

通讯录管理系统

1. 系统需求

通讯录是一个可以记录亲人、好友信息的工具。

主要利用C++来实现一个通讯录管理系统系统中需要实现的功能如下：

- 添加联系人：向通讯录中添加新人，信息包括(姓名、性别、年龄、联系电话、家庭住址) 最多记录1000人
- 显示联系人：显示通讯录中所有联系人信息
- 删除联系人：按照姓名进行删除指定联系人
- 查找联系人：按照姓名查看指定联系人信息
- 修改联系人：按照姓名重新修改指定联系人
- 清空联系人：清空通讯录中所有信息
- 退出通讯录：退出当前使用的通讯录

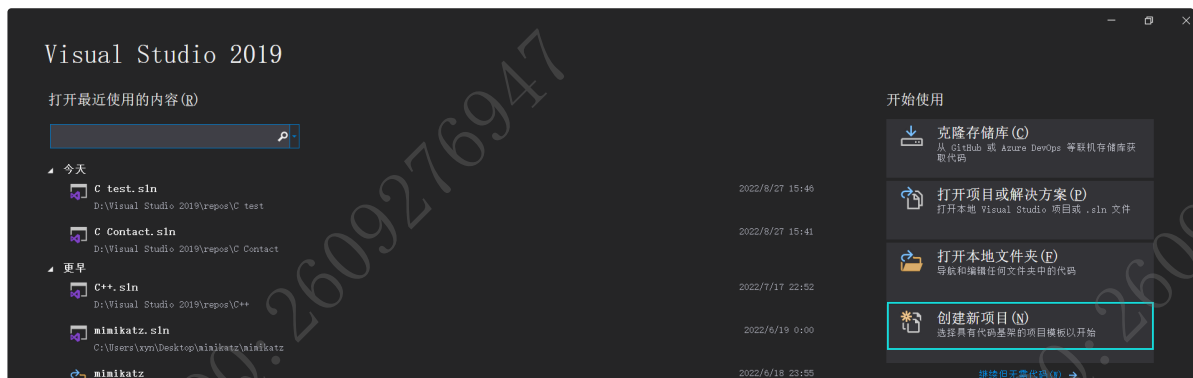
2. 创建项目

创建项目步骤如下：

- 创建新项目
- 添加文件

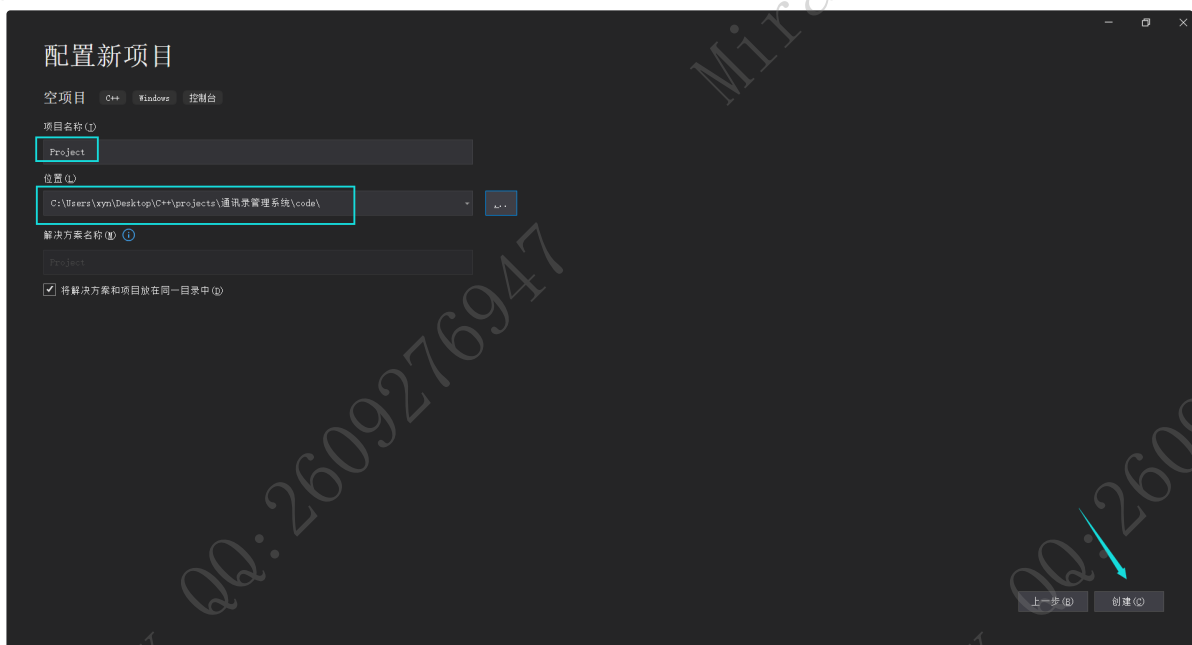
2.1 创建项目

打开Visual Studio 2019后，点击创建新项目，创建新的C++项目



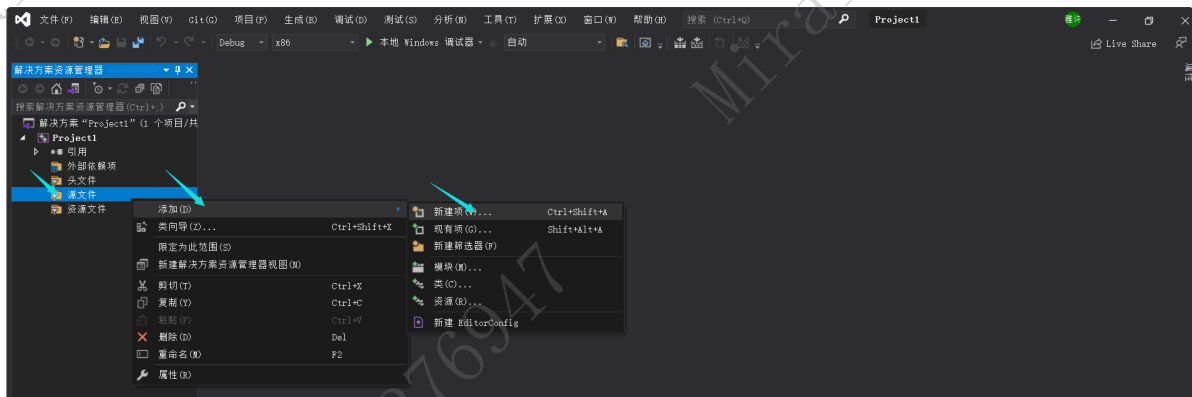
创建项目，填写项目名称，选择项目路径



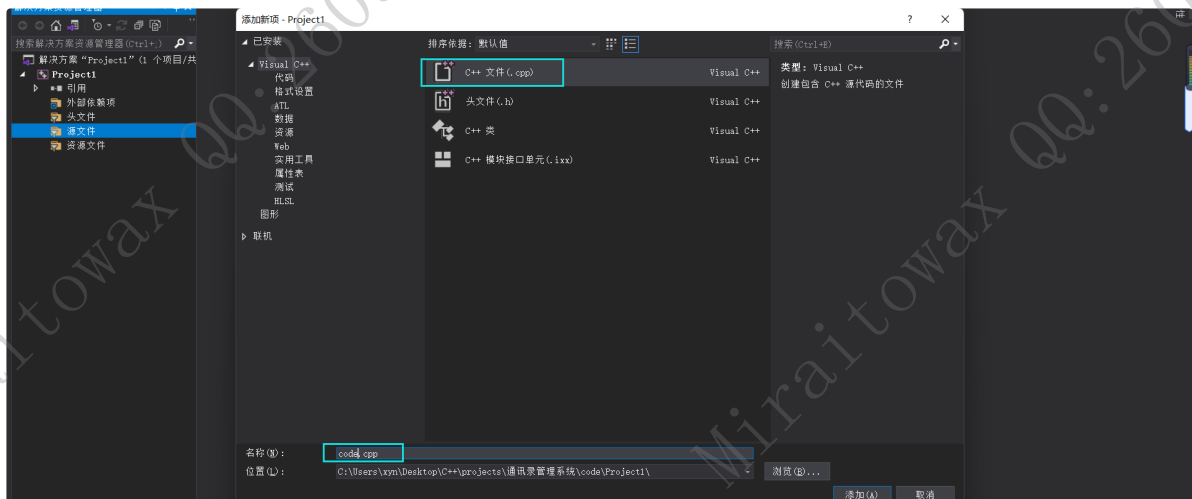


2.2 添加文件

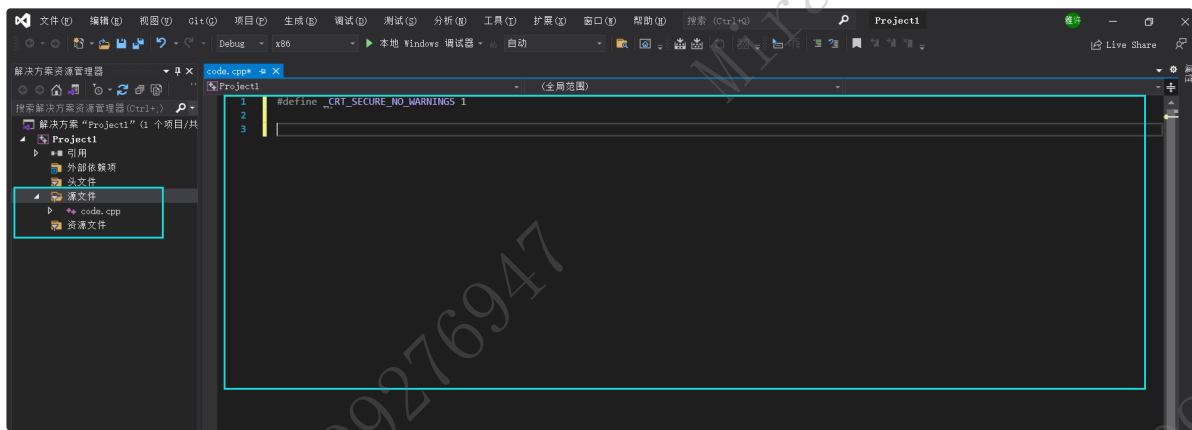
添加源文件，右击源文件添加



命名，文件添加完成



完成后，效果如图：



项目创建完成

3. 菜单功能

功能描述：

用户选择功能的界面

步骤：

- 封装函数显示该界面，例：`showMenu()`
- 在main函数中调用封装好的函数

```
1  #define _CRT_SECURE_NO_WARNINGS 1
2  #include<iostream>
3  using namespace std;
4
5  // 菜单界面显示
6  void showMenu() {
7      cout << "*****" << endl;
8      cout << "***** 1. 添加联系人 *****" << endl;
9      cout << "***** 2. 显示联系人 *****" << endl;
10     cout << "***** 3. 删除联系人 *****" << endl;
11     cout << "***** 4. 查找联系人 *****" << endl;
12     cout << "***** 5. 修改联系人 *****" << endl;
13     cout << "***** 6. 清空联系人 *****" << endl;
14     cout << "***** 0. 退出通讯录 *****" << endl;
15     cout << "*****" << endl;
16 }
17
18 int main() {
19     // 菜单的调用
```

```
20     showMenu();
21     return 0;
22 }
```

4. 退出功能

功能描述:退出通讯录系统

思路:根据用户不同的选择,进入不同的功能,可以选择switch分支结构,将整个架构进行搭建

当用户选择0时候,执行退出,选择其他先不做操作,也不会退出程序

```
1  int main() {
2      //用户选择输入
