

1. 读取职业

```
1 void read_occupation(std::string file_path, std::map<std::string,
2   std::string>& keyValueMap){
3     std::ifstream file(file_path); // 打开名为"occupation.txt"的文件
4     if (!file.is_open()) {
5         std::cerr << "cant not open the file." << std::endl;
6         return;
7     }
8     std::string line;
9     while (std::getline(file, line)) {
10        size_t pos = line.find(':');
11        if (pos != std::string::npos) {
12            std::string key = line.substr(0, pos);
13            std::string value = line.substr(pos + 1);
14            keyValueMap[key] = value;
15        }
16    }
17    file.close(); // 关闭文件
18 }
```

2. user

```
1
2 class UserInfo {
3 public:
4     enum class AgeRange {
5         UNDER_18,
6         AGE_18_24,
7         AGE_25_34,
8         AGE_35_44,
9         AGE_45_49,
10        AGE_50_55,
11        ABOVE_56
12    };
13
14    UserInfo(int userID, char gender, AgeRange age, int occupation, int
15      zipCode)
16        : userID(userID), gender(gender), age(age),
17      occupation(occupation), zipCode(zipCode) {}
18
19    [[nodiscard]] int getUserID() const { return userID; }
20    [[nodiscard]] char getGender() const { return gender; }
21    [[nodiscard]] AgeRange getAge() const { return age; }
22    [[nodiscard]] int getOccupation() const { return occupation; }
23    [[nodiscard]] int getZipCode() const { return zipCode; }
24
25 private:
26     int userID;
27     char gender;
```

```

26     AgeRange age;
27     int occupation;
28     int zipCode;
29 };
30

```

```

1  class BitCompressor {
2  public:
3      BitCompressor() : data_(0), numBits_(0) {}
4
5      void Append(uint32_t value, uint8_t numBits) {
6          if (numBits_ + numBits > 32) {
7              // Not enough space in the current 32 bits, flush and append to
              // the next 32 bits.
8              Flush();
9          }
10         data_ |= (value & ((1U << numBits) - 1)) << numBits_;
11         numBits_ += numBits;
12     }
13
14     void Flush() {
15         if (numBits_ > 0) {
16             compressedData_.push_back(data_);
17             data_ = 0;
18             numBits_ = 0;
19         }
20     }
21
22     [[nodiscard]] const std::vector<uint32_t>& GetCompressedData() const {
23         return compressedData_;
24     }
25
26 private:
27     uint32_t data_;
28     uint8_t numBits_;
29     std::vector<uint32_t> compressedData_;
30 };
31

```

```

1  int main() {
2      MyError error;
3      std::map<std::string, std::string> occupation;
4      std::vector<UserInfo> users;
5      read_occupation(DATADIR+"occupation.txt", occupation, error);
6      if(!error.pass()){
7          std::cout<<error.getErrorCode()<<" "<<error.getErrorDescription()
<<std::endl;
8      }
9      read_user_info(DATADIR+"users.dat", users, error);
10     if(!error.pass()){
11         std::cout<<error.getErrorCode()<<" "<<error.getErrorDescription()
<<std::endl;
12     }
13
14
15     BitCompressor compressor;

```

```

16
17 // 压缩数据
18 for (const UserInfo& user : users) {
19     compressor.Append(user.getUserID(), 16);
20     compressor.Append(user.getGender() == 'M' ? 0 : 1, 1);
21     compressor.Append(static_cast<uint32_t>(user.getAge()), 3);
22     compressor.Append(user.getOccupation(), 10);
23     compressor.Append(user.getZipCode(), 32);
24 }
25
26 // 结束并刷新压缩器
27 compressor.Flush();
28
29 // 获取压缩后的数据
30 const std::vector<uint32_t>& compressedData =
compressor.GetCompressedData();
31
32 // 输出压缩前的数据大小
33 std::cout << "before zip: " << users.size() * sizeof(UserInfo) << "
bytes" << std::endl;
34
35 // 输出压缩后的数据大小
36 std::cout << "after zip: " << compressedData.size() * sizeof(uint32_t)
<< " bytes" << std::endl;
37 return 0;
38 }

```

```

1 before zip: 120800 bytes
2 after zip: 48320 bytes

```

```

1 Compressed value: 00000000101000010000000000000001
2 Compressed value: 00000000000000000000000000000000
3 Compressed value: 00000001000011000000000000000010
4 Compressed value: 00000000000000000000000000000000
5 Compressed value: 00000000111101000000000000000011
6 Compressed value: 00000000000000000000000000000000
7 Compressed value: 00000000011110000000000000000100
8 Compressed value: 00000000000000000000000000000000
9 Compressed value: 000000010100010000000000000000101
10 Compressed value: 00000000000000000000000000000000
11 Compressed value: 000000001001101100000000000000110
12 Compressed value: 00000000000000000000000000000000
13 Compressed value: 000000000001011000000000000000111
14 Compressed value: 00000000000000000000000000000000
15 Compressed value: 000000001100010000000000000001000
16 Compressed value: 00000000000000000000000000000000
17 Compressed value: 000000010001010000000000000001001

```

```

1
2
3 #ifndef DATA_DEAL_COMPRESSOR_H
4 #define DATA_DEAL_COMPRESSOR_H
5
6

```

```

7  #include <iostream>
8  #include <vector>
9  #include <cstdint>
10 #include <bitset>
11 class BitCompressor {
12 public:
13     BitCompressor() : data_(0), numBits_(0) {}
14
15     template <typename T>
16     void Append(T value, uint8_t numBits) {
17         if (numBits_ + numBits > 32) {
18             // 如果当前 32 位不够存放，需要刷新
19             Flush();
20         }
21         data_ |= (value & ((1U << numBits) - 1)) << numBits_;
22         numBits_ += numBits;
23     }
24
25     void Flush() {
26
27         if (numBits_ > 0) {
28             // 将当前 32 位添加到压缩数据
29             compressedData_.push_back(data_);
30             data_ = 0;
31             numBits_ = 0;
32         }
33     }
34
35     [[nodiscard]] const std::vector<uint32_t>& GetCompressedData() const {
36         return compressedData_;
37     }
38
39 private:
40     uint32_t data_; // 用于存放当前 32 位数据
41     uint8_t numBits_; // 当前已存的比特数
42     std::vector<uint32_t> compressedData_; // 存放压缩后的数据
43 };
44
45 class BitDecompressor {
46 public:
47     explicit BitDecompressor(const std::vector<uint32_t>& compressedData)
48         : compressedData_(compressedData), dataIndex_(0),
49         currentData_(0), currentBit_(0) {}
50
51     template <typename T>
52     bool GetNextValue(T& value, uint8_t numBits) {
53         if (dataIndex_ >= compressedData_.size() * 32) {
54             return false; // 已经处理完所有数据
55         }
56
57         // 从当前数据中提取 numBits 位数据
58         uint32_t mask = (1U << numBits) - 1;
59         value = static_cast<T>((compressedData_[dataIndex_ / 32] >>
60             currentBit_) & mask);
61
62         // 更新当前位偏移和索引
63         currentBit_ += numBits;
64         if (currentBit_ >= 32) {

```

```
63         currentBit_ = 0;
64         dataIndex_++;
65     }
66
67     return true;
68 }
69
70 private:
71     const std::vector<uint32_t>& compressedData_;
72     size_t dataIndex_;
73     uint32_t currentData_;
74     uint8_t currentBit_;
75 };
76
77 #endif //DATA_DEAL_COMPRESSOR_H
78
```