1.

/\*\*

\* 比较两个字符串的字符序列。

\*

\* @param p1 字符串1的指针。

\* @param p2 字符串2的指针。

\* @return 如果字符串1小于字符串2，则返回负值；如果字符串1大于字符串2，则返回正值；如果两个字符串相等，则返回0。

\*/

int strcmp(const char \*p1, const char \*p2) {

while (\*p1 != '\0' && \*p2 != '\0') {

if (\*p1 > \*p2) return \*p1 - \*p2;

else if (\*p1 < \*p2) return \*p1 - \*p2;

else {

p1++;

p2++;

}

}

if (\*p1 == '\0' && \*p2 == '\0') return 0;

else if (\*p1 == '\0') return -(\*p2);

else return \*p1;

}

/\*\*

\* 比较两个字符串的前n个字符。

\*

\* @param p1 字符串1的指针。

\* @param p2 字符串2的指针。

\* @param n 要比较的字符数。

\* @return 如果字符串1的前n个字符小于字符串2的前n个字符，则返回负值；如果大于，则返回正值；如果相等，则返回0。

\*/

int strncmp(const char \*p1, const char \*p2, int n) {

int i = 0;

while (i < n && \*p1 != '\0' && \*p2 != '\0') {

if (\*p1 > \*p2) return \*p1 - \*p2;

else if (\*p1 < \*p2) return \*p1 - \*p2;

else {

p1++;

p2++;

}

i++;

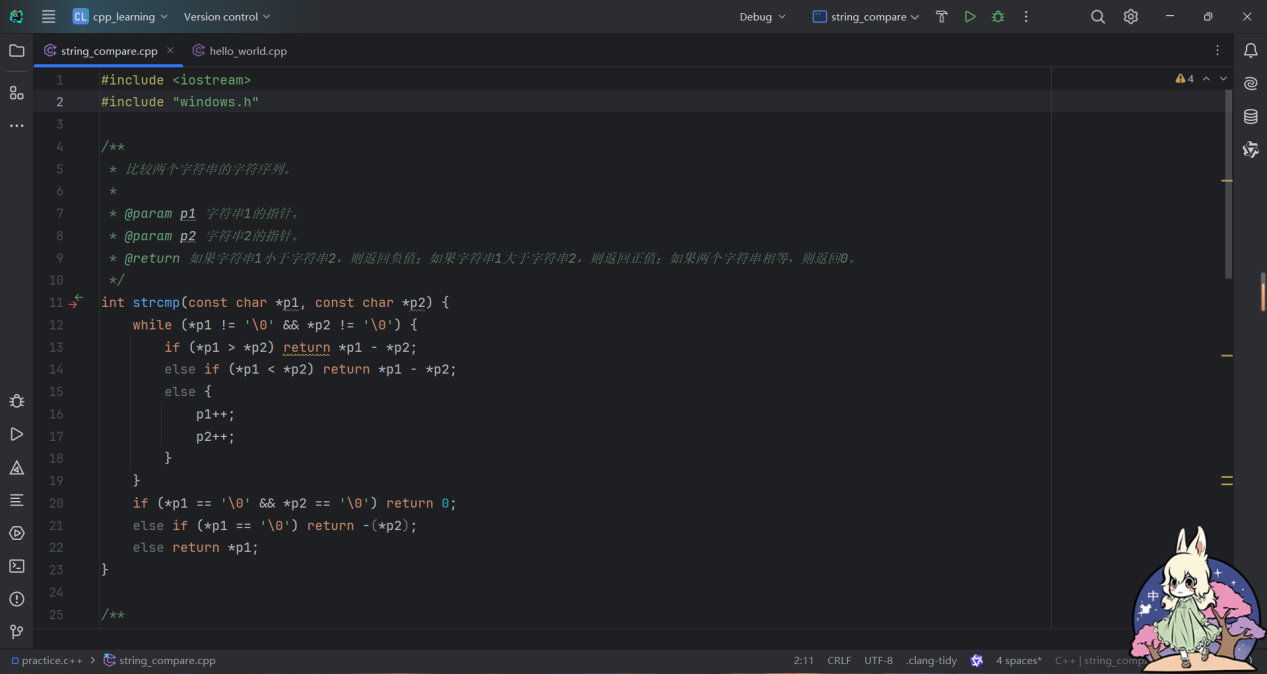
}

if (i == n) return 0;

else if (\*p1 == '\0') return -(\*p2);

else return \*p1;

}



实现结果

// 使用strcmp比较两个字符串的完整内容

int result = strcmp(str1, str2);

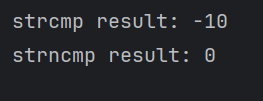
std::cout << "strcmp result: " << result << std::endl;

// 使用strncmp比较两个字符串的前3个字符

int n = 3;

int result\_n = strncmp(str1, str2, n);

std::cout << "strncmp result: " << result\_n << std::endl;



2.

/\*\*

\* 自定义实现字符串连接函数strcat

\* @param dest 目标字符串，将源字符串src追加到该字符串末尾

\* @param src 源字符串，内容将被追加到目标字符串dest末尾

\* @return 返回修改后的目标字符串指针

\*/

// 自己实现的strcat函数

char\* strcat(char \*dest, const char \*src) {

char \*ptr = dest + strlen(dest); // 指针指向dest字符串的末尾

while (\*src != '\0') {

\*ptr++ = \*src++;

}

\*ptr = '\0';

return dest;

}

/\*\*

\* 自定义实现限定长度的字符串连接函数strncat

\* @param dest 目标字符串，将源字符串src的前n个字符追加到该字符串末尾

\* @param src 源字符串，内容将被追加到目标字符串dest末尾

\* @param n 指定追加的字符数上限

\* @return 返回修改后的目标字符串指针

\*/

// 自己实现的strncat函数

char\* strncat(char \*dest, const char \*src, size\_t n) {

char \*ptr = dest + strlen(dest); // 指针指向dest字符串的末尾

int count = 0;

while (\*src != '\0' && count < n) {

\*ptr++ = \*src++;