3      int select = 0;
4      while (true) {
5          // 菜单的调用
6          showMenu();
7          cin >> select;
8          switch (select) {
9              case 1://添加联系人
10                 break;
11              case 2://显示联系人
12                 break;
13              case 3://删除联系人
14                 break;
15              case 4://查找联系人
16                 break;
17              case 5://修改联系人
18                 break;
19              case 6://清空联系人
20                 break;
21              case 0://退出通讯录
22                 cout << "欢迎下次使用\n";
23                 system("pause");
24                 return 0;
25                 break;
26              default:
```

```
27         break;
28     }
29 }
30 return 0;
31 }
```

5. 添加联系人

功能描述：

实现添加联系人功能，联系人上限为1000人，联系人信息包括（姓名、性别、年龄、联系电话、家庭住址）

添加联系人实现步骤：

- 设计联系人结构体
- 设计通讯录结构体
- main函数中创建通讯录
- 封装添加联系人函数
- 测试添加联系人功能

5.1 设计联系人结构体

联系人信息包括（姓名、性别、年龄、联系电话、家庭住址）

```
1 #include<string> //string头文件
2 //联系人结构体
3 struct Person{
4     string m_Name; //姓名
5     int m_Sex; //性别: 1: 男 2: 女
6     int m_Age; //年龄
7     string m_Phone; //电话
8     string m_Addr; //住址
9 }
```

5.2 设计通讯录结构体

设计时候可以在通讯录结构体中，维护一个容量为1000的存放联系人的数组，并记录当前通讯录中联系人数量设计如下

```

1  #define MAX 1000 //最大人数
2  //通讯录结构体
3  struct Addressbooks{
4      struct Person personArray[MAX]; //通讯录中保存的联系人数组
5      int m_Size; //通讯录中人员个数
6  }

```

5.3 main函数中创建通讯录

添加联系人函数封装好之后，在main函数中创建一个通讯录变量，这个就是需要一直维护的通讯录

```

1  //main函数起始位置添加
