整数 n，表示学生的数量，学生的学号由 1 到 n 编号。第二行包含一个整数 m，表示调整的次数。接下来 m 行，每行两个整数 p, q，如果 q

为正，表示学号为 p 的同学向后移动 q，如果 q 为负，表示学号为 p 的同学向前移动-q。

输出一行，包含 n 个整数，相邻两个整数之间由一个空格分隔，表示最终从前向后所有学生的学号

3.算法的具体步骤

总结概括算法的步骤为：

1.找到 element 域为 p 的结点，然后删除该结点，并记录该结点前一个结点的位置 pos；

2.根据得到的 pos 值，加上 q 的值，因为 q 本身有正负，正数即为位置增大，进行右移，负数即为位置减小，位置左移，此时得到了新的 pos 值；

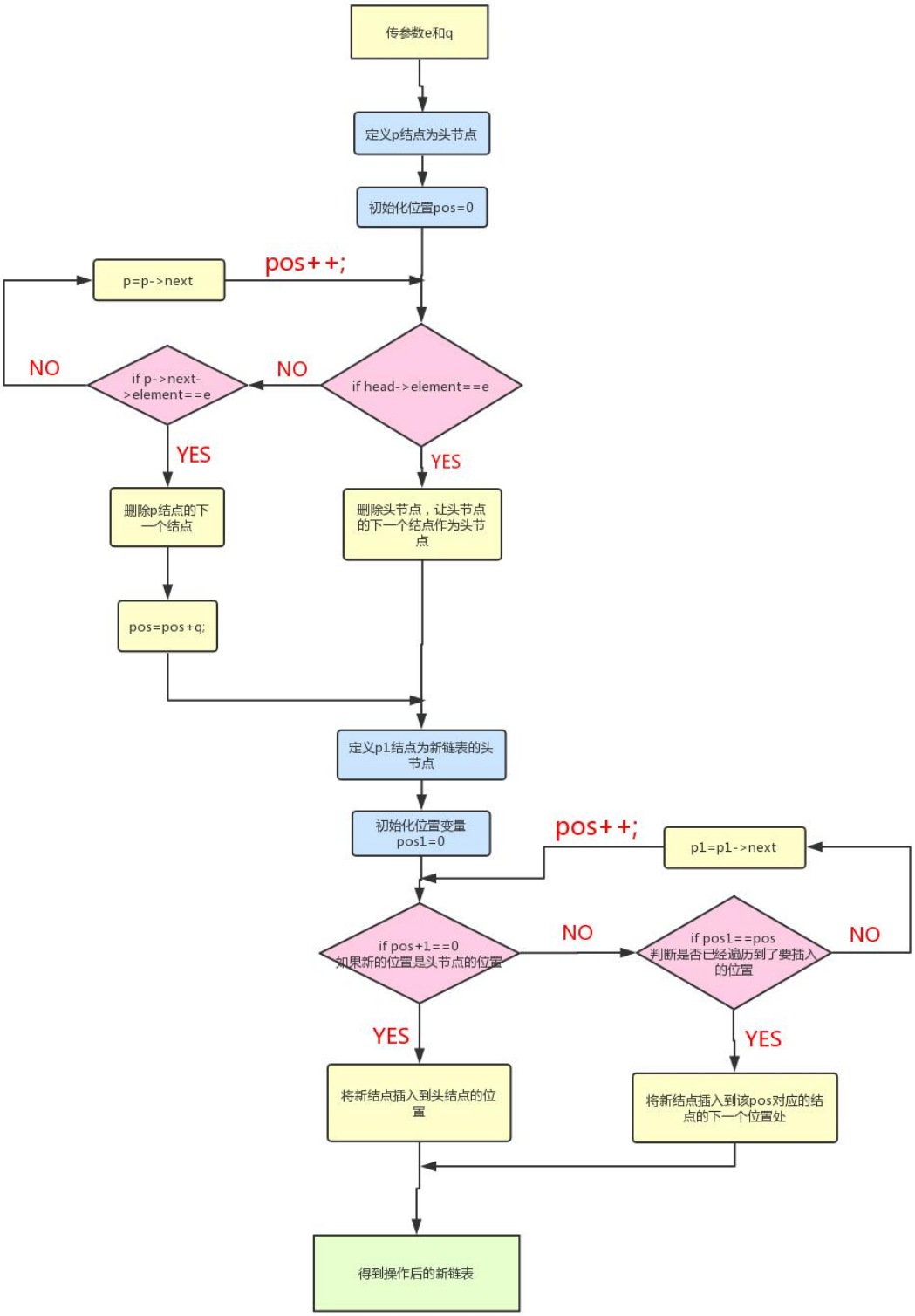
3.找到新的 pos 值对应的结点，在该节点后面插入新的结点，并且定义新的结点的 element 域即为 p 的值。

伪代码如下：

```
void LList::func(int e,int n,int q)
{
    Node *p=head;
    int pos=0;
    Node *a;
```

```
    while(p!=NULL)
    {
        if(head->element==e)
        {
            head=head->next;
            delete p;
            break;
        }
        else if(p->next->element==e)
        {
            Node *q1;
            q1=p->next;
            p->next=p->next->next;
            pos=pos+q;
            delete q1;
            break;
        }
        else p=p->next;
        pos++;
    }
    Node *p1=head;
    int pos1=0;
    while(pos1!=n-1)
    {
        if(pos+1==0)
        {
            a= (Node *)new Node[1];
            a->element=e;
            a->next=p1;
            head=a;
            break;
        }
        else if(pos1==pos)
        {
            a= (Node *)new Node[1];
            a->element=e;
            a->next=p1->next;
            p1->next=a;
            break;
        }
        else p1=p1->next;
        pos1++;
    }
}
```