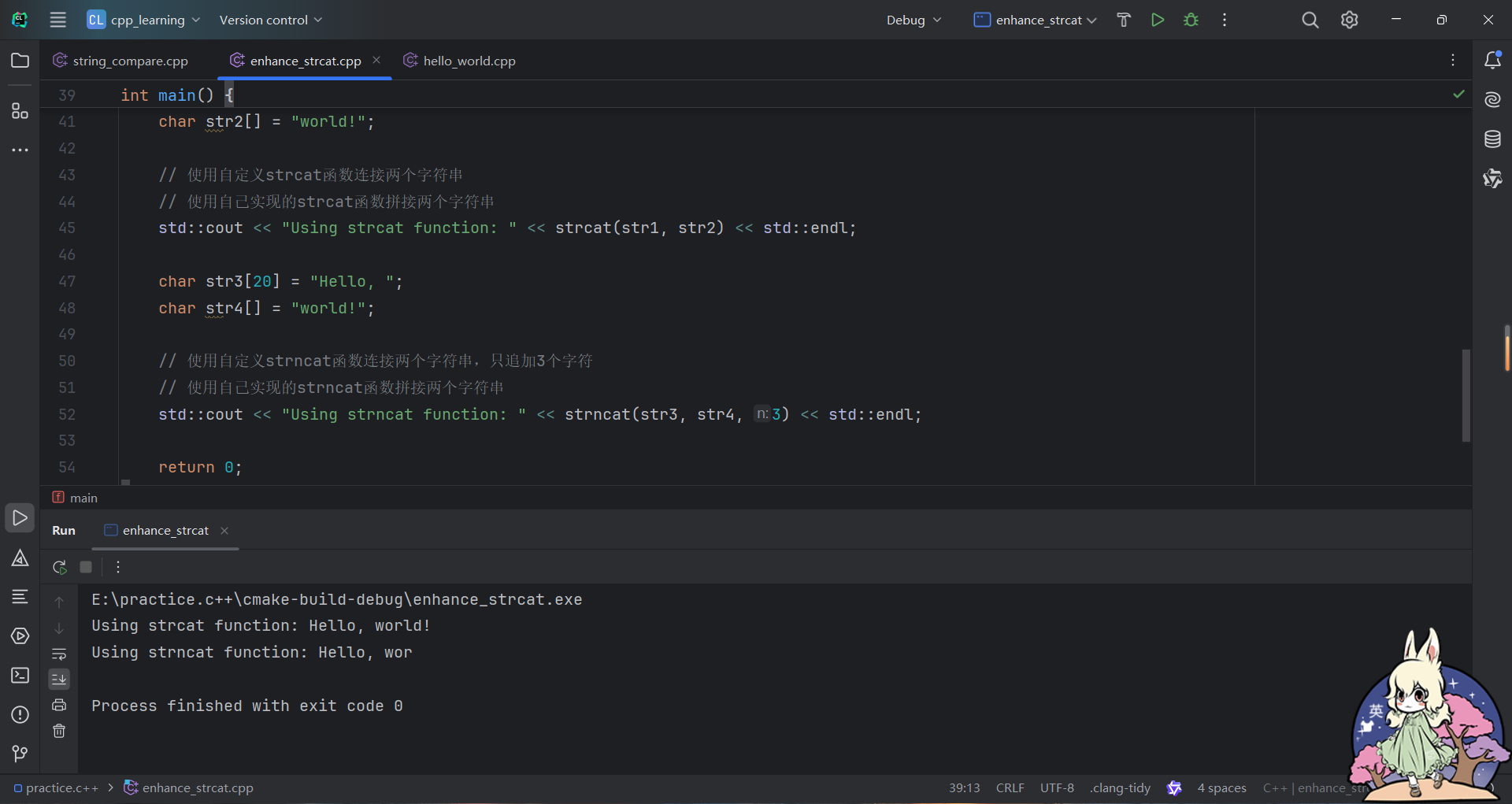
count++;

}

\*ptr = '\0';

return dest;

}



运行结果

int main() {

char str1[20] = "Hello, ";

char str2[] = "world!";

// 使用自定义strcat函数连接两个字符串

// 使用自己实现的strcat函数拼接两个字符串

std::cout << "Using strcat function: " << strcat(str1, str2) << std::endl;

char str3[20] = "Hello, ";

char str4[] = "world!";

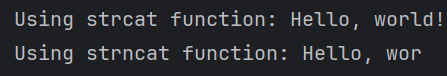
// 使用自定义strncat函数连接两个字符串，只追加3个字符

// 使用自己实现的strncat函数拼接两个字符串

std::cout << "Using strncat function: " << strncat(str3, str4, 3) << std::endl;

return 0;

}



3.

#include <iostream>

#include <string>

#include <algorithm>

using namespace std;

int main() {

const int SIZE = 10;

string\* arr = new string[SIZE]; // 动态分配一个数组用于存储字符串

// 输入十个字符串

cout << "请输入10个字符串：" << endl;

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

cin >> arr[i];

}

// 使用标准库的 sort 函数对字符串进行排序

sort(arr, arr + SIZE);

// 输出已排好序的字符串

cout << "已排好序的字符串：" << endl;