2  //创建通讯录结构体变量
3  Addressbooks abs;
4  //初始化通讯录中当前人员个数
5  abs.m_Size = 0;

```

5.4 封装添加联系人函数

思路：添加联系人前判断通讯录是否已满，如果满了就不再添加，未满情况将新联系人信息逐个加入到通讯录

```

1  //1. 添加联系人
2  void addPerson(Addressbooks* abs) {
3      //判断通讯录是否已满，如果满了就不再添加
4      if (abs->m_Size == MAX) {
5          cout << "通讯录已满，无法添加！" << endl;
6      }
7      else {
8          //添加具体联系人
9          //姓名
10         string name;
11         cout << "请输入姓名：" << endl;
12         cin >> name;
13         abs->personArray[abs->m_Size].m_Name = name;
14         //性别
15         cout << "请输入性别：" << endl;
16         cout << "1 --- 男" << endl;
17         cout << "2 --- 女" << endl;

```

```
18     int sex = 0;
19     while (true) {
20         cin >> sex;
21         if (sex == 1 || sex == 2) {
22             abs->personArray[abs->m_Size].m_Sex = sex;
23             break;
24         }
25         cout << "输入有误, 请重新出入! " << endl;
26     }
27     //年龄
28     cout << "请输入年龄: " << endl;
29     int age = 0;
30     cin >> age;
31     abs->personArray[abs->m_Size].m_Age = age;
32     //电话
33     cout << "请输入联系电话: " << endl;
34     string phone;
35     cin >> phone;
36     abs->personArray[abs->m_Size].m_Phone = phone;
37     //住址
38     cout << "请输入住址: " << endl;
39     string address;
40     cin >> address;
41     abs->personArray[abs->m_Size].m_Addr = address;
42     //更新通讯录人数
43     abs->m_Size++;
44     cout << "添加联系人成功\n" << endl;
45
46     system("pause");
47     system("cls");//清屏
48 }
49 }
```

