

# 项目文档

谈瑞

项目二：约瑟夫生者死者游戏

目录

项目简介 \_\_\_\_\_ 3

    项目概要 \_\_\_\_\_ 3

    项目功能及要求 \_\_\_\_\_ 3

项目结构 \_\_\_\_\_ 4

项目类的实现 \_\_\_\_\_ 5

    Joseph 类 \_\_\_\_\_ 5

    Passenger 类 \_\_\_\_\_ 6

运行测试 \_\_\_\_\_ 7

    正常运行 \_\_\_\_\_ 7

    检测可能出现的输入错误 \_\_\_\_\_ 7

## 项目简介

### 项目概要

约瑟夫生者死者游戏的大意是：30 个旅客同乘一条船，因为严重超载，加上风高浪大危险万分；因此船长告诉乘客，只有将全船一半的旅客投入海中，其余人才能幸免于难。无奈，大家只得统一这种方法，并议定 30 个人围成一圈，由第一个人开始，依次报数，数到第 9 人，便将他投入大海中，然后从他的下一个人数起，数到第 9 人，再将他投入大海，如此循环，直到剩下 15 个乘客为止。问哪些位置是将被扔下大海的位置。

### 项目功能及要求

本游戏的数学建模如下：假如  $N$  个旅客排成一个环形，依次顺序编号  $1, 2, \dots, N$ 。从某个指定的第  $S$  号开始。沿环计数，每数到第  $M$  个人就让其出列，且从下一个人开始重新计数，继续进行下去。这个过程一直进行到剩下  $K$  个旅客为止。

本游戏要求用户输入的内容包括：

1. 旅客的个数，也就是  $N$  的值；
2. 离开旅客的间隔数，也就是  $M$  的值；
3. 所有旅客的序号作为一组数据要求存放在某种数据结构中。

本游戏要求输出的内容是包括：

1. 离开旅客的序号；
2. 剩余旅客的序号。

## 项目结构

---

本项目采用单循环链表实现，基本上就是考察链表的插入、删除等基本操作，应该说写起来很顺畅，按照题设思路一步一步往下做就能写成功。

首先通过用户输入约瑟夫环的几个限制条件总人数、开始位置、死亡数字、剩余人数，同时判断输入是否合理，不合理则要求重新输入。而后通过这些数据构建链表完成游戏过程最后输出死亡的人以及剩余人的位置。

# 项目类的实现

## Joseph 类

采用正交循环双向链表构建迷宫

类成员		作用
public 成员	Joseph ();	Joseph 类的默认构造函数，此处用户输入约瑟夫环的限制条件
	~Joseph ();	Joseph 类的默认析构函数，在程序结束后释放链表中所有在游戏中存活下的结点
	void setNumbers(int, const int);	设置限制条件，同时判断输入是否合理，若不合理则重新输入
	Passenger* setJosephNode(int);	递归设置约瑟夫环结点，最后构建约瑟夫环
	Passenger* setBeginNode(int);	根据开始位置设置开始结点
	Passenger* deleteNode(Passenger*);	删除结点，即杀人
	void startKilling(Passenger*);	游戏开始入口
private 成员	Passenger *present, *head, *tail;	头尾结点、当前结点
	int amount, firstIndex, deathNumber, amountLeft;	约瑟夫环的限制条件

Passenger 类

Passenger 类存储了迷宫的结点

类成员		作用
public 成员	Passenger();	Passenger 类的默认构造函数
	Passenger(int, Passenger*);	Passenger 类的重载构造函数，通过参数 number 及 next 指针构造实例
	~Passenger();	Passenger 类的默认析构函数
private 成员	int number;	旅客的位置编号
	Passenger* next;	指向下一个旅客的指针

# 运行测试

## 正常运行

```
D:\tanrui\DataStructure\object2>main.exe
现有N人围成一圈，从第S个人开始依次报数，报M的人出局，再由下一人开始报数，如此循环，直至剩下K个人为止！

请输入生死游戏的总人数N:      30
请输入游戏开始的位置S:         1
请输入死亡数字M:                 9
请输入剩下的生者人数K:         15

第1个死者的位置是:              9
第2个死者的位置是:             18
第3个死者的位置是:             27
第4个死者的位置是:              6
第5个死者的位置是:             16
第6个死者的位置是:             26
第7个死者的位置是:              7
第8个死者的位置是:             19
第9个死者的位置是:             30
第10个死者的位置是:            12
第11个死者的位置是:            24
第12个死者的位置是:              8
第13个死者的位置是:            22
第14个死者的位置是:              5
第15个死者的位置是:            23

最后剩下:      15人
剩余生者的位置分别是:
1      2      3      4      10      11      13      14      15      17      20      21      25      28      29
游戏结束! (*~*)
D:\tanrui\DataStructure\object2>
```

## 检测可能出现的输入错误

对于给定的总人 N 数来说，S、M 和 K 均被限定下来了，看代码段：

```
case 0: {
    while (_input < 1 || _input > MAXAMOUNT) {
        cout << "输入的总人数有误，请重新输入: ";
        cin >> _input;
    }
    amount = _input;
    break;
}
```

总人数 amount 的取值范围为 1-MAXAMOUNT（这里宏定义为 20000）闭区间

```
case 1: {
    while (_input < 1 || _input > amount) {
        cout << "输入的开始位置有误，请重新输入: ";
        cin >> _input;
    }
    firstIndex = _input;
    break;
}
```

开始位置 firstIndex 的取值范围为 1-amount 闭区间

```
case 2: {
    while (_input <= 1) {
        cout << "输入的死亡数字有误，请重新输入：";
        cin >> _input;
    }
    if (_input > amount) _input = _input % amount;
    deathNumber = _input;
    break;
}
```

死亡数字 deathNumber 的取值范围为>1 的整数，若 deathNumber 大于 amount，则将其置为用其对 amount 取模的结果即可。

```
case 3: {
    while (_input >= amount || _input <= 0) {
        cout << "输入的剩余人数有误，请重新输入：";
        cin >> _input;
    }
    amountLeft = _input;
    break;
}
```

剩余的人数 amountLeft 的取值范围为 1-amount 的开区间

如图所示，错误的输入均可被检测出，并要求重新输入。

```
游戏结束! (*~*)
D:\tanrui\DataStructure\object2>main.exe
现有N人围成一圈，从第S个人开始依次报数，报M的人出局，再由下一人开始报数，如此循环，直至剩下K个人为止！

请输入生死游戏的总人数N: 20001
    总人数取值范围为1-20000闭区间
输入的总人数有误，请重新输入: -1
    总人数的取值范围为1-20000闭区间
请输入游戏开始时的位置S: 0
    开始位置取值范围为1-amount闭区间
输入的起始位置有误，请重新输入: 20001
    输入的起始位置有误，请重新输入: 20000
请输入死亡数字M: 0
    死亡数字为大于1的整数，若大于amount对其取模，这里20005对20000取模结果为5
输入的死亡数字有误，请重新输入: -1
    输入的死亡数字有误，请重新输入: 20005
请输入剩余的人数K: 0
    剩余的人数的取值范围为1-amount的开区间
输入的剩余人数有误，请重新输入: 20001
    输入的剩余人数有误，请重新输入: 20000
输入的剩余人数有误，请重新输入: 19999

第1个死者位置是: 4

最后剩下: 19999人
剩余生者的位置分别是:
1 2 3
27 28 29 30 31
52 53 54 55 56
77 78 79 80 81
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
```