目录

**V0.0 ：从无到有 2**

**V0.1 ：命名准备** **2**

**V0.2 ：准备动手** **2**

**V0.3 ：字斟句酌 3**

**V0.4 ：初具雏形 3**

**V0.5 ：胜利在望 4**

**V0.6 ：加大力度** **8**

**V0.7 ：箭在弦上** **9**

**V1.0 ：横空出世** **11**

**V1.1 ：锦上添花** 12

医院新冠状肺炎病人管理系统项目日志

V0.0 ：从无到有

分析该系统大致功能需求：

1. 分别建立两个结构体（现存感染病人：病人号码，病人名，入院时间，籍贯，年龄，病情现状，病人状态判断变量（新增）；痊愈病人和死亡病人：病人号码，病人名，入院时间，籍贯，年龄，出院时间（死亡时间）；）
2. 菜单栏：现存病人，痊愈和死亡病人，增添现存病人，增添痊愈和死亡病人，查找病人信息，退出系统，删除现存病人，删除痊愈和死亡病人。

3、增添现存病人、增添痊愈和死亡病人。

4、打印现存病人、打印痊愈和死亡病人列表。

5、查询病人信息（病人号码，病人名）（精确查找与模糊查找）；条件查找（病人号码，入院时间等条件）查询多个病人信息；多条件查询.

6、拓展功能：病人信息批量导入、时间段新增病人统计。

V0.1 ：命名准备

结构体：

病人号码——patientsNum

病人名——patientsName

入院时间——admissionTime

籍贯——nativePlace

年龄——age

病情现状——patientsSituation

出院/死亡时间——chageTime

标记flag——1（现存）2（痊愈）3（死亡）4（所有）

函数名：

创建表头——creatHead

新建结点——creatPatients

打印表——showPatients

打印菜单——Menu

键入选项——keyInput

查找病人——findPatients

更新病情——updatePatients

退出系统——exitSystem

删除病人信息——deletePatients

V0.2 ：准备动手

1、将主界面（菜单）写成独立函数。

1. **void** menu()
2. {
3. printf("====================【医院新冠状肺炎病人管理系统】====================\n");
4. printf("\t  \t\t1.现存病人信息\n");
5. printf("\t  \t\t2.痊愈和死亡病人信息\n");
6. printf("\t  \t\t3.查询病人信息\n");
7. printf("\t  \t\t4.删除病人信息\n");
8. printf("\t  \t\t5.增加病人信息\n");
9. printf("\t  \t\t6.现存病人信息修改\n");
10. printf("\t  \t\t7.退出系统\n");
11. printf("======================================================================\n");
12. }

~~2、同时将0.0计划的两个结构体设想否决，痊愈或死亡病人数据和现存病人信息只有一项不同，可将痊愈或死亡病人的病情现况改为已痊愈-（时间）/已死亡-（时间），在展示这些病人信息时候可以通过遍历链表并且判断对应项前几个字是否为已痊愈||已死亡，然后输出。~~

V0.3 ：字斟句酌

1、更新对于现存病人转为已治愈||已死亡的判定方式，在结构体中设置变量flag来判断病人状态：flag = 1 表示现存病人，flag = 2 表示痊愈病人，flag = 3 表示死亡病人，flag = 4 表示所有病人。