6. 显示联系人

功能描述：显示通讯录中已有的联系人信息

显示联系人实现步骤：

- 封装显示联系人函数
- 测试显示联系人功能

6.1 封装显示联系人函数

思路：判断当前通讯录中人员：没有人员，显示记录为空，人数大于0，显示通讯录中的信息

```
1 //2. 显示联系人
2 void showPerson(Addressbooks* abs) {
3     //判断通讯录中人数是否为空，如果为0，提示记录为空
4     //如果不为0，显示记录的联系人信息
5     if (abs->m_Size == 0) {
6         cout << "当前记录为空" << endl;
7     }
8     else {
9         for (int i = 0; i < abs->m_Size; i++) {
10             cout << "姓名: " << abs->personArray[i].m_Name << "\t";
11             cout << "性别: " << (abs->personArray[i].m_Sex == 1 ? "男" :
12             "女") << "\t";
13             cout << "年龄: " << abs->personArray[i].m_Age << "\t";
14             cout << "联系方式: " << abs->personArray[i].m_Phone << "\t";
15             cout << "家庭住址: " << abs->personArray[i].m_Addr << endl;
16         }
17     }
18     system("pause");
19     system("cls");
20 }
```

6.2 测试显示联系人功能

在switch case语句中，case 2中添加：

```
1 showPerson(&abs);
```


7. 删除联系人

功能描述：按照姓名进行删除指定联系人

删除联系人实现步骤：

- 封装检测联系人是否存在
- 封装删除联系人函数
- 测试删除联系人功能

7.1 封装检测联系人是否存在

设计思路：

删除联系人前，我们需要先判断用户输入的联系人是否存在，如果存在删除，不存在提示用户没有要删除的联系人，因此我们可以把检测联系人是否存在封装成一个函数中，如果存在，返回联系人在通讯录中的位置，不存在返回-1。

```
1 //检测联系人是否存在，如果存在，返回联系人所在数组中的具体位置，不存在返回-1
2 int isExit(Addressbooks* abs, string name) {
3     for (int i = 0; i < abs->m_Size; i++) {
4         if (abs->personArray[i].m_Name == name) {
5             return i; //找到，返回下标编号
6         }
7     }
8     return -1; //遍历结束未找到，返回-1
9 }
```

7.2 封装删除联系人函数

根据用户输入的联系人判断该通讯录中是否

有此人查找到进行制除，并提示删除成功

查不到提示查无此人。

```
1 // 3. 删除联系人
2 //检测联系人是否存在，如果存在，返回联系人所在数组中的具体位置，不存在返回-1
3 int isExit(Addressbooks* abs, string name) {
4     for (int i = 0; i < abs->m_Size; i++) {
5         if (abs->personArray[i].m_Name == name) {
6             return i; //找到，返回下标编号
7         }
8     }
9     return -1; //遍历结束未找到，返回-1
10 }
```

```

11 //删除联系人
12 void deletePerson(Addressbooks* abs) {
13     cout << "请输入您要删除的联系人: " << endl;
14     string name;
15     cin >> name;
16     int ret = isExit(abs, name);
17     //ret != -1 查到了
18     //ret == -1 未查到
19     if (ret != -1) {
20         //数据前移
21         for (int i = ret; i < abs->m_Size; i++) {
22             abs->personArray[i] = abs->personArray[i + 1];
23         }
24         //更新通讯录中的人员数
25         abs->m_Size--;
26         cout << "删除成功\n";
27     }
28     else {
29         cout << "查无此人\n";
30     }
31     system("pause");
32     system("cls");
33 }

```

8. 查找联系人

功能描述：按照姓名查看指定联系人信息

查找联系人实现步骤：

- 封装查找联系人函数
- 测试查找指定联系人

8.1 封装查找联系人的函数

实现思路：判断用户指定的联系人是否存在，如果存在显示信息，不存在则提示查无此人。

```

1 //4. 查找指定联系人信息
2 void findPerson(Addressbooks* abs) {
3     cout << "请输入您要查找的联系人: " < endl;

```

```

4     string name;
5     cin >> name;
6     //判断指定的联系人是否存在在通讯录中
7     int ret = isExit(abs);
8     if (ret != -1) { //找到
9         cout << "姓名: " << abs->personArray[ret].m_Name << "\t";
10        cout << "性别: " << (abs->personArray[ret].m_Sex == 1 ? "男" : "女")
11        << "\t";
12        cout << "年龄: " << abs->personArray[ret].m_Age << "\t";
13        cout << "联系方式: " << abs->personArray[ret].m_Phone << "\t";
14        cout << "家庭住址: " << abs->personArray[ret].m_Addr << endl;
15    }
16    else { //未找到
17        cout << "查无此人\n" << endl;
18    }
19    system("pause");
20    system("cls");
21 }

