CS205 C/ C++ Programming - Lab Assignment 4

Name: 唐千栋(Qiandong Tang)

SID: 11612730

Part 1 - Analysis

The problem is to display on the standard output the name of the block to which most characters belong. First, we define a struct `Block` to store the language of each Unicode block. Second, we write a funciton `search` to binary search which block the unicode value belongs to. Then for each unicode value we read from the text, we find the corresponding block of it and count it. Finally, we would print the max count of block.

Part 2 - Code

```
#include <algorithm>
#include <cstdio>
#include <cstring>
#include <fstream>
#include <iostream>
#include <sstream>
#include <string>
#include <vector>
#include "utf8.h"
#define pb push_back
using namespace std;

const int maxn = 3e3 + 10;

string trim(string s) {
    s.erase(s.find_last_not_of(" \t\r\n") + 1);
    s.erase(0, s.find_first_not_of(" \t\r\n"));
    return s;
}

vector<string> Split(const string& s, const string& split_str) {
    string::size_type pos1, pos2;
    pos1 = 0;
    pos2 = s.find(split_str);
    vector<string> v;
    while (pos2 != string::npos) {
        v.pb(s.substr(pos1, pos2 - pos1));
        pos1 = pos2 + split_str.size();
        pos2 = s.find(split_str, pos1);
    }
```

```
    if (pos1 != s.size()) v.pb(s.substr(pos1));
    return v;
}

int hex_string_to_int(const string& s) {
    int res = stoi(s.c_str(), NULL, 16);
    return res;
}

struct Block {
    int start, end;
    string lang;
};

ostream& operator<<(ostream& out, const Block& block) {
    out << "Language: " << block.lang << ", Range: " << block.start << "-"
        << block.end;
    return out;
}

int sz;
Block blocks[maxn];

void read_utf(string path) {
    ifstream ifile(path);
    string line;
    sz = 0;
    while (getline(ifile, line)) {
        if (line[0] == '#' || line.empty()) continue;
        istringstream sin(line);
        string field;
        vector<string> v;
        while (getline(sin, field, ';')) {
            field = trim(field);
            v.pb(field);
        }
        vector<string> vs = Split(v[0], "..");
        blocks[sz].start = hex_string_to_int(vs[0]);
        blocks[sz].end = hex_string_to_int(vs[1]);
        blocks[sz].lang = v[1];
        sz++;
    }
}

int search(unsigned int codepoint) {
    int l = 0;
    int r = sz - 1;
    int mid;
    if (codepoint > blocks[r].end) return -1;
```

```
    while (l <= r) {
        mid = (l + r) >> 1;
        if (blocks[mid].start <= codepoint && codepoint <= blocks[mid].end) {
            break;
        }
        if (codepoint > blocks[mid].end) {
            l = mid + 1;
        } else {
            r = mid - 1;
        }
    }
    return mid;
}

int cnts[maxn];

int main() {
    read_utf("Blocks.txt");
    memset(cnts, 0, sizeof(int));
    string s;
    unsigned char* p;
    int bytes_in_char;
    unsigned int codepoint;
    int x;

    while (!cin.eof()) {
        cin >> s;
        p = (unsigned char*)s.c_str();
        while (*p) {
            codepoint = utf8_to_codepoint(p, &bytes_in_char);
            if (codepoint) {
                // printf("%c %u (%X) %d byte character\n", *p, codepoint,
                //    codepoint, bytes_in_char);
                // p += bytes_in_char;  // Same as the line that follows
                _utf8_incr(p);
            } else {
                printf("%c Invalid UTF-8\n", *p);
                p++;
            }
            x = search(codepoint);
            if (x >= 0) cnts[x]++;
        }
    }
    cout << blocks[max_element(cnts, cnts + sz) - cnts].lang << endl;
    return 0;
}

```

Part 3 - Result & Verification

Test case:

```
→  Assignment4 git:(master) ✗ ./main < ./TEST\ DATA\ FOR\ LAB\ 4/sample.txt
Armenian
→  Assignment4 git:(master) ✗ ./main < ./TEST\ DATA\ FOR\ LAB\ 4/sample2.txt
Georgian
→  Assignment4 git:(master) ✗ ./main < ./TEST\ DATA\ FOR\ LAB\ 4/sample3.txt
Lao
→  Assignment4 git:(master) ✗ ./main < ./TEST\ DATA\ FOR\ LAB\ 4/sample4.txt
Malayalam
→  Assignment4 git:(master) ✗ ./main < ./TEST\ DATA\ FOR\ LAB\ 4/sample5.txt
Devanagari
→  Assignment4 git:(master) ✗ ./main < ./TEST\ DATA\ FOR\ LAB\ 4/sample6.txt
Georgian
```

Part 4 - Difficulties & Solutions

1. How to convert character to unicode value

   using `codepoint = utf8_to_codepoint(p, &bytes_in_char);`

2. How to split string

```
vector<string> Split(const string& s, const string& split_str) {
    string::size_type pos1, pos2;
    pos1 = 0;
    pos2 = s.find(split_str);
    vector<string> v;
    while (pos2 != string::npos) {
        v.pb(s.substr(pos1, pos2 - pos1));
        pos1 = pos2 + split_str.size();
        pos2 = s.find(split_str, pos1);
    }
    if (pos1 != s.size()) v.pb(s.substr(pos1));
    return v;
}
```