2、在菜单中新增打印所有病人信息。

V0.4 ：初具雏形

1. 写好结构体和链表框架
2. **typedef** **struct** person
3. {
4. **int** patientsNum;
5. **char** patientsName[10];
6. **char** admissionTime[12];
7. **char** nativePlace[20];
8. **int** age;
9. **char** patientsSituation[20];
10. **char** changeTime [20];
11. **int** flag;           //flag  1——现存  2——痊愈  3——死亡
12. }patients;
13. **typedef** **struct** link
14. {
15. patients p;
16. **struct** link \*next;
17. }Node;
18. 创建表头
19. Node \*creatHead()
20. {
21. Node \*head = (Node \*)malloc(**sizeof**(Node));
22. head ->next = NULL;
23. **return** head;
24. }
25. 创建结点
26. **void** creatPatients(Node \*head)
27. {
28. Node \*newnode = (Node\*)malloc(**sizeof**(Node));
29. newnode ->next = head ->next;
30. head ->next = newnode;
31. printf("请输入病人编号：");
32. scanf("%d",&newnode ->p.patientsNum);
33. printf("请输入病人名字：");
34. scanf("%s",newnode ->p.patientsName);
35. printf("请输入入院时间：");
36. scanf("%s",newnode ->p.admissionTime);
37. printf("请输入病人籍贯：");
38. scanf("%s",newnode ->p.nativePlace);
39. printf("请输入病人年龄：");
40. scanf("%d",&newnode ->p.age);
41. printf("请输入病人的病情现状：");
42. scanf("%s",newnode ->p.patientsSituation);
43. newnode ->p.flag = 1;
44. }

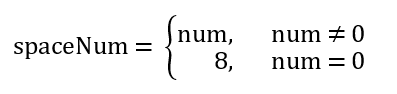
4、打印表格（分类打印，通过flag判断其病情状况）

//教训：每次改进代码时多测试运行并备份，以防被小错误无限耽搁时间

//经验：\t代表的不一定是固定数目的空格

令  num = |n-8|%8， 其中n表示\t前面的字符占的位置（前面的字符也可能是转换说明，如%d，%10d等）。

那么\t对应的空格的个数 spaceNum,有



1. **void** showPatients(Node \*head,**int** n)
2. {
3. Node \*temp = head->next;
5. printf("\t病人号码\t病人名\t入院时间  \t籍贯\t\t\t年龄\t病情现状\t痊愈或死亡时间\n");
6. **while**(temp)
7. {
8. **if**((temp ->p.flag == n && n ==1) ||(n == 4 && temp ->p.flag == 1))
9. {
10. printf("\t%-8d\t%-6s\t%-10s\t%-20s\t%-2d  \t%-16s\n",temp ->p.patientsNum,temp ->p.patientsName,temp ->p.admissionTime,temp ->p.nativePlace,temp ->p.age,temp ->p.patientsSituation);
12. }
13. **else** **if**((temp ->p.flag == n && n ==2)||(n == 4 && temp ->p.flag == 2))
14. {
16. printf("\t%-8d\t%-6s\t%-10s\t%-20s\t%-2d  \t已痊愈\t\t%-10s\n",temp ->p.patientsNum,temp ->p.patientsName,temp ->p.admissionTime,temp ->p.nativePlace,temp ->p.age,temp ->p.changeTime);
18. }
19. **else** **if**((temp ->p.flag == n && n ==3) || (n == 4 && temp ->p.flag == 3))
20. {
21. printf("\t%-8d\t%-6s\t%-10s\t%-20s\t%-2d  \t已死亡\t\t%-10s\n",temp ->p.patientsNum,temp ->p.patientsName,temp ->p.admissionTime,temp ->p.nativePlace,temp ->p.age,temp ->p.changeTime);
23. }
24. temp = temp ->next;
25. }
27. }