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

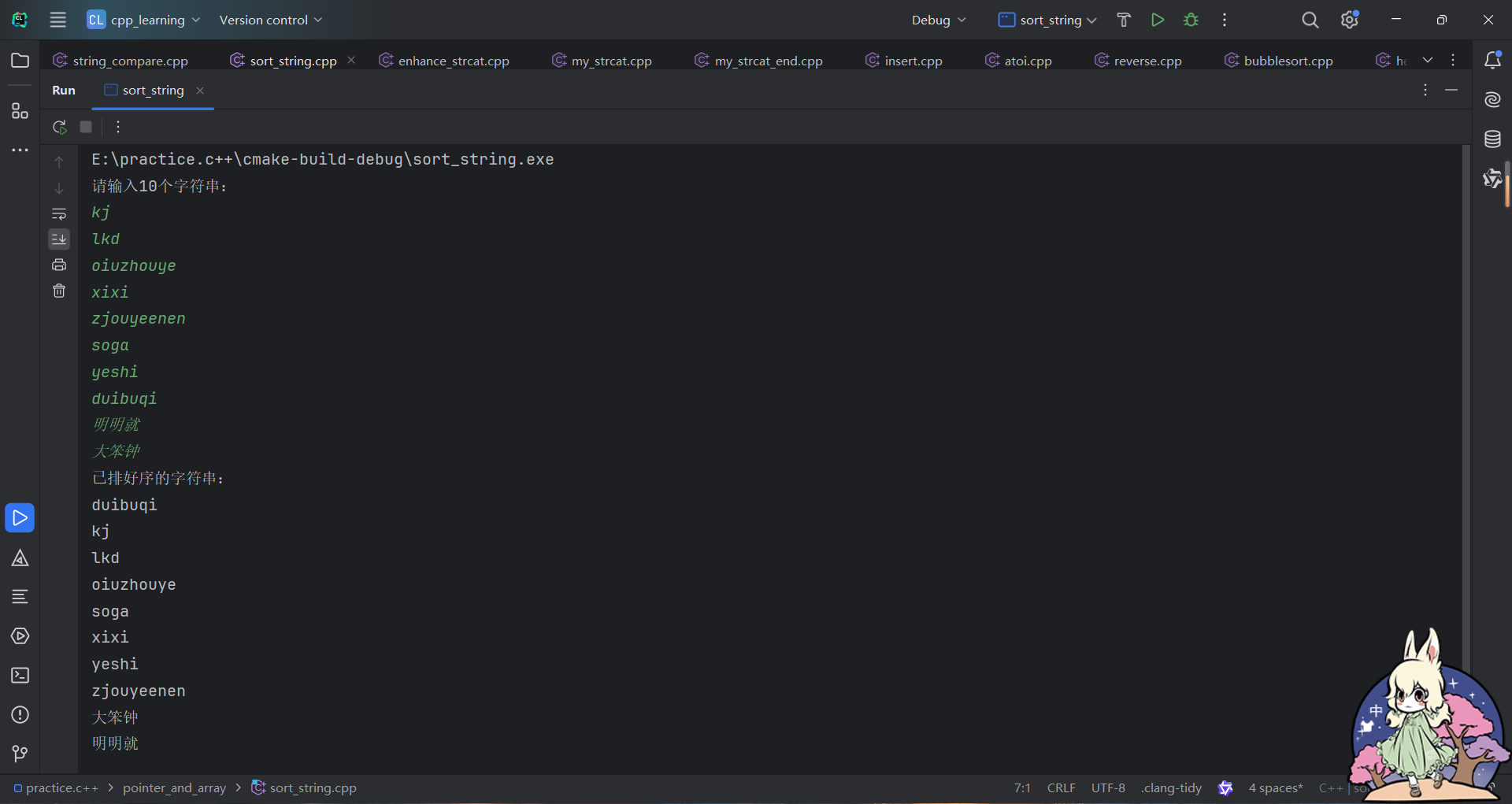
cout << arr[i] << endl;

}

delete[] arr; // 释放动态分配的内存

return 0;

}



4.

#include <iostream>

#include <sstream>

#include <string>

using namespace std;

int main() {

string line;

int count = 0;

cout << "请输入文本（按 Ctrl+D 结束输入）：" << endl;

// 持续读取标准输入直到 EOF

while (getline(cin, line)) {

istringstream iss(line);

string word;

// 逐个单词提取并比较

while (iss >> word) {

if (word == "the") { // 精确匹配小写的 "the"

++count;

