数据结构课程设计报告

——LZW压缩算法及其应用

学号：201800301238

姓名：陈振国

班级：软件工程18级5班

指导教师：任国珍

目录

1. **任务概述**

1.1总体任务要求

1. **需求分析**

2.1总体需求概述

2.2功能需求

2.3非功能需求

1. **概要设计**

3.1总体架构设计

3.2关键流程设计

1. **详细设计**

4.1界面详细设计

4.2数据结构详细设计

1. **系统实现**
2. **总结**
3. **任务概述**

**1.1总体任务要求**

在一个文本文件上实现LZW压缩和解压缩，其中每个字符就是该文本的8位ASCII码。在实现LZW过程中需要仔细考虑如何在编译表中找到匹配或找不到匹配，需要注意匹配算法的时间、空间开销。应用LZW算法实现256色灰度BMP图像文件的压缩和解压缩。

1. **需求分析**

**2.1总体需求概述**

能够实现仅包含ascii字符文本文档和使用256色colortable的bmp文件的压缩和解压。

**2.2功能需求**

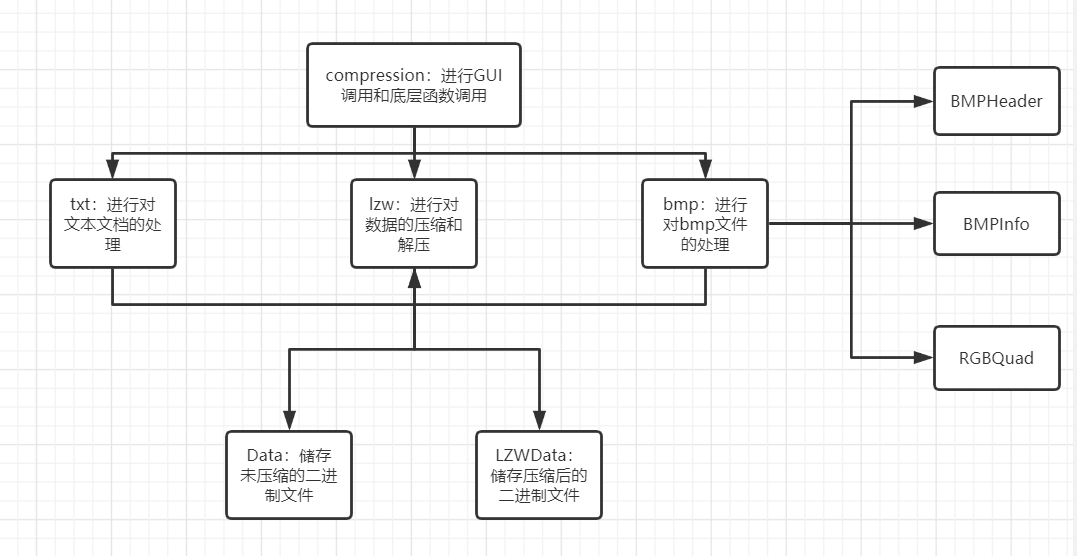
**(1)文本压缩**

选择文本文件,要求该文本文件中字符全部为ascii码,

1. **概要设计**

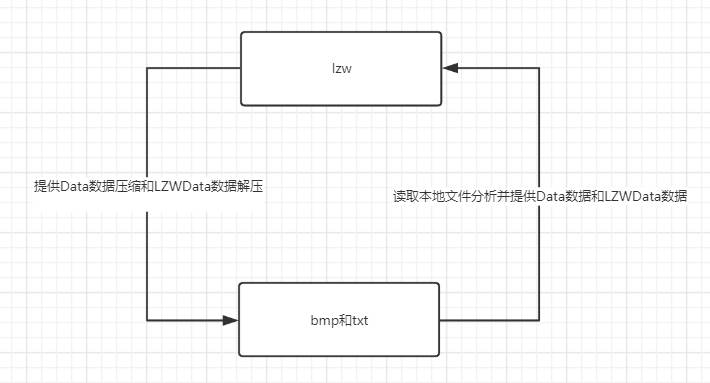
**3.1总体架构设计**

类之间的关系大致如下：

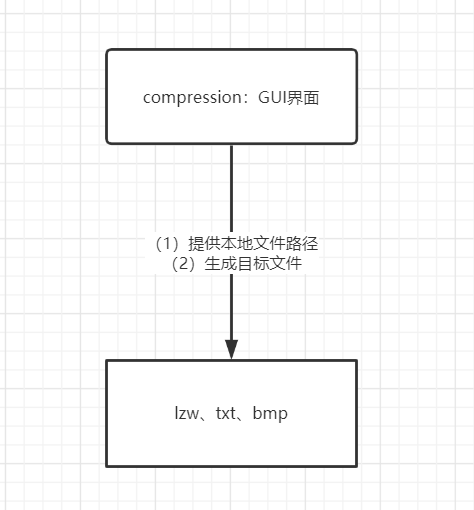


**3.2具体类关系设计**

（1）lzw类和bmp、txt的关系

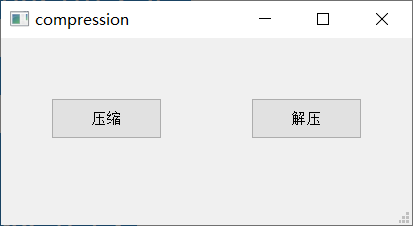


（2）compression类和底层类关系

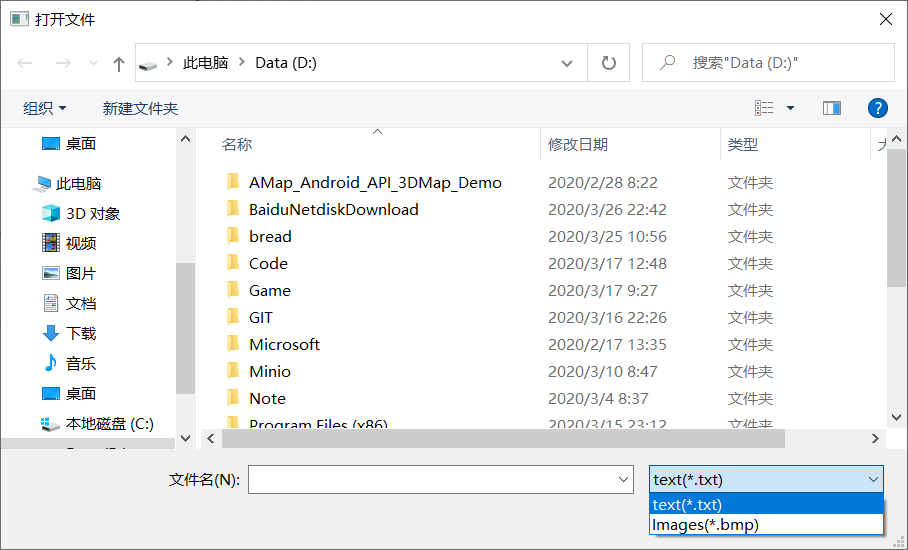