V0.5 ：胜利在望

1. 通过病人编号查找病人
2. Node \*findPatients(Node \*head,**int** num)
3. {
4. Node \*temp = head ->next;
5. **while**(temp)
6. {
7. **if**(temp->p.patientsNum == num)
8. {
9. **return** temp;
10. }
11. temp = temp->next;
12. }
13. **return** NULL;
14. }
15. 更新病情
16. **void** updatePatients(Node \*head)
17. {
18. printf("请输入需要修改病人编号：");
19. **int** num;
20. scanf("%d",&num);
21. Node \*temp = findPatients(head,num);
22. **if**(temp)
23. {
24. printf("病人%s记录病情为：%s\n请输入该病人最新病情:",temp ->p.patientsName,temp ->p.patientsSituation);
25. **char** sit [20];
26. scanf("%s",sit);
27. **if**(strcmp(sit,"已痊愈") == 0)
28. {
29. temp ->p.flag = 2;
31. strcpy(temp ->p.patientsSituation,sit);
32. printf("请输入痊愈时间：");
33. scanf("%s",temp ->p.changeTime);
34. }
35. **else** **if**(strcmp(sit,"已死亡") == 0)
36. {
37. temp ->p.flag = 3;
39. strcpy(temp ->p.patientsSituation,sit);
40. printf("请输入死亡时间：");
41. scanf("%s",temp ->p.changeTime);
42. }
43. **else** strcpy(temp ->p.patientsSituation,sit);
45. }
46. **else** printf("未找到病人信息，即将返回主菜单\n");
47. }
48. 键入信息对应执行函数
49. **void** keyInput(Node \*head)
50. {
51. **int** temp;
52. printf("请输入您需要操作的序号：\n");
53. scanf("%d",&temp);
55. **switch**(temp)
56. {
57. **case** 0:
59. printf("==================================================所有病人信息==================================================\n");
60. showPatients(head,4);
61. system("pause");
62. **break**;
63. **case** 1:
65. printf("==================================================现存病人信息==================================================\n");
66. showPatients(head,1);
67. system("pause");
68. **break**;
69. **case** 2:
70. printf("==================================================痊愈病人信息==================================================\n");
71. showPatients(head,2);
72. printf("==================================================死亡病人信息==================================================\n");
73. showPatients(head,3);
74. system("pause");
75. **break**;
76. **case** 3:
77. **break**;
78. **case** 4:
79. **break**;
80. **case** 5:
81. printf("\t  \t\t增加病人信息\n");
82. creatPatients(head);
83. system("pause");
84. **break**;
85. **case** 6:
86. printf("\t  \t\t现存病人病情信息修改\n");
87. updatePatients(head);
88. system("pause");
89. **break**;
90. **case** 7:
91. **break**;
92. **default**:
93. printf("输入格式有误，请重新输入!\n");
94. system("pause");
95. **break** ;
96. }
97. }
98. 主函数
99. **int** main()
100. {
101. Node \*head = creatHead();
102. **while**(1)
103. {
104. Menu();
105. keyInput(head);
106. system("cls");
108. }

111. **return** 0;
112. }

V0.6 ：加大力度

1. 补充键入函数中的case7
2. **case** 7:
3. printf("是否现在退出系统？请输入Y/N：");
4. **char** c;
5. getchar();
6. scanf("%c",&c);
7. **if**(c =='Y'||c =='y')exit(0);
8. **else** **if**(c =='n'||c =='N')
9. {
10. printf("即将返回主菜单\n");
11. system("pause");
12. **break**;
13. }
14. **else**
15. {
16. printf("输入有误，即将返回主菜单\n");
17. system("pause");
18. **break**;
19. }

V0.7 ：箭在弦上

1、在case 3中调用findPatients实现通过病人编号查询病人信息

1. **case** 3:
2. {
3. printf("==================================================查询病人信息==================================================\n");
4. printf("请输入病人编号：");
5. **int** n;
6. scanf("%d",&n);
7. Node \*temp = findPatients(head,n);
8. printf("以下为编号%d的病人住院信息\n",temp ->p.patientsNum);
9. printf("病人姓名：%s\n",temp ->p.patientsName);
10. printf("病人入院时间：%s\n",temp ->p.admissionTime);
11. printf("病人籍贯：%s\n",temp ->p.nativePlace);
12. printf("病人年龄：%d\n",temp ->p.age);
13. printf("病人病情：%s\n",temp ->p.patientsSituation);