}

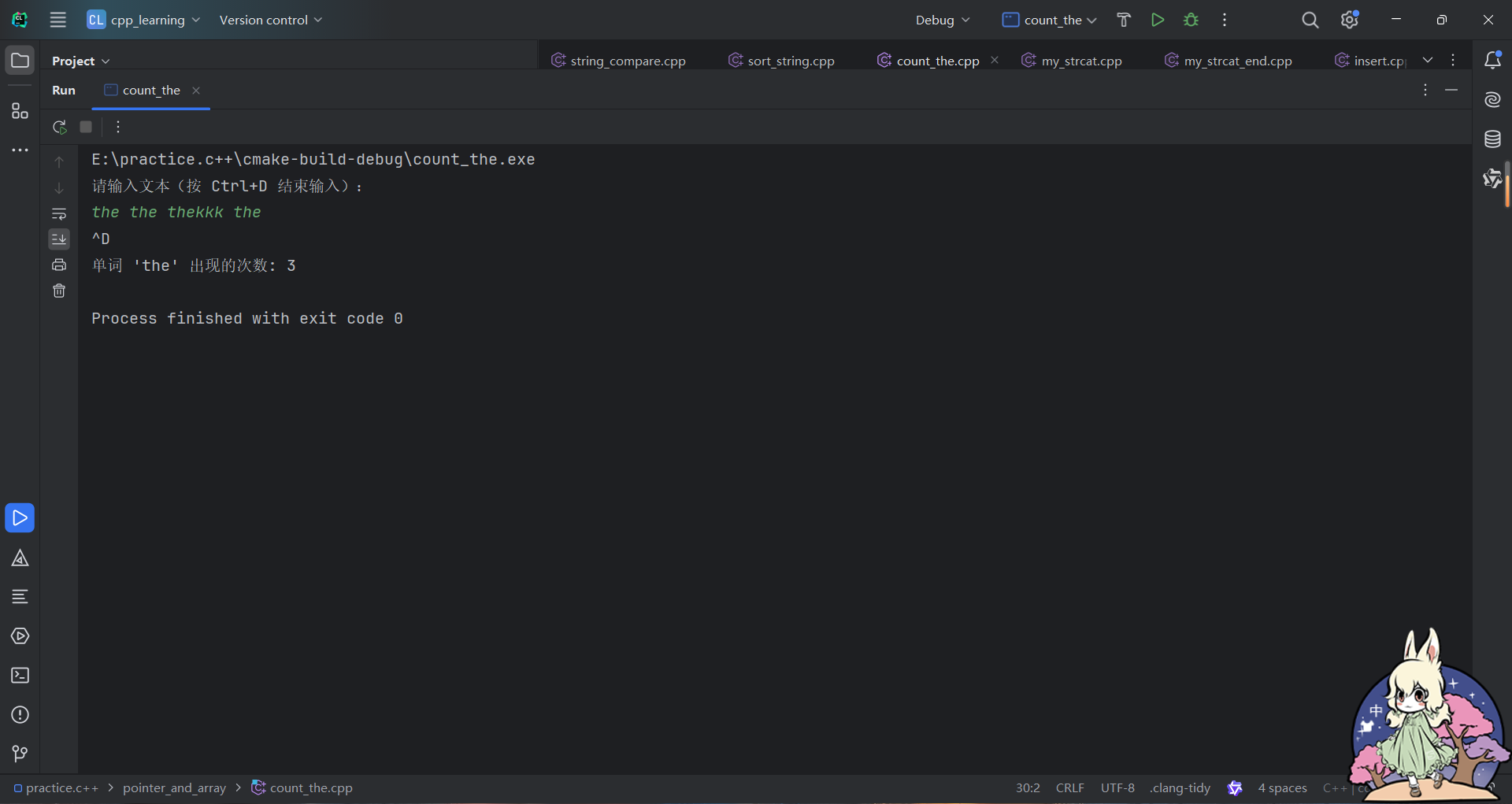
}

}

// 输出结果

cout << "单词 'the' 出现的次数: " << count << endl;

return 0;

}

5.

#include <iostream>

using namespace std;

// 函数声明

void search(int \*pa, int n, int \*pmax, int \*pflag);

int main() {

const int SIZE = 10; // 定义数组的大小

int arr[SIZE]; // 声明一维数组

int maxValue; // 存储最大值

int maxIndex; // 存储最大值的下标

// 输入 10 个整数

cout << "请输入 10 个整数：" << endl;

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

cin >> arr[i];

}

// 调用 search 函数

search(arr, SIZE, &maxValue, &maxIndex);

// 输出最大值及其下标

cout << "数组中的最大值是: " << maxValue << endl;

cout << "该值所在的下标是: " << maxIndex << endl;

return 0;

