同濟大學
TONGJI UNIVERSITY

高级语言程序设计大作业

装

订

线

目录

装 订 线

1.设计思路与功能描述

1.1 设计思路：

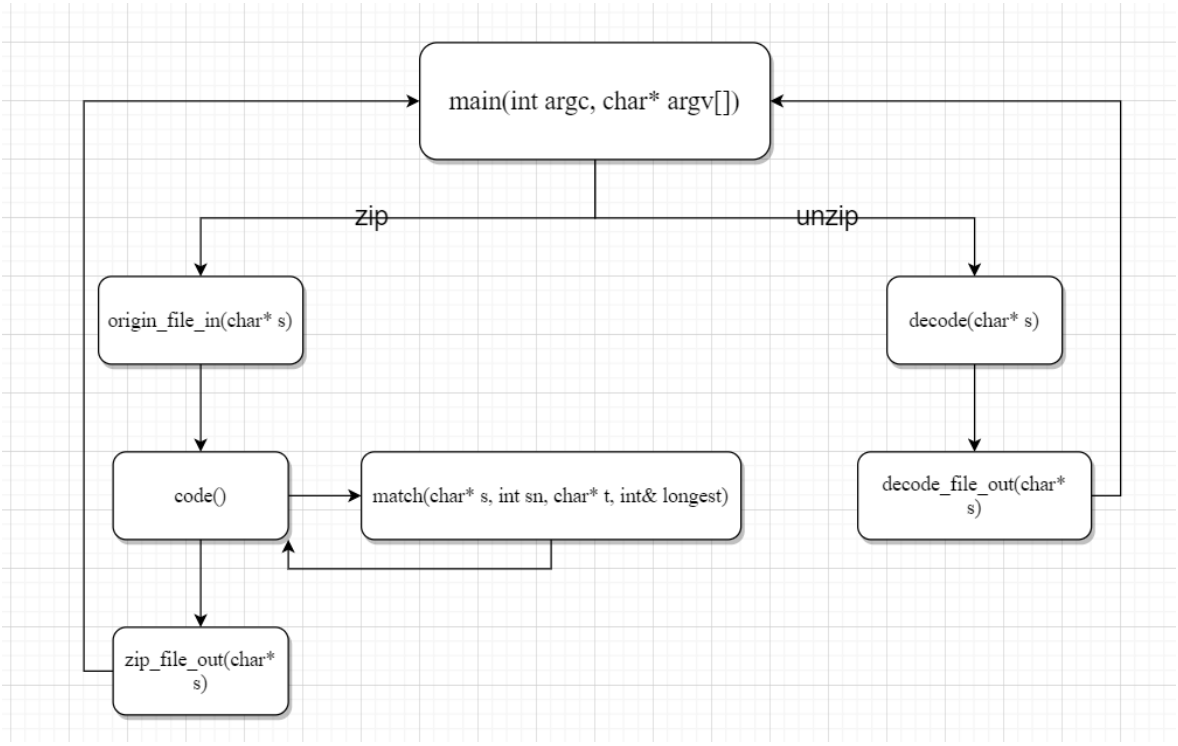
本次大作业核心在于寻找合适的压缩算法，实现文件压缩。

文件压缩有各类算法，例如 Huffman 编码、字典算法、LZ 系算法，以及各类有损压缩等。相较于有损压缩，无损压缩通用性更好，且有损压缩往往在图片、音频、视频领域使用。针对本次大作业的日志文件，选择采用无损压缩的 LZ77 算法。

LZ77 算法是采用字典做数据压缩的算法，其通用性好，因为它无需像 Huffman 编码那样基于统计的先验知识。其核心思想是利用数据的重复结构信息来进行数据压缩，使用关键是对滑动窗口和前向缓冲区的操作，找到滑动窗口和前向缓冲区中匹配的最长字符串，利用偏移量、长度和下一位字符来实现编码，从而描述文件中的重复部分信息。

大作业函数：main(int argc, char* argv[])、origin_file_in(char* s)、code()、match(char* s, int sn, char* t, int& longest)、zip_file_out(char* s)、decode(char* s)、decode_file_out(char* s)，共 7 个。