1. **详细设计**

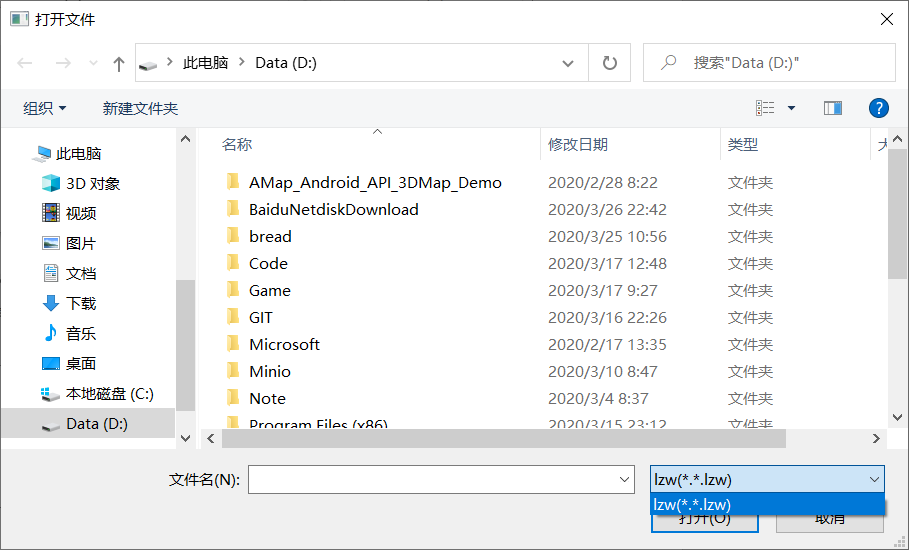
**4.1界面详情**



界面有两个按钮，分别进行压缩和解压，点击后弹出文件选择界面：



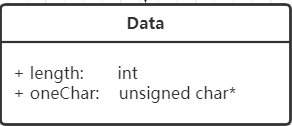
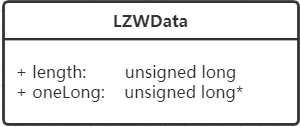
压缩界面可以选择txt或者bmp文件进行压缩



解压界面可以选择lzw文件进行解压

**4.2数据结构设计**

（1）数据结构在程序中储存方式为Data和LZWData数据结构

（2）数据在文件中存储

若是非压缩的txt和bmp文件，以其本身的形式储存，即根据Data结构中的oneChar字节组输出或读取

若是压缩的lzw文件，其oneLong的前四个字节（即第一个unsigned long）为压缩字节的长度，方便读取。

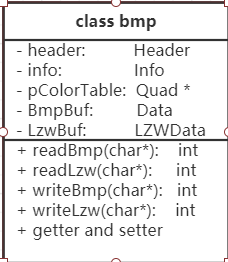
**4.3文件命名**

压缩的\*.bmp命名为\*.bmp.lzw；压缩的\*.txt命名为\*.txt.lzw

1. **代码清单**

**5.1文件处理类**

（1）bmp文件处理



代码：

//bmp.cpp

#include <cstdio>

#include "bmp.h"

/\*\*

\* 读函数

\*/

int bmp::**readBmp**(char \*bmpName) {

//打开文件

FILE \*fp = fopen(bmpName, "rb");

if(fp == nullptr)

return 0;

//读取文件头

fread(&header, sizeof(Header), 1, fp);

//读取信息头

fread(&info, sizeof(Info), 1, fp);

//计算图片行数和数据长度

int lineByte = (info.biWidth \* info.biBitCount / 8 + 3) / 4 \* 4;

this->BmpBuf.length = lineByte \* info.biHeight;

//获取颜色板

pColorTable = new Quad[256];

fread(pColorTable, sizeof(RGBQuad), 256, fp);

//获取数据

this->BmpBuf.oneChar = new unsigned char[this->BmpBuf.length];

fread(this->BmpBuf.oneChar, 1, this->BmpBuf.length, fp);

//关闭流

fclose(fp);

}

int bmp::**readLzw**(char \*lzwName) {

//打开文件

FILE \*fp = fopen(lzwName, "rb");

if(fp == nullptr)

return 0;

//读取文件头

fread(&header, sizeof(Header), 1, fp);

//读取信息头

fread(&info, sizeof(Info), 1, fp);

//计算图片行数和数据长度

// int lineByte = (info.biWidth \* info.biBitCount / 8 + 3) / 4 \* 4;

// this->DataLength = lineByte \* info.biHeight;

//获取颜色板

pColorTable = new Quad[256];

fread(pColorTable, sizeof(RGBQuad), 256, fp);