}

// search 函数定义

void search(int \*pa, int n, int \*pmax, int \*pflag) {

\*pmax = pa[0]; // 初始化最大值为数组的第一个元素

\*pflag = 0; // 初始化下标为 0

// 遍历数组寻找最大值及其下标

for (int i = 1; i < n; ++i) {

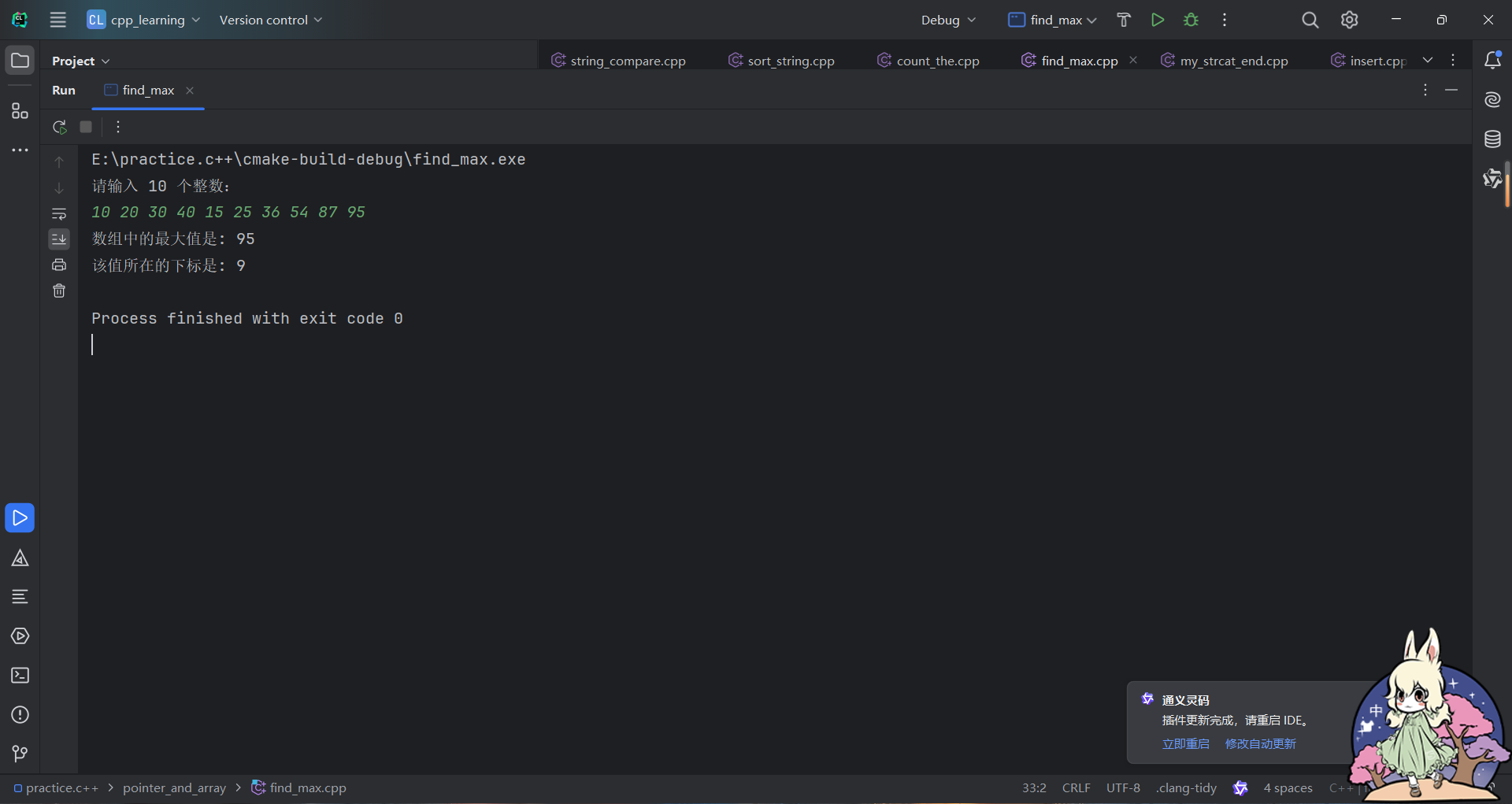
if (pa[i] > \*pmax) {

\*pmax = pa[i]; // 更新最大值

\*pflag = i; // 更新最大值的下标

}

}

}

6.

#include <iostream>

#include <cstring> // 用于 strlen

using namespace std;

// 函数声明

void inputStrings(char \*\*arr, int &count);

char\* findLongestString(char \*\*arr, int count);

int main() {

const int MAX\_STRINGS = 100; // 最多保存的字符串数量

const int MAX\_LENGTH = 6; // 每个字符串的最大长度（包含结束符'\0'）

// 动态分配内存以存储字符串

char \*\*arr = new char\*[MAX\_STRINGS];

for (int i = 0; i < MAX\_STRINGS; ++i) {

arr[i] = new char[MAX\_LENGTH]; // 每个字符串的空间

}

int count = 0; // 输入的字符串数量

// 输入字符串

inputStrings(arr, count);

// 查找最长的字符串