16. **if**(temp ->p.flag == 1){}
17. **else** **if**(temp ->p.flag == 2)printf("痊愈时间为：%s\n",temp ->p.changeTime);
19. **else** **if**(temp ->p.flag == 3)printf("死亡时间为：%s\n",temp ->p.changeTime);
21. }
22. system("pause");
23. **break**;
24. 实现函数deletePatients删除病人信息
25. **void** deletePatients(Node \*head)
26. {
27. printf("==================================================删除病人信息==================================================\n");
28. printf("请输入您需要删除的病人编号：\n");
29. **int** n;
30. scanf("%d",&n);
31. Node \*temp = findPatients(head,n);
32. **if**(temp == NULL )
33. {
34. printf("未找到该病人信息，即将返回主菜单\n");
35. **return** ;
36. }
37. **else**
38. {
39. printf("您确定要删除病人%s的信息？请输入Y/N：",temp ->p.patientsName);
40. **char** c;
41. getchar();
42. scanf("%c",&c);
43. **if**(c =='n'||c == 'N')
44. {
45. printf("即将返回主菜单\n");
46. **return** ;
47. }
48. **else** **if**(c =='y'||c == 'Y')
49. {
50. **if**(temp ->next == NULL)
51. {
52. free(temp);
53. }
55. **else**
56. {
57. temp ->p = temp ->next->p;
58. temp ->next = temp ->next->next;
60. }
61. printf("删除成功，即将返回主菜单\n");
63. }
64. **else** **if**(c !='n'&&c != 'N'&&c !='y'&&c != 'Y')
65. {
66. printf("输入有误，即将返回主菜单\n");
67. **return** ;
68. }
69. }
71. }

V1.0：横空出世

1. 检查键入输出中的每种功能都有system（“pause”）并返回主函数
2. 当前版本尚不支持导入导出，后续增加病人信息导入和导出功能，最好实现信息实时操作，在文本文档内储存。

V1.1：锦上添花

1. 实现内容从文本中导入至链表（用C++实现read函数）
2. **void** read(Node \*head)
3. {
4. ifstream infile;
5. infile.open("date.txt",ios::in);
6. **if**(!infile)
7. {
8. cout<<"打开文件失败！"<<"TEST.txt"<<endl;
9. }
11. **int** x;
12. **while**(infile >> x) //可以像用cin那样用ifstream对象
13. {
14. Node \*newnode = (Node\*)malloc(**sizeof**(Node));
15. newnode ->next = head ->next;
16. head ->next = newnode;
17. **if**(x == 1){
18. newnode->p.flag = 1;
19. infile>>newnode->p.patientsNum;
20. infile>>newnode->p.patientsName;
21. infile>>newnode->p.admissionTime;
22. infile>>newnode->p.nativePlace;
23. infile>>newnode->p.age;
24. infile>>newnode->p.patientsSituation;
25. }**else**{
26. newnode->p.flag = x;
27. infile>>newnode->p.patientsNum;
28. infile>>newnode->p.patientsName;
29. infile>>newnode->p.admissionTime;
30. infile>>newnode->p.nativePlace;
31. infile>>newnode->p.age;
32. infile>>newnode->p.patientsSituation;
33. infile>>newnode->p.changeTime;
34. }
35. }
36. infile.close();
38. }

2、在退出系统时，将链表内的数据写入文本，实现保存功能。通过函数write实现并在退出函数exitSystem中调用。

1. **void** write(Node \*head){
2. Node \*temp = head->next;
3. ofstream out;
4. out.open("date.txt", ios::out);
5. out.clear();
6. **while**(temp){
7. **if**(temp->p.flag == 1){
8. out<<temp->p.flag<<" ";
9. out<<temp->p.patientsNum<<" ";
10. out<<temp->p.patientsName<<" ";
11. out<<temp->p.admissionTime<<" ";
12. out<<temp->p.nativePlace<<" ";
13. out<<temp->p.age<<" ";
14. out<<temp->p.patientsSituation<<endl;
15. }**else**{
16. out<<temp->p.flag<<" ";
17. out<<temp->p.patientsNum<<" ";
18. out<<temp->p.patientsName<<" ";
19. out<<temp->p.admissionTime<<" ";
20. out<<temp->p.nativePlace<<" ";
21. out<<temp->p.age<<" ";
22. out<<temp->p.patientsSituation<<" ";
23. out<<temp->p.changeTime<<endl;
24. }
25. temp = temp->next;
27. }
28. out.close();
30. }