```

8.2 测试查找指定联系人

在switch case语句中, case 4中添加:

```

1 findPerson(&abs);

```

9. 修改联系人

功能描述: 按照姓名重新修改指定联系人

修改联系人实现步骤:

- 封装修改联系人函数
- 测试修改联系人功能

9.1 封装修改联系人函数

实现思路: 查找用户输入的联系人, 如果查找成功进行修改操作, 查找失败提示查无此人

```

1 // 5. 修改指定联系人信息
2 void modifyPerson(Addressbooks* abs) {
3     cout << "请输入您要修改的联系人: " << endl;
4     string name;

```

```
5     cin >> name;
6     int ret = isExit(abs, name);
7     if (ret != -1) {
8         //姓名
9         string name;
10        cout << "请输入姓名: " << endl;
11        cin >> name;
12        abs->personArray[ret].m_Name = name;
13        //性别
14        int sex = 0;
15        cout << "请输入性别: " << endl;
16        cout << "1 --- 男" << endl;
17        cout << "2 ---女" << endl;
18        while (true) {
19            cin >> sex;
20            if (sex == 1 || sex == 2) {
21                abs->personArray[ret].m_Sex = sex;
22                break;
23            }
24            cout << "输入有误, 请重新输入" << endl;
25        }
26        //年龄
27        int age = 0;
28        cout << "请输入年龄: " << endl;
29        cin >> age;
30        abs->personArray[ret].m_Age = age;
31        //电话
32        string phone;
33        cout << "请输入联系电话: " << endl;
34        cin >> phone;
35        abs->personArray[ret].m_Phone = phone;
36        //地址
37        string address;
38        cout << "请输入家庭住址: " << endl;
```

```

39         cin >> address;
40         abs->personArray[ret].m_Addr = address;
41     }
42     else {
43         cout << "查无此人\n" << endl;
44     }
45     system("pause");
46     system("cls");
47 }

```

9.2 测试修改联系人功能

在switch case语句中，case 4中添加：

```

1 modifyPerson(&abs);

```

10. 清空联系人

功能描述：清空通讯录中所有信息

清空联系人实现步骤

- 封装清空联系人函数
- 测试清空联系人

10.1 封装清空联系人函数

实现思路：将通讯录所有联系人信息清除掉，只要将通讯录记录的联系人数量置为0，做逻辑清空即可。清空联系人代码：

```

1 //6. 清空联系人函数
2 void cleanPerson(Addressbooks* abs) {
3     abs->m_Size = 0;
4     // 将当前记录的联系人数量置为0，做逻辑清空操作
5     cout << "通讯录已清空\n" << endl;
6     system("pause");
7     system("cls");
8 }

```

11. 代码

```
1  #define _CRT_SECURE_NO_WARNINGS 1
2  #include<iostream>
3  #include<string>
4  #define MAX 1000
5  using namespace std;
6
7  //联系人结构体
8  struct Person {
9      string m_Name; //性别
10     int m_Sex; //性别
11     int m_Age; //年龄
12     string m_Phone; //电话
13     string m_Addr; //住址
14 };
15
16 //通讯录结构体
17 struct Addressbooks {
18     //通讯录中保存联系人数组
19     struct Person personArray[MAX];
20     //通讯录当前记录的人员个数
21     int m_Size;
22 };
23
24 //菜单界面显示
25 void showMenu() {
26     cout << "*****" << endl;
27     cout << "***** 1. 添加联系人 *****" << endl;
28     cout << "***** 2. 显示联系人 *****" << endl;
29     cout << "***** 3. 删除联系人 *****" << endl;
30     cout << "***** 4. 查找联系人 *****" << endl;
31     cout << "***** 5. 修改联系人 *****" << endl;
32     cout << "***** 6. 清空联系人 *****" << endl;
33     cout << "***** 0. 退出通讯录 *****" << endl;
```

```
34     cout << "*****" << endl;
35 }
36
37 //1. 添加联系人
38 void addPerson(Addressbooks* abs) {
39     //判断通讯录是否已满, 如果满了就不再添加
40     if (abs->m_Size == MAX) {
41         cout << "通讯录已满, 无法添加! " << endl;
42     }
43     else {
44         //添加具体联系人
45         //姓名
46         string name;
47         cout << "请输入姓名: " << endl;
48         cin >> name;
49         abs->personArray[abs->m_Size].m_Name = name;
50         //性别
51         cout << "请输入性别: " << endl;
52         cout << "1 --- 男" << endl;
53         cout << "2 --- 女" << endl;
54         int sex = 0;
55         while (true) {
56             cin >> sex;
57             if (sex == 1 || sex == 2) {
58                 abs->personArray[abs->m_Size].m_Sex = sex;
59                 break;
60             }
61             cout << "输入有误, 请重新输入! " << endl;
62         }
63         //年龄
64         cout << "请输入年龄: " << endl;
65         int age = 0;
66         cin >> age;
67         abs->personArray[abs->m_Size].m_Age = age;
```

```

68         //电话
69         cout << "请输入联系电话: " << endl;
70         string phone;
71         cin >> phone;
72         abs->personArray[abs->m_Size].m_Phone = phone;
73         //住址
74         cout << "请输入住址: " << endl;
75         string address;
76         cin >> address;
77         abs->personArray[abs->m_Size].m_Addr = address;
78         //更新通讯录人数
79         abs->m_Size++;
80         cout << "添加联系人成功\n" << endl;
81
82         system("pause");
83         system("cls");//清屏
84     }
85 }
86
87 //2. 显示联系人
88 void showPerson(Addressbooks* abs) {
89     //判断通讯录中人数是否为空, 如果为0, 提示记录为空
90     //如果不为0, 显示记录的联系人信息
91     if (abs->m_Size == 0) {
92         cout << "当前记录为空" << endl;
93     }
94     else {
95         for (int i = 0; i < abs->m_Size; i++) {
96             cout << "姓名: " << abs->personArray[i].m_Name << "\t";
97             cout << "性别: " << (abs->personArray[i].m_Sex == 1 ? "男" :
"女") << "\t";
98             cout << "年龄: " << abs->personArray[i].m_Age << "\t";
99             cout << "联系方式: " << abs->personArray[i].m_Phone << "\t";
100            cout << "家庭住址: " << abs->personArray[i].m_Addr << endl;