char \*longestString = findLongestString(arr, count);

// 输出最长的字符串

if (longestString) {

cout << "最长的字符串是: " << longestString << endl;

} else {

cout << "没有输入有效的字符串。" << endl;

}

// 释放动态分配的内存

for (int i = 0; i < MAX\_STRINGS; ++i) {

delete[] arr[i];

}

delete[] arr;

return 0;

}

// 输入字符串的函数

void inputStrings(char \*\*arr, int &count) {

cout << "请输入多个字符串（每个串不超过5个字符），以'\*\*\*\*\*'结束：" << endl;

while (true) {

cin >> arr[count];

if (strcmp(arr[count], "\*\*\*\*\*") == 0) {

break; // 输入结束

}

count++;

if (count >= 100) {

cout << "超出最大输入数量。" << endl;

break;

}

}

}

// 查找最长字符串的函数

char\* findLongestString(char \*\*arr, int count) {

char \*longest = nullptr;

int maxLength = 0;

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

int length = strlen(arr[i]);

if (length > maxLength) {

maxLength = length;

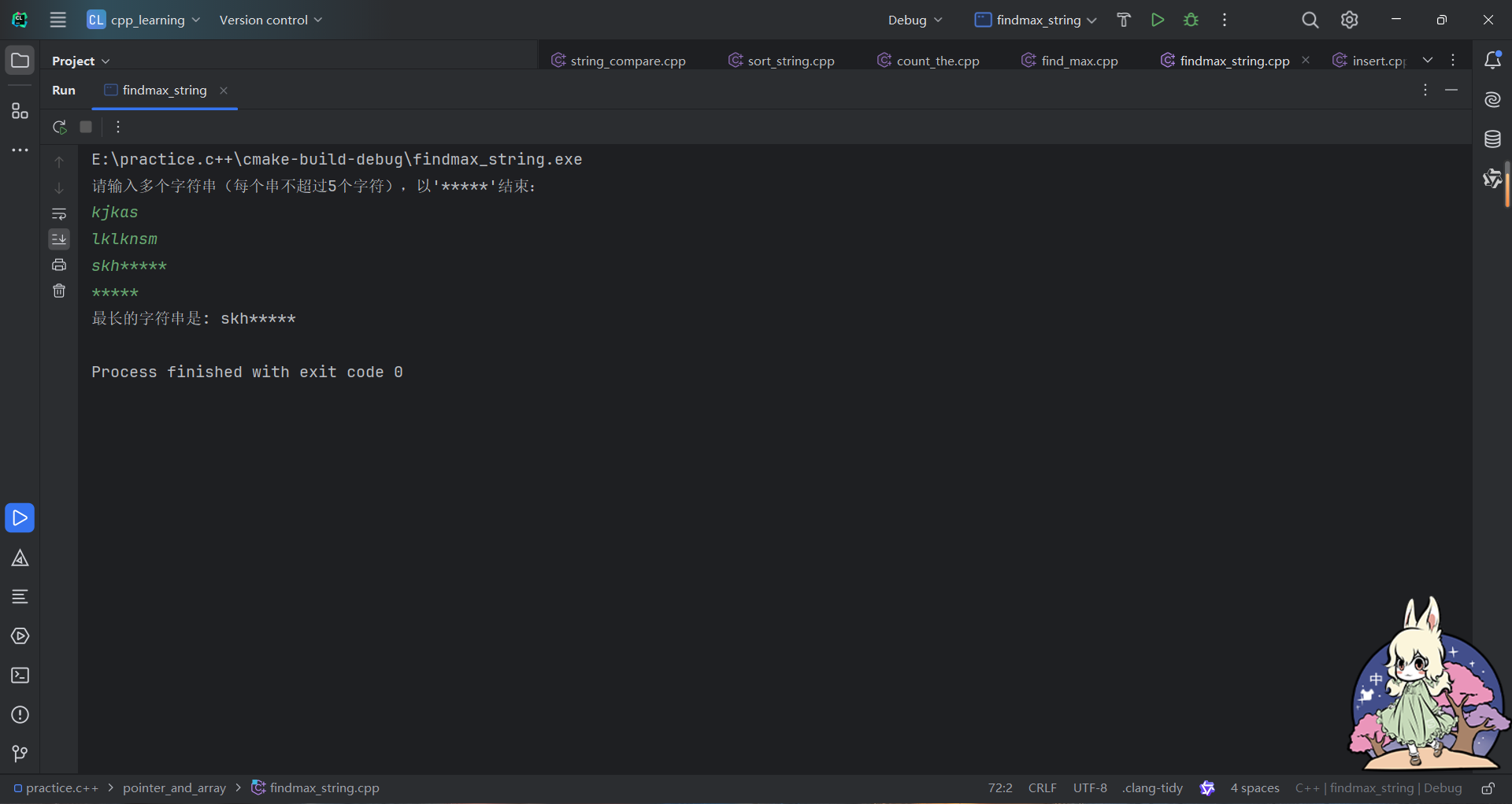
longest = arr[i];