//获取数据

unsigned long length;

fread(&length, sizeof(unsigned long), 1, fp);

this->LzwBuf.length = length + 1;

this->LzwBuf.oneLong = new unsigned long[length + 1];

fread(this->LzwBuf.oneLong+1, sizeof(unsigned long), length, fp);

this->LzwBuf.oneLong[0] = length;

//关闭流

fclose(fp);

}

Header bmp::**getHeader**() {

return this->header;

}

Quad\* bmp::**getColorTable**() {

return this->pColorTable;

}

Info bmp::**getInfo**() {

return this->info;

}

Data bmp::**getData**() {

return this->BmpBuf;

}

LZWData bmp::**getLzw**() {

return this->LzwBuf;

}

/\*\*

\* 写函数

\*/

int bmp::**writeBmp**(char \*bmpName) {

//打开文件

FILE \*fp = fopen(bmpName, "wb");

if(fp == nullptr)

return 0;

//输出文件头

fwrite(&header, sizeof(Header), 1, fp);

//输出信息头

fwrite(&info, sizeof(Info), 1, fp);

//写颜色板

fwrite(this->pColorTable, sizeof(RGBQuad), 256, fp);

//获取数据

fwrite(this->BmpBuf.oneChar, 1, this->BmpBuf.length, fp);

//关闭流

fclose(fp);

}

int bmp::**writeLzw**(char \*lzwName) {

//打开文件

FILE \*fp = fopen(lzwName, "wb");

if(fp == nullptr)

return 0;

//读取文件头

fwrite(&header, sizeof(Header), 1, fp);

//读取信息头

fwrite(&info, sizeof(Info), 1, fp);

//获取颜色板

fwrite(pColorTable, sizeof(RGBQuad), 256, fp);

//获取数据

fwrite(this->LzwBuf.oneLong, sizeof(unsigned long), this->LzwBuf.length, fp);

//关闭流

fclose(fp);

}

void bmp::**setHeader**(Header inHeader) {

this->header = inHeader;

}

void bmp::**setInfo**(Info inInfo) {

this->info = inInfo;

}

void bmp::**setColorTable**(Quad \*inQuad) {

this->pColorTable = inQuad;

}

void bmp::**setData**(Data inData) {

this->BmpBuf = inData;

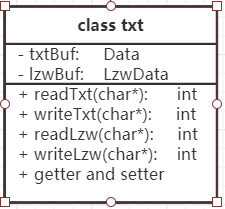
}

void bmp::**setLzw**(LZWData inLzw) {

this->LzwBuf = inLzw;

}

（2）txt文件处理



代码：

//txt.cpp

#include "txt.h"

int txt::**readTxt**(char \*txtName) {

//打开文件

FILE \*fp = fopen(txtName, "rb");

if(fp == nullptr)

return 0;

fseek(fp,0L,SEEK\_END); /\* 定位到文件末尾 \*/

this->txtBuf.length = ftell(fp);

fseek(fp,0L,SEEK\_SET); /\* 定位到文件开头 \*/

this->txtBuf.oneChar = new unsigned char[this->txtBuf.length];

fread(this->txtBuf.oneChar, 1, this->txtBuf.length, fp);

fclose(fp);

}

int txt::**writeTxt**(char \*txtName) {

//打开文件

FILE \*fp = fopen(txtName, "wb");

if(fp == nullptr)

return 0;

fwrite(this->txtBuf.oneChar, 1, this->txtBuf.length, fp);

fclose(fp);

}

int txt::**readLzw**(char \*lzwName) {

//打开文件

FILE \*fp = fopen(lzwName, "rb");

if(fp == nullptr)

return 0;

fseek(fp,0L,SEEK\_END); /\* 定位到文件末尾 \*/

this->lzwBuf.length = ftell(fp)/4;

fseek(fp,0L,SEEK\_SET); /\* 定位到文件开头 \*/

this->lzwBuf.oneLong = new unsigned long[this->txtBuf.length];

fread(this->lzwBuf.oneLong, sizeof(unsigned long), this->lzwBuf.length, fp);

fclose(fp);