对于程序中的处理模块，进行流程图描述：



4. 算法的时空分析

1）.输入学生人数的、存储链表数据、输入命令条数的基本操作时间复杂度是Θ(n)；

2）.循环 m 次依次输入命令，在每一条命令执行过程中，进行删除结点，插入结点的操作，时间复杂度为Θ(mn)；

3）. 依次输出的时间复杂度为Θ（n）；

4）.综上所述时间复杂度为Θ(nm).

四、调试分析

1.调试方案设计

调试目的：

测试程序是否可运行，发现代码的语法错误、链接错误、逻辑错误、和运算错误， 是否有不严谨之处。

测试样例：

8
3
3 2
8 -3
3 -2

调试计划：

设定好断点，单步执行，add watch 观察变量 pos 和 pos1 的值，以及观察 p->element 和 p1->element 的值。想办法对算法进行优化，寻找问题，检查遗漏。

设置断点：

在第一次输入命令后设置断点，之后单步执行，调用函数，跳转到对应函数中执行命令，观察对命令的执行情况（删除，插入）是否正确，及时进行修改，对代码的不完善之处进行修改。

2.调试过程和结果，及分析

第一次在编译通过，开始循行程序的时候，程序没办法输出，一直都是在执行过程中，可见是程序的某一个循环不能终止，但是我刚开始没意识到是死循环的问题，所以我开始进行调试，后来发现，在执行到第一个 while 循环的时候，循环一直执行，而这个错误的原因在于进行插入结点的时候定义新结点的语句，改掉之后，程序果然可以正常输出了，并且样例执行正确；

在测试特殊情况的一些样例的时候，比如说将某一个学生移到队伍的最前面或者最后面的时候，输出的结果就是错误的。于是我开始了第二次调试，第二次调试并没有调试完就可以发现错误的原因了，原因是 pos 的值在处理上出了问题，在进行改正之后发现“将学生移到队伍的最前面”这一个样例对了，但是将学生移到队伍的最后面这个样例错误。

于是我开始了第三次调试，很容易就发现了，是第二个 while 循环的终止条件设置错误。经过改正，终于将代码 AC 了。

五、测试结果

样例 1：
普通数据，得到预期效果：

```
8
3
3 2
8 -3
3 -2
1 2 4 3 5 8 6 7

--------------------------------
Process exited after 19.79 seconds with return value 0
请按任意键继续. . .
```

样例 2：
移动到队首，得到预期效果：

```
10
3
5 -4
3 -2
8 2
5 3 1 2 4 6 7 9 10 8

--------------------------------
Process exited after 13.7 seconds with return value 0
请按任意键继续. . .
```

样例 3：
移动到队尾，得到预期结果：

```
15
2
8 7
4 -3
4 1 2 3 5 6 7 9 10 11 12 13 14 15 8

--------------------------------
Process exited after 13.48 seconds with return value 0
请按任意键继续. . .
```

样例 4：
移动位置为 0，得到预期效果。

```
9
2
5 0
6 -3
1 2 6 3 4 5 7 8 9

--------------------------------
Process exited after 8.504 seconds with return value 0
请按任意键继续. . .
```

样例 5：
较大数据，得到预期效果：

```
1000
2
99 188
999 -998
999 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 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 98 100 101 102 103 104 105 1
06 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 14
5 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184
 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 2
63 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 99 288 289 290 291 292 293 294 295 296 297 298 299 300 301
 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 3
80 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 41
9 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458
 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 5
37 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 57
6 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615
 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 6
94 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 73
3 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772
 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811
812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 8
51 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 89
0 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929
 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968
969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 1000

--------------------------------
Process exited after 22.43 seconds with return value 0
请按任意键继续. . .
```

六、实验日志

2019 年 3 月 24 日星期日，作业截止的最后一天，从最开始的选择题目到现在大约过去了十天，中间有好几天的时间没有去做，具体开始写代码反而是在昨天上午，然后就这样一直改代码改到了今天，错误层出不穷，调试来调试去，终于将这个代码跑成功了。

我之所以选择了这一个题目，是因为因为这个题目的数据的线性特征比较明显，可以用顺序表和链表两种方式解决，刚好和我们现在学习的 ADT 能呼应上。由于这个题目涉及到了删除和插入的操作，所以我选择了链表。在理清思路之后，首先我面对的第一个难题就是线性表的定义，参考了实验一的实验要求和课程资料之后，我开始从中过滤有用的信息，我需要的函数有链表结点的插入和删除，还有存储和遍历，但是又不完全和书上的代码一样，于是在这个题目的基础之上，我写了一个新的函数，包含了插入和删除结点等符合该题目的操作的函数。

通过这个题目，我是真的从理论上理解链表晋升到真实意义上的理解并运用链表。

除此之外的另一个收获就是，在面对代码的层出不穷的错误的时候，只要我们有耐心 debug step by step 就一定可以将程序运行正确并且成功。