}

}

return longest;

}



7.

#include <iostream>

#include <cstring> // 用于 strcmp 和 strcpy

using namespace std;

// 函数声明

void inputStrings(char \*\*arr, int count);

void sortStrings(char \*\*arr, int count);

void printStrings(char \*\*arr, int count);

int main() {

const int MAX\_STRINGS = 5; // 输入字符串数量

const int MAX\_LENGTH = 21; // 每个字符串的最大长度（包含结束符'\0'）

// 动态分配内存以存储字符串指针

char \*\*arr = new char\*[MAX\_STRINGS];

for (int i = 0; i < MAX\_STRINGS; ++i) {

arr[i] = new char[MAX\_LENGTH]; // 每个字符串的空间

}

// 输入5个字符串

inputStrings(arr, MAX\_STRINGS);

// 输出输入的字符串

cout << "输入的字符串是：" << endl;

printStrings(arr, MAX\_STRINGS);

// 对字符串进行排序

sortStrings(arr, MAX\_STRINGS);

// 输出已排好序的字符串

cout << "按字典序排序后的字符串是：" << endl;

printStrings(arr, MAX\_STRINGS);

// 释放动态分配的内存

for (int i = 0; i < MAX\_STRINGS; ++i) {

delete[] arr[i];

}

delete[] arr;

return 0;

}

// 输入字符串的函数

void inputStrings(char \*\*arr, int count) {

cout << "请输入5个字符串（每个字符串不超过20个字符）：" << endl;

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

cin >> arr[i];

}

}

// 输出字符串的函数

void printStrings(char \*\*arr, int count) {

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

cout << arr[i] << endl;

}

}

// 字符串排序函数

void sortStrings(char \*\*arr, int count) {

for (int i = 0; i < count - 1; i++) {

for (int j = 0; j < count - i - 1; j++) {

if (strcmp(arr[j], arr[j + 1]) > 0) {

// 交换字符串

char \*temp = arr[j];

arr[j] = arr[j + 1];

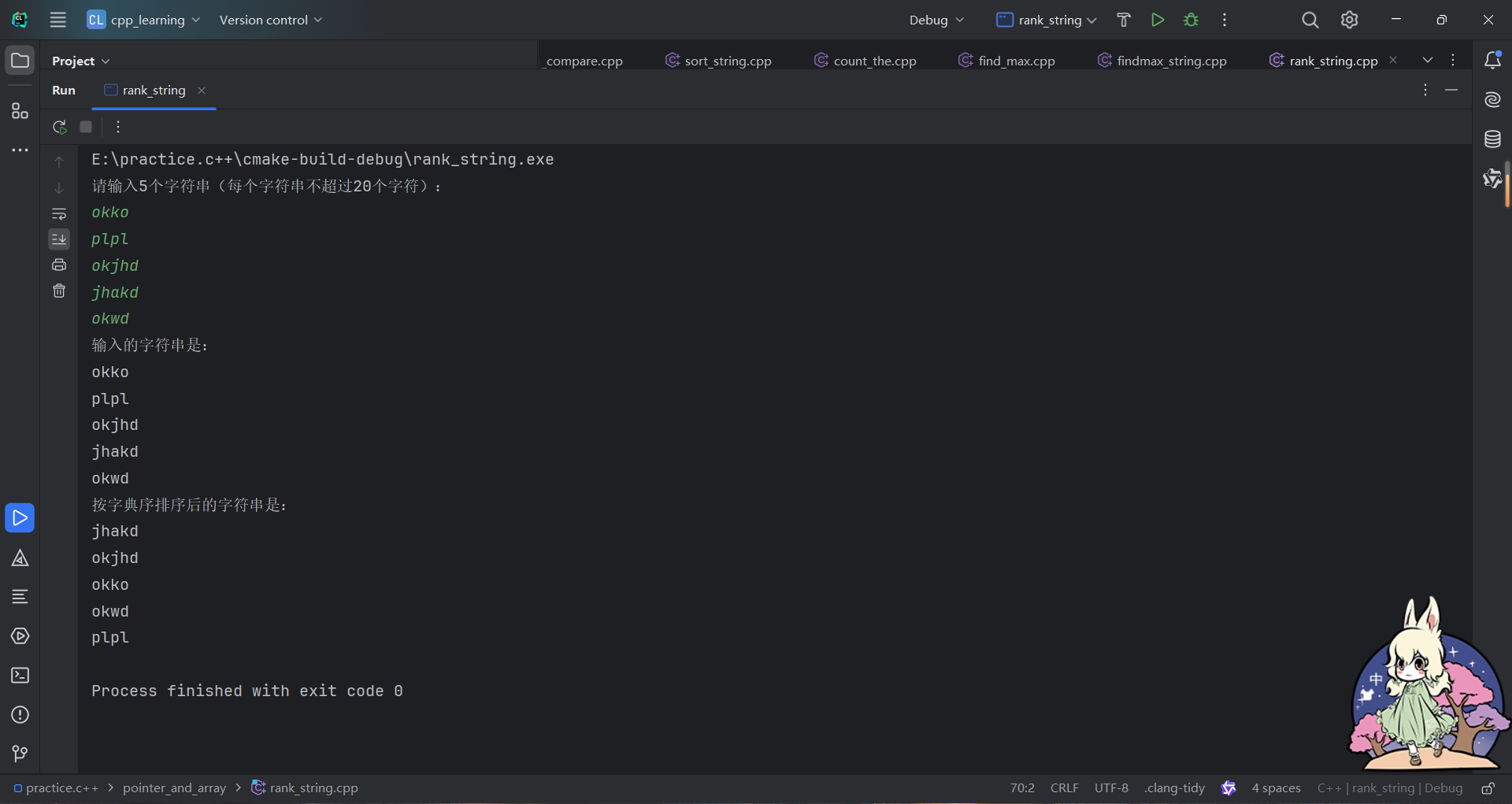
arr[j + 1] = temp;