}

int txt::**writeLzw**(char \*lzwName) {

//打开文件

FILE \*fp = fopen(lzwName, "wb");

if(fp == nullptr)

return 0;

fwrite(this->lzwBuf.oneLong, sizeof(unsigned long), this->lzwBuf.length, fp);

fclose(fp);

}

Data txt::**getTxtBuf**() {

return this->txtBuf;

}

LZWData txt::**getLzwBuf**() {

return this->lzwBuf;

}

void txt::**setTxt**(Data inData) {

this->txtBuf = inData;

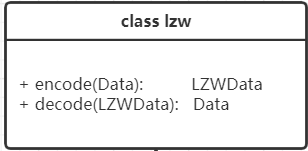
}

void txt::**setLzw**(LZWData inLzw) {

this->lzwBuf = inLzw;

}

**5.2数据压缩类**



代码如下：

//lzw.cpp

#include <iostream>

#include <fstream>

#include "lzw.h"

LZWData lzw::**encode**(Data inData) {

vector<int> V;

V.push\_back(0);

string P;

string C;

string P\_C;

std::map<string,int> dic;

int len = 256;

std::map<string, int>::iterator iter;

for(int i = 0 ; i < 256 ; i++) {

string tmp{(char)i};

dic.insert(pair<string,int>(*tmp*,*i*));

}

for(int i = 0 ; i < inData.length ; i++) {

C = inData.oneChar[i];

P\_C =P + C;

iter = dic.find(P\_C);

if(iter != dic.end()) {

P = P\_C;

}else {

V.push\_back(dic[P]);

dic.insert(pair<string,int>(*P\_C*,len++));

P = C;

}

}

if(P.length() > 0) {

V.push\_back(dic[P]);

}

V[0] = V.size() - 1;

LZWData ans;

ans.length = V.size();

ans.oneLong = new unsigned long[ans.length];

for(int i = 0;i<V.size();i++){

ans.oneLong[i] = V[i];

}

return ans;

}

Data lzw::**decode**(LZWData inData) {

vector<int> V;

for(int i = 0; i < inData.length; i++) {

V.push\_back(inData.oneLong[i]);

}

int pW,cW;

string str;

string P;

string C;

string P\_C;

std::map<int,string> dic;

int len = 256;

std::map<int,string>::iterator iter;

for(int i = 0 ; i < 256 ; i++) {

string tmp{(char)i};

dic.insert(pair<int,string>(*i*,*tmp*));

}

cW = V[1];

str = dic[cW];

for(int i = 2 ; i < V.size() ; i++) {

pW = cW;

cW = V[i];

iter = dic.find(cW);

if(iter != dic.end()) {

str.append(dic[cW]);

P = dic[pW];

C = dic[cW][0];

dic.insert(pair<int,string>(len++,P + C));

}else {

P = dic[pW];

C = dic[pW][0];

dic.insert(pair<int,string>(len++,P + C));

str.append(P + C);

}

}

Data ans;

ans.length = str.length();

ans.oneChar = new unsigned char[ans.length];

for(int i = 0 ; i < str.length() ; i++) {

ans.oneChar[i] = str.at(i);

}

return ans;

}

1. **总结**

**此次课程设计感觉难点在于bmp文件中画板数据的提取和压缩之后的写入，以及文件和程序中的储存使用形式不同引发的数据结构设计问题。理解并熟悉bmp文件的形式花费了比较多的时间，在编写完bmp文件的类后又发现了其和lzw压缩以及txt文件的类的接口对接不上的问题，一开始各种修改参数对接接口导致程序逻辑混乱，索性重写。程序的设计必须一开始进行整体结构、模块以及模块之间接口的规划，避免再次发生这种情况。**