```



```

101     }
102 }
103
104     system("pause");
105     system("cls");
106 }
107
108 //3. 删除联系人
109 //检测联系人是否存在, 如果存在, 返回联系人所在数组中的具体位置, 不存在返回-1
110 int isExit(Addressbooks* abs, string name) {
111     for (int i = 0; i < abs->m_Size; i++) {
112         if (abs->personArray[i].m_Name == name) {
113             return i; //找到, 返回下标编号
114         }
115     }
116     return -1; //遍历结束未找到, 返回-1
117 }
118 //删除联系人
119 void deletePerson(Addressbooks* abs) {
120     cout << "请输入您要删除的联系人: " << endl;
121     string name;
122     cin >> name;
123     int ret = isExit(abs, name);
124     //ret != -1 查到了
125     //ret == -1 未查到
126     if (ret != -1) {
127         //数据前移
128         for (int i = ret; i < abs->m_Size; i++) {
129             abs->personArray[i] = abs->personArray[i + 1];
130         }
131         //更新通讯录中的人员数
132         abs->m_Size--;
133         cout << "删除成功\n";
134     }

```

```
135     else {
136         cout << "查无此人\n";
137     }
138     system("pause");
139     system("cls");
140 }
141
142 //4. 查找指定联系人信息
143 void findPerson(Addressbooks* abs) {
144     cout << "请输入您要查找的联系人: " << endl;
145     string name;
146     cin >> name;
147     //判断指定的联系人是否存在通讯录中
148     int ret = isExit(abs,name);
149     if (ret != -1) { //找到
150         cout << "姓名: " << abs->personArray[ret].m_Name << "\t";
151         cout << "性别: " << (abs->personArray[ret].m_Sex == 1 ? "男" :
152         "女") << "\t";
153         cout << "年龄: " << abs->personArray[ret].m_Age << "\t";
154         cout << "联系方式: " << abs->personArray[ret].m_Phone << "\t";
155         cout << "家庭住址: " << abs->personArray[ret].m_Addr << endl;
156     }
157     else { //未找到
158         cout << "查无此人\n" << endl;
159     }
160     system("pause");
161     system("cls");
162 }
163 //5. 修改指定联系人信息
164 void modifyPerson(Addressbooks* abs) {
165     cout << "请输入您要修改的联系人: " << endl;
166     string name;
167     cin >> name;
```

```
168     int ret = isExit(abs, name);
169     if (ret != -1) {
170         //姓名
171         string name;
172         cout << "请输入姓名: " << endl;
173         cin >> name;
174         abs->personArray[ret].m_Name = name;
175         //性别
176         int sex = 0;
177         cout << "请输入性别: " << endl;
178         cout << "1 --- 男" << endl;
179         cout << "2 ---女" << endl;
180         while (true) {
181             cin >> sex;
182             if (sex == 1 || sex == 2) {
183                 abs->personArray[ret].m_Sex = sex;
184                 break;
185             }
186             cout << "输入有误, 请重新输入" << endl;
187         }
188         //年龄
189         int age = 0;
190         cout << "请输入年龄: " << endl;
191         cin >> age;
192         abs->personArray[ret].m_Age = age;
193         //电话
194         string phone;
195         cout << "请输入联系电话: " << endl;
196         cin >> phone;
197         abs->personArray[ret].m_Phone = phone;
198         //地址
199         string address;
200         cout << "请输入家庭住址: " << endl;
201         cin >> address;
```

```

202         abs->personArray[ret].m_Addr = address;
203     }
204     else {
205         cout << "查无此人\n" << endl;
206     }
207     system("pause");
208     system("cls");
209 }
210
211 //6. 清空联系人函数
212 void cleanPerson(Addressbooks* abs) {
213     abs->m_Size = 0;
214     //将当前记录的联系人数量置为0, 做逻辑清空操作
215     cout << "通讯录已清空\n" << endl;
216     system("pause");
217     system("cls");
218 }
219
220 int main() {
221     //创建通讯录结构体变量
222     Addressbooks abs;
223     //初始化通讯录中当前人员个数
224     abs.m_Size = 0;
225     //用户选择输入
226     int select = 0;
227     while (true) { //重复选择选项
228         //菜单的调用
229         showMenu();
230         cin >> select;
231         switch (select) {
232             case 1: //添加联系人
233                 addPerson(&abs); //利用地址传递, 可以修饰实参
234                 break;
235             case 2: //显示联系人