}

}

}

}



8.

#include <iostream>

using namespace std;

// 函数声明

int FindMax(int p[][3], int m, int n, int \*pRow, int \*pCol);

int main() {

const int rows = 2;

const int cols = 3;

int matrix[rows][cols];

// 输入矩阵元素

cout << "请输入 6 个整数，以逗号分隔 (例如: 12,23,10,11,19,17): ";

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

for(int j = 0; j < cols; ++j) {

char comma;

if (j > 0) {

cin >> comma; // 读取逗号

}

cin >> matrix[i][j];

}

}

int maxRow, maxCol;

int maxValue = FindMax(matrix, rows, cols, &maxRow, &maxCol);

// 输出结果

cout << "矩阵中的最大值: " << maxValue << endl;

cout << "最大值的行下标: " << maxRow << endl;

cout << "最大值的列下标: " << maxCol << endl;

return 0;

}

// 找到最大值的函数

int FindMax(int p[][3], int m, int n, int \*pRow, int \*pCol) {

int maxVal = p[0][0]; // 假设第一个元素为最大值

\*pRow = 0; // 初始化行下标

\*pCol = 0; // 初始化列下标

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

for (int j = 0; j < n; ++j) {

if (p[i][j] > maxVal) {

maxVal = p[i][j]; // 更新最大值

\*pRow = i; // 更新行下标

\*pCol = j; // 更新列下标

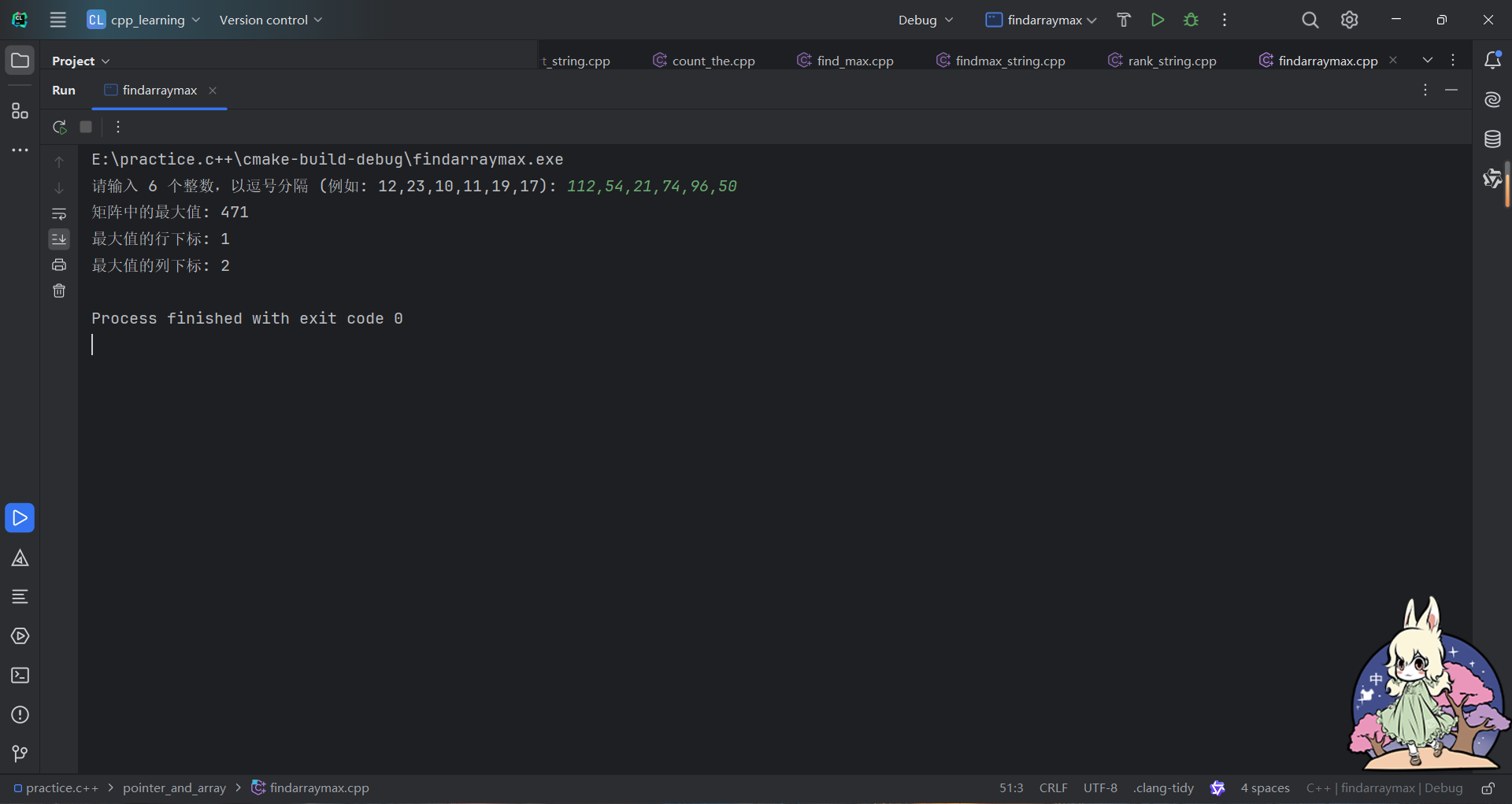
}

}

}

return maxVal; // 返回最大值

}



9.

#include <iostream>

using namespace std;

void insertStrings(const char\* s1, const char\* s2, char\* result) {

// 初始化

const char\* p1 = s1;

const char\* p2 = s2;

char\* pResult = result;

// 插入 s2 到 s1

while (\*p1 != '\0' && \*p2 != '\0') {

// 将 s1 的字符添加到结果字符串

\*pResult++ = \*p1++;

// 将 s2 的字符添加到结果字符串

\*pResult++ = \*p2++;

}

// 如果 s1 还有剩余，添加它

while (\*p1 != '\0') {

\*pResult++ = \*p1++;

}

// 如果 s2 还有剩余，添加它

while (\*p2 != '\0') {

\*pResult++ = \*p2++;

}

// 添加字符串结束符

\*pResult = '\0';

}

int main() {

char s1[27], s2[27]; // 26个字符 + 1个结束符

char result[53]; // 最长可能是 52 个字符 + 1 个结束符

// 输入字符串

cout << "请输入字符串 s1: ";

cin.getline(s1, 27); // 最多读取 26 个字符

cout << "请输入字符串 s2: ";

cin.getline(s2, 27); // 最多读取 26 个字符

// 插入操作

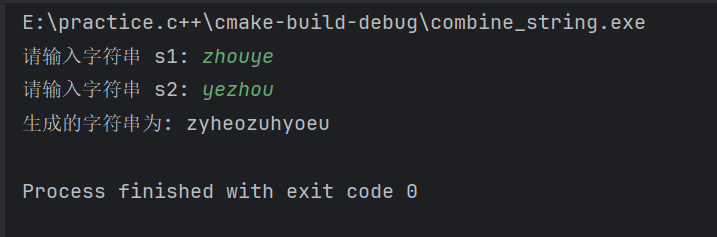
insertStrings(s1, s2, result);

// 输出结果

cout << "生成的字符串为: " << result << endl;

return 0;

}



10.

#include <iostream>

#include <cstring> // for strlen

using namespace std;

void shiftString(char\* str, int m) {

int length = strlen(str);

// 如果 m 大于字符串长度，进行取模处理

m = m % length;

// 先将后半部分拷贝到一个新的字符串

char\* temp = new char[length + 1]; // +1 for the null terminator

// 复制从 m 到结尾的部分

for (int i = 0; i < length - m; ++i) {

temp[i] = str[m + i];

}

// 复制前 m 个字符

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

temp[length - m + i] = str[i];

}

// 添加结束符

temp[length] = '\0';

// 将新的字符串内容再复制回原字符串

strcpy(str, temp);

// 释放临时字符串的内存

delete[] temp;

}

int main() {

char str[21]; // 最长20个字符 + 1个结束符

int m;

cout << "请输入一个长度不超过20的字符串: ";

cin.getline(str, 21); // 读取输入字符串

cout << "请输入平移的值 m: ";

cin >> m; // 读取平移值

// 调用函数进行字符串平移

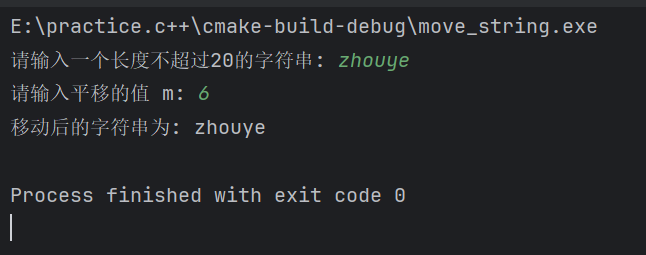
shiftString(str, m);

// 输出结果

cout << "移动后的字符串为: " << str << endl;

return 0;

}



11.

#include <iostream>

using namespace std;

// 函数用于统计字母、数字和空格的个数

void countCharacters(const char\* str, int& letterCount, int& digitCount, int& spaceCount) {

letterCount = 0;

digitCount = 0;

spaceCount = 0;

while (\*str) { // 遍历字符串

if (isalpha(\*str)) { // 字母

letterCount++;

} else if (isdigit(\*str)) { // 数字

digitCount++;

} else if (isspace(\*str)) { // 空格

spaceCount++;

}

str++; // 移动到下一个字符

}

}

int main() {

char str[256]; // 定义合理大小的字符串数组，最多255个字符 + 1个结束符

cout << "请输入一个字符串: ";

cin.getline(str, 256); // 读取输入字符串

int letters, digits, spaces; // 存储统计结果

// 调用函数进行统计

countCharacters(str, letters, digits, spaces);

// 输出结果

cout << "字母个数: " << letters << endl;

cout << "数字个数: " << digits << endl;

cout << "空格个数: " << spaces << endl;

return 0;

}

