

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

 Josephth 类 _____ 5

 Passenger 类 _____ 6

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
请输入剩余的生者人数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L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
输入的总人数有误，请重新输入: 20000

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

请输入剩余的生存人数L: 20001
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: 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

剩余生存者的位置分别是:

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	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	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	851	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	890	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	999	1000	1001	1002	1003	1004	1005	1006	1007	1008	1009	1010	1011	1012	1013	1014
1015	1016	1017	1018	1019	1020	1021	1022	1023	1024	1025	1026	1027	1028	1029	1030	1031	1032	1033	1034	1035	1036	1037	1038	1039	1040
1041	1042	1043	1044	1045	1046	1047	1048	1049	1050	1051	1052	1053	1054	1055	1056	1057	1058	1059	1060	1061	1062	1063	1064	1065	1066
1067	1068	1069	1070	1071	1072	1073	1074	1075	1076	1077	1078	1079	1080	1081	1082	1083	1084	1085	1086	1087	1088	1089	1090	1091	1092
1093	1094	1095	1096	1097	1098	1099	1100	1101	1102	1103	1104	1105	1106	1107	1108	1109	1110	1111	1112	1113	1114	1115	1116	1117	1118
1119	1120	1121	1122	1123	1124	1125	1126	1127	1128	1129	1130	1131	1132	1133	1134	1135	1136	1137	1138	1139	1140	1141	1142	1143	1144
1145	1146	1147	1148	1149	1150	1151	1152	1153	1154	1155	1156	1157	1158	1159	1160	1161	1162	1163	1164	1165	1166	1167	1168	1169	1170
1171	1172	1173	1174	1175	1176	1177	1178	1179	1180	1181	1182	1183	1184	1185	1186	1187	1188	1189	1190	1191	1192	1193	1194	1195	1196
1197	1198	1199	1200	1201	1202	1203	1204	1205	1206	1207	1208	1209	1210	1211	1212	1213	1214	1215	1216	1217	1218	1219	1220	1221	1222
1223	1224	1225	1226	1227	1228	1229	1230	1231	1232	1233	1234	1235	1236	1237	1238	1239	1240	1241	1242	1243	1244	1245	1246	1247	1248
1249	1250	1251	1252	1253	1254	1255	1256	1257	1258	1259	1260	1261	1262	1263	1264	1265	1266	1267	1268	1269	1270	1271	1272	1273	1274
1275	1276	1277	1278	1279	1280	1281	1282	1283	1284	1285	1286	1287	1288	1289	1290	1291	1292	1293	1294	1295	1296	1297	1298	1299	1300
1301	1302	1303	1304	1305	1306	1307	1308	1309	1310	1311	1312	1313	1314	1315	1316	1317	1318	1319	1320	1321	1322	1323	1324	1325	1326
1327	1328	1329	1330	1331	1332	1333	1334	1335	1336	1337	1338	1339	1340	1341	1342	1343	1344	1345	1346	1347	1348	1349	1350	1351	1352
1353	1354	1355	1356	1357	1358	1359	1360	1361	1362	1363	1364	1365	1366	1367	1368	1369	1370	1371	1372						