```

```

236         showPerson(&abs);
237         break;
238     case 3: // 删除联系人
239         deletePerson(&abs);
240         break;
241     case 4: // 查找联系人
242         findPerson(&abs);
243         break;
244     case 5: // 修改联系人
245         modifyPerson(&abs);
246         break;
247     case 6: // 清空联系人
248         cleanPerson(&abs);
249         break;
250     case 0: // 退出通讯录
251         cout << "欢迎下次使用\n";
252         system("pause"); // 请按任意键继续
253         return 0;
254         break;
255     default:
256         break;
257 }
258 }
259 return 0;
260 }

```

12. .exe文件

Visual Studio生成exe可执行文件

Visual Studio是我们常用的集成开发环境。每当我们运行我们编写的代码的时候，我们都是先在Visual Studio中打开对应的项目软件，点击 **调试** 来运行。但是当我们开发一个软件时，我们并不能让用户做同样的操作，往往是提供一个可执行文件以供执行。

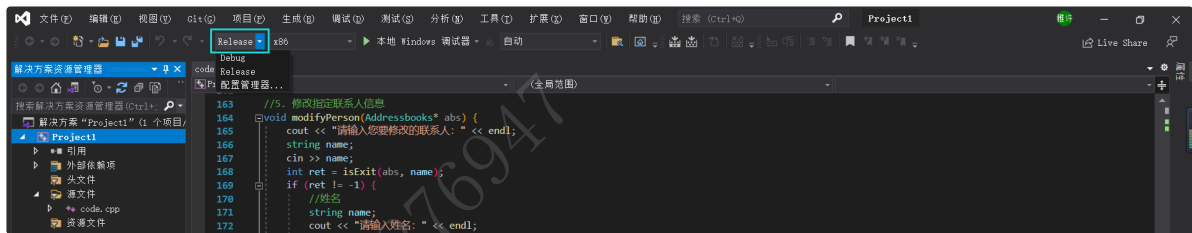
将已经调试完毕的代码项目生成.exe

实施步骤：

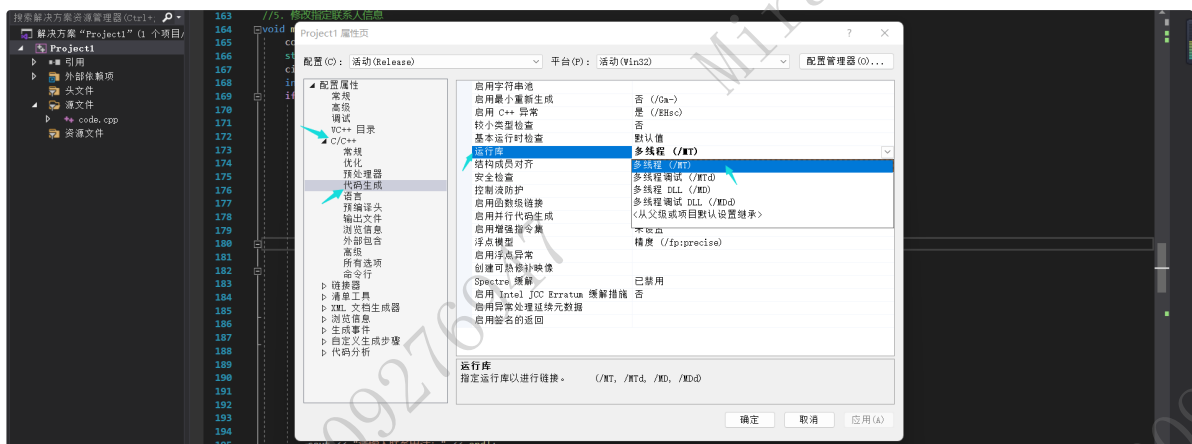
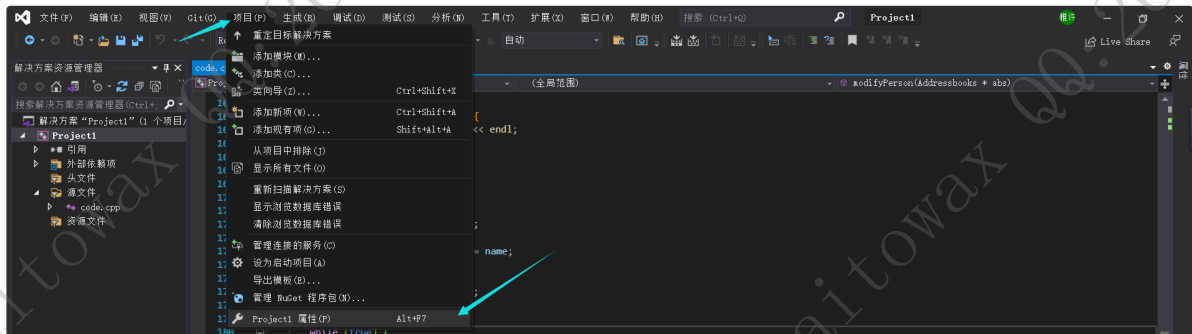
- 设置解决方案配置为Release
- 修改项目运行库为多线程(/MT)
- 重新生成解决方案

