**华中科技大学计算机科学与技术学院**

**《C语言与程序设计》课程设计**

**实验报告**

**题目： 物流信息管理系统**

**专业： 物联网工程**

**班级： 物联网1601**

**学号： U201614890**

**姓名： 徐光磊**

**成绩：**

**指导教师： 刘芳**

**完成日期： 2017年 9月 1日**

**物流信息管理系统**

1. **系统功能结构模块图**
2. **数据结构设计及其用法说明**
3. **货物信息结构体：**

/\*\*

\* 货物的基本信息：\*Type : 装货、卸货的货物的种类

\* \*Volume : 货物所占容量，（单位立方米）

\*/

typedef struct Good {

char uploadType[10];

float upVolume;

char downloadType[10];

float downVolume;

} good;

1. **车辆信息结构体**

/\*\*

\* 配送车辆的基本信息： carID： 车辆牌照

\* routeID：执行配送路线编号

\* driverName： 司机姓名

\* driverTel： 司机移动电话

\* carID: 车辆编号

\* goodList： 运货清单

\*/

typedef struct Car {

char carID[9];

char routeID[7];

char driverName[9];

char driverTel[12];

good \* good;

struct Car \* next;

} car;

1. **站点信息结构体**

/\*\*

\* 经停站点详细信息: routeNums: 固定配送路线数量

\* routeIDArray: 固定配送路线编号的数组

\* siteSID: 站点序号

\* siteID: 站点编号

\* siteNames: 站点名称

\* d2Start: 与初始站点距离

\* d2Last: 与上一个站点距离

\* t2Last: 与上一个站点交通耗时

\* tWait: 停留耗时

\* next: 指向下一个站点的指针

\*/

typedef struct Site {

char routeID[7];

int siteSID;

char siteID[11];

char siteName[51];

float d2Start;

float d2Last;

float time2Last;

float waitTime;

char routeIDArray[50];

struct Car \* carHeadP;

struct Site \* next;

} site;

1. **路线信息结构体**

/\*\*

\* 配送路线详细信息: routeID: 固定配送路线编号

\* name: 固定配送路线编号名称

\* siteNums: 固定配送路线总站点数

\* miles: 固定配送路线总公里数

\* period: 全站点配送总耗时

\* startSite: 起始站点编号

\* endSite: 终止站点编号

\* adminName: 负责人姓名

\* tel: 负责人办公室电话

\* mTel: 负责人移动电话

\* email: 负责人电子邮箱

\* firstSite: 指向站点（即第一个站点）的指针

\* next: 指向下一个路线的指针

\*/

typedef struct Route {

char routeID[7];

char name[21];

short siteNums;

float miles;

float period;

char startSite[11];

char endSite[11];

char adminName[9];

char tel[9];

char mTel[12];

char email[51];

struct Site \* firstSite;//指向该路线的第一个站点，并形成路线链表

struct Route \* next;

} route;

1. **使用方法**

以上的结构体均可直接通过malloc函数开辟内存空间后直接赋值使用，创建、插入、修改、删除、释放空间等操作均可直接调用ListTool.h文件中的链表相关函数来进行。

1. **数据类型及范例**
2. 路线

|  |  |  |
| --- | --- | --- |
| **中文字段名** | **类型及长度** | **举例** |
| 固定配送路线编号 | char[6] |  |
| 固定配送路线名称 | char[20] | 如：关山线 |
| 固定配送路线总站点数 | short |  |
| 固定配送路线总公里数 | float |  |
| 全站点配送总耗时 | float | 正常交通情况下的总耗时（包括站点停留耗时），分钟 |
| 起始站点编号 | char[10] |  |
| 终止站点编号 | char[10] |  |
| 负责人姓名 | char[8] |  |
| 负责人办公室电话 | char[8] |  |
| 负责人移动电话 | char[11] |  |
| 负责人电子邮箱 | char[50] |  |

1. 站点

|  |  |  |
| --- | --- | --- |
| **中文字段名** | **类型及长度** | **举例** |
| 固定配送路线编号 | char[6] |  |
| 站点序号 | int |  |
| 站点编号 | char[10] |  |
| 站点名称 | char[50] |  |
| 与起始站点距离 | float | 公里 |
| 与上一个站点距离 | float | 公里 |
| 与上一个站点交通耗时 | float | 分钟 |
| 停留耗时 | float | 分钟 |
| 经过本站点固定路线编号，如为多条固定路线的交汇站点，需要给出多条固定路线编号 |  |  |

1. 车辆

|  |  |  |
| --- | --- | --- |
| **中文字段名** | **类型及长度** | **举例** |
| 车辆牌照 | char[8] |  |
| 执行配送路线编号 | char[6] |  |
| 司机姓名 | char[8] |  |
| 司机移动电话 | char[11] |  |
| 运货清单 | struct | 自行设计，至少需要有车辆在本站点载货/卸货的货物信息，特别是容量信息 |

|  |  |  |
| --- | --- | --- |
| **中文字段名** | **类型及长度** | **举例** |
| 载货种类 | char[8] |  |
| 载货容量 | char[6] |  |
| 卸货种类 | char[8] |  |
| 卸货容量 | char[11] |  |

1. 货物
2. 十字链表结构

货物基本信息

货物基本信息

货物基本信息

head

路线1

路线2

路线n

NULL

**…**

站点1详细信息

站点2详细信息

站点n基本信息

经停站点1的车辆基本信息链

经停站点2的车辆基本信息链

经停站点n的车辆基本信息链

**…**

。

1. **程序结构**
2. **各模块的功能**
3. **测试**
4. **体会**

在本次的课程设计中，让我感觉最遗憾的就是没有用c语言去实现一个更佳的GUI界面，而只是在控制台界面下进行粗略的优化控制台界面（即修改了标题颜色、对齐方式和数字菜单）。对于一个面向用户的软件，没有真正图形界面是一个大大的硬伤，影响用户体验，更影响这款应用的传播。我在规划的阶段，便开始了解c语言对GUI的支持。发现真正用C语言写GUI程序的很少很老，像Gtk对win32 api的包装思路就落伍于当前GUI开发的发展。类似Qt等包装较完善的库又需要进而掌握C++的各种特性，学习曲线之陡之望而生畏。

本次课程设计中，我并没有像给的模板一样去严格地分明几个模块的作用，这样的操作对于控制台程序来说，操作过于繁琐。频繁的输入确定并不如键鼠的点击操作，复杂界面下的表现明显不够人性化。与之相反，我在打开程序的一开始便在界面中展示出所有的路线信息，然后直接提供再进一步的功能选项。这样的选择某种程度上也是在文本菜单下的一种妥协。

作为一个信息管理系统，明显的特征就是信息量大，处理信息多，所以我在系统中为可以自动补全的数据都进行了自动添加或更新的功能（如站点的序号，离终点站的距离、站点数等等数据），希望能以此提高用户的录入效率与使用上的友好。

本程序中我另一个面对的问题就是如何向用户直观地体现所有的信息，C语言并没有和Excel等

1. **致谢**
2. **参考文献**

**附录：源代码清单**