函数执行顺序为：



1.2 功能描述：

1）main(int argc, char* argv[])--主函数：

程序开始，选择压缩或解压，调用其他函数，记录压缩或解压时间。

2）origin_file_in(char* s)--原始文件读取函数：

从原始文件读取字符串。

3）code()--编码函数：

使用 LZ77 算法编码字符串，让滑动窗口向后移动，调用 match(char* s, int sn, char* t, int& longest)函数，查找前向缓冲区与滑动窗口匹配的最长字符串，进行编码。

4）match(char* s, int sn, char* t, int& longest)--匹配函数：

被 code()调用，查找前向缓冲区与滑动窗口匹配的最长字符串。从滑动窗口和前向缓冲区的首元素开始，找向匹配的最长字符串。

5）zip_file_out(char* s)--压缩文件输出函数：

将压缩的字符串输出到压缩文件

6）decode(char* s)--解码函数：

将压缩文件读取进来，按照 LZ77 的解码方式进行解码。

7）decode_file_out(char* s)--解压文件输出函数：

将解压好的字符串输出到解压文件

装 订 线

2.在实验过程中遇到的问题及解决方法

2.1 压缩算法选择：

如何选择合适的无损压缩算法？

通过阅读《Data Compression--The Complete Reference》以及网上查找资料，针对大作业的日志文件，暂定了 Huffman 编码、字典算法和 LZ77 算法。Huffman 编码虽然是通用的无损压缩算法，但其需要基于统计的先验知识、且并不符合日志文件的特点以及需要位操作等，被首先排除。针对字典算法和 LZ77 算法，进行了程序编写和测试，最终选择 LZ77 算法。

2.2 输入输出方式选择：

如何提高文件输入输出速度？

一般而言，采用 C 的输入输出方式比 C++的 cin、cout 要快、一次性输入后集中处理比一边输入一边处理要快、二进制文件输入比 ASCII 文件输入要快，但同时也要考虑程序处理的复杂度。结合大作业的要求，部分采用二进制文件的 read()进行输入，提高了文件输入速度。

2.3 平衡压缩时间和压缩比：

如何缩短压缩时间？如何提高压缩比？

LZ77 算法有滑动窗口长度和前向缓冲区长度两个参数，这两个参数在一定范围内变大可提高压缩比，但也会增长压缩时间。通过输出最大偏移量和最大匹配字符串长度，以及调整参数，最终选择滑动窗口长度为 4100，前向缓冲区长度为 650。

3.心得体会

3.1 程序和算法设计的体会：

1）从理论上讲，无损压缩可以压缩到文件的信息熵大小，但其成本往往过高。因此，对于各类不同的压缩算法，要针对要求，选择合适的压缩算法，同时也要注意通用性。

2）I/O 方式是运行速度的极大限制，但也关系到程序处理的难易，应当平衡运行速度和处理难度，选择合适的 I/O 方式。

3.2 编程过程的体会：

1）编程过程中要善于 Debug。程序中出现 bug 非常正常，通过断点、单步调试、变量监控等往往能快速锁定 bug。

2）实践出真知。通过亲自实验、反复尝试，就能找到相对较好的程序和算法设计。

3）搜索引擎是自学过程中的好老师。在遇到自己不了解的部分时，可以去 cppreference、微软 c++文档、csdn、stackoverflow 等地方寻找解决方法。