- 修改生成的可执行文件的名字

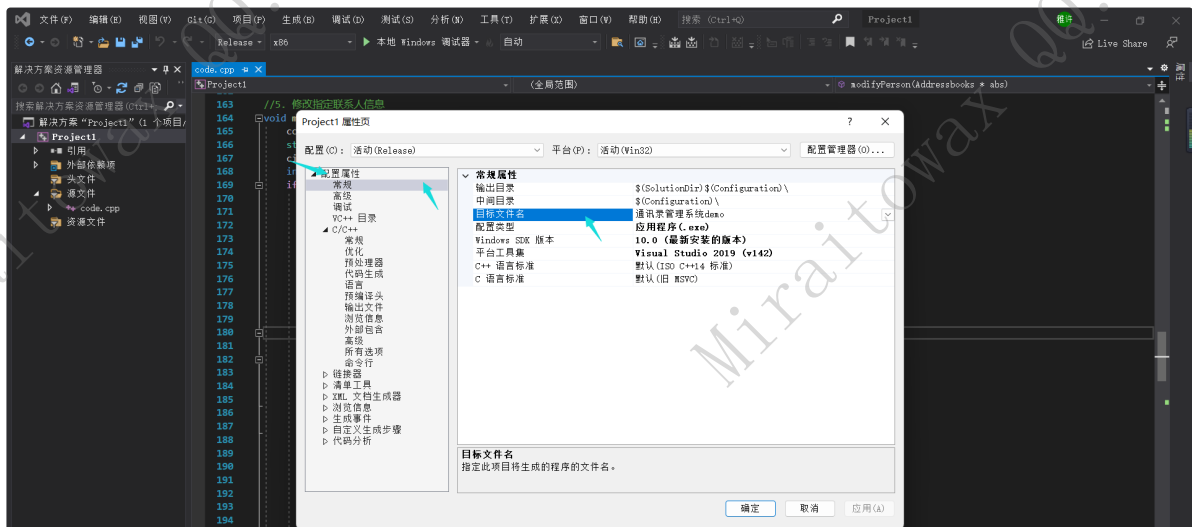
12.1 设置解决方案配置为Release



12.2 修改项目运行库为多线程 (/MT)

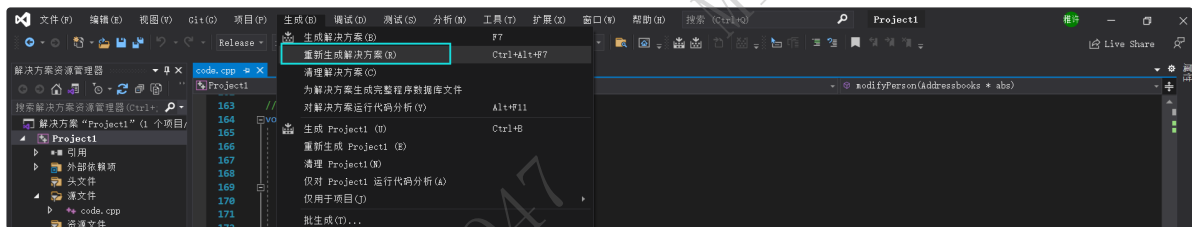


12.2.1 修改生成的可执行文件的名字



默认为\$(ProjectName)。如图修改为 通讯录管理系统demo，修改之后重新生成解决方案即可

12.3 重新生成解决方案



之后就会在项目文件目录中多出 **Release** 文件夹，里面的exe文件就是可执行文件

13. 相关报错

- 由于找不到MSVCP140.dll，无法继续执行代码。重新安装程序可能会解决此问题

在Visual Studio中：调试模式 (Debug) 选择Multi-threaded Debug (/MTd)，发布模式选择 (Release) Multi-threaded DLL (/MD)。

/MT使用多线程静态编译，/MD使用多线程和使用DLL文件，/MTd使用多线程静态编译并生成调试信息，/MDd使用多线程和DLL文件并生成调试信息。

VS在生成调试模式下的二进制文件时，就会把运行时库的一些代码静态编入，同时这也是为什么debug版本的二进制文件往往非常大的原因。