4. 源代码：

```
#define _CRT_SECURE_NO_WARNINGS
#include<stdio.h>
#include<stdlib.h>
#include<fstream>
#include<iostream>
#include<ctime>
#include <string>
#include<cstring>
using namespace std;
#define CLOCKS_PER_SEC ((clock_t)1000)
#define window_length 4100//滑动窗口长
#define cache 650//缓冲区长
#define max_length 10400000//原文件大小
struct CODE //lz77编码的结构体
{
    int off, len;
    char c;
};
string tocode;//用于压缩输出
CODE zip_file[max_length] = { 0,0,0 };
char ori_file[max_length] = { 0 };//读入的字符串
char decode_file[max_length] = { 0 };//解压后的字符串

void origin_file_in(char* s)  //从文件读入字符串
{
    ifstream fin(s, ios::binary);
    if (!fin)
    {
        cerr << "Can not open the input file!" << endl;
        exit(-1);
```

```
    }
    fin.read(ori_file, max_length);
    fin.close();
}

int match(char* s, int sn, char* t, int& longest)//字符串的匹配
{
    //找到最长的匹配字符串
    int i = 0, j = 0, match = 0, offset = 0;
    longest = 0;
    for (int k = 0; k < sn; k++)
    {
        i = k;
        j = 0;
        match = 0;
        while (i < sn && j < cache)//逐个匹配
        {
            if (s[i] != t[j])
                break;
            match++;
            i++;
            j++;
        }
        if (match > longest)
        {
            offset = k;
            longest = match;
        }
    }
    return offset;
}

void code()//编码
{
    int maxoff = 0;
    int maxlen = 0;
    int all_length = strlen(ori_file), window_end = 0, i = 1, offset = 0, len = 0;
    if (all_length > 0)//对字符串第一个字符进行编码输出
    {
        tocode += "0 0 ";
        tocode += ori_file[0];
        tocode += ' ';
```

```
    }
    for (; i < all_length; i++)
    {
        if (i < window_length)//更新窗口右边界
            window_end = i;
        else
            window_end = window_length;
        offset = match(ori_file + i - window_end, window_end, ori_file + i, len);//
匹配结果返回
        if (len == 0)
            tocode += '0';
        else
            tocode += to_string(window_end - offset);
        tocode += ' ';
        tocode += to_string(len);
        tocode += ' ';
        tocode += ori_file[i + len];
        tocode += ' ';
        i = i + len;
    }
}

void zip_file_out(char* s)  //将压缩的字符串输出到文件
{
    ofstream fout(s);
    if (!fout)
    {
        cerr << "Can not open the output file!" << endl;
        exit(-1);
    }
    fout << tocode;
    fout.close();
}

void decode(char* s)//解压
{
    ifstream fin(s);
    if (!fin)
    {
        cerr << "Can not open the input file!" << endl;
        exit(-1);
    }
```

```
    int co = 0;
    while (!fin.eof())
    {
        fin >> zip_file[co].off >> zip_file[co].len;
        fin.get();
        fin.get(zip_file[co].c);
        co++;
    }
    fin.close();
    int count = 0;
    int i = 0, j = 0;
    //解码
    for (; i < co; ++i)
    {
        if (zip_file[i].len == 0)//正常赋值
        {
            decode_file[count] = zip_file[i].c;
            ++count;
        }
        else//偏移赋值
        {
            for (j = 0; j < zip_file[i].len; ++j)
            {
                decode_file[count + j] = decode_file[count - zip_file[i].off + j];
            }
            decode_file[count + j] = zip_file[i].c;
            count = count + zip_file[i].len + 1;
        }
    }
}

void decode_file_out(char* s)//将解码输出到文件中
{
    ofstream fout(s);
    if (!fout)
    {
        cerr << "Can not open the output file!" << endl;
        exit(-1);
    }
    fout << decode_file;
    fout.close();
}
```

装 订 线

```
int main(int argc, char* argv[])
{
    cout << "Zipper 0.001! Author: root" << endl;
    cout << "可选择压缩或解压" << endl;
    if (argc != 4) {
        cerr << "Please make sure the number of parameters is correct." << endl;
        return -1;
    }

    if (strcmp(argv[3], "zip")&& strcmp(argv[3], "unzip")) {
        cerr << "Unknown parameter!\nCommand list:\nzip\nunzip" << endl;
        return -1;
    }
    if (!strcmp(argv[3], "zip"))
    {
        clock_t starttime, finishtime;//记录时间
        double  duration;
        starttime = clock();
        origin_file_in(argv[1]);
        code();
        zip_file_out(argv[2]);
        finishtime = clock();
        duration = ((double)finishtime - (double)starttime) / CLOCKS_PER_SEC;
        cout << "压缩用时：" << duration << "s" << endl;
    }
    else if (!strcmp(argv[3], "unzip"))
    {
        clock_t starttime, finishtime;//记录时间
        double  duration;
        starttime = clock();
        decode(argv[1]);
        decode_file_out(argv[2]);
        finishtime = clock();
        duration = ((double)finishtime - (double)starttime) / CLOCKS_PER_SEC;
        cout << "解压用时：" << duration << "s" << endl;
    }
    return 0;
}
```

装 订 线