**《程序设计基础课程设计》课程实验报告**

|  |
| --- |
| **实验题目：**检查C源程序的圆括号和大括号是否匹配（字符串处理） |
| **班级：** **学号1：** **姓名1：**  **班级：** **学号2：** **姓名2：**  **班级：** **学号3：** **姓名3：** |
| **一、实验目的**  **（包括实验环境、实现目标等等）**  实验环境：ArchLinux 64位，gcc 版本 11.2.0 (GCC)  实现目标：检查C源程序的圆括号和大括号是否匹配。 |
| **二、方案设计**  **（包括背景、原理、程序的模块说明等等）**  背景：检查C源程序中的圆括号和大括号是否匹配  原理：编写C语言程序，通过用户输入的C源文件路径，以文本形式读取进程序，判断括号是否匹配，在控制台输出判断结果  程序模块：程序分为三个模块：分别是main函数、get\_c\_source\_file\_content函数和is\_matching\_brackets函数。 |
| **三、方案实现**  **（包括算法流程图、主要函数的介绍、算法实现的主要代码等等）**  **算法流程图：**  流程图  主要函数介绍：   1. main函数：向控制台打印提示性语句并调用其他函数，代码如下：   int main(){  char content[1000000];  int is\_valid\_file\_path = get\_c\_source\_file\_content(content);  if(!is\_valid\_file\_path){ //用户输入路径不合法  printf("The file may not exist\n");  printf("Check the path and try again\n");  return 0;  }  int result = is\_matching\_brackets(content);  if(result == 1){  printf("OK ! The brackets are matching\n");  }else{  printf("Warnning ! The brackets are not matching\n");  }  return 0;  }   1. get\_c\_source\_file\_content函数：根据用户输入的C源文件路径，利用File\*指针和fopen函数，读取文件内容，保存在参数content中，返回值为用户输入是否合法，由main函数判断返回值并进行相应操作，代码如下：   int get\_c\_source\_file\_content(char \*content){  FILE \*fp;  char path[100];  char row\_buffer[1000];  printf("Give me a C source file within 1000 lines and I'll check if the brackets match\n");  printf("Enter the path of a C source file : ");  scanf("%s", path);  fp = fopen(path, "r");  if(fp == NULL){  return 0;  }  while (fgets(row\_buffer, 1000, fp) != NULL){  strcat(content, row\_buffer);  }  fclose(fp);  return 1;  } 3. is\_matching\_brackets函数：判断函数。用stack数组模拟栈操作，遍历文件中的内容，switch判断，如果是左括号则入栈，如果是右括号，则查看栈顶元素是否为相应的左括号，如是则栈顶出栈，否则结束遍历，即括号不匹配，直到结束循环，检查栈中是否为空，空则说明匹配，否则不匹配，代码如下：  int is\_matching\_brackets(char \* content){  int i = 0;  char stack[10000];  int stack\_cnt = -1;  switch (content[i]){  case '(':  stack[++stack\_cnt] = '(';  break;  case ')':  if(stack[stack\_cnt] == '('){  stack\_cnt--;  }  else{  return 0;  }  break;  case '{':  stack[++stack\_cnt] = '{';  break;  case '}':  if(stack[stack\_cnt] == '{'){  stack\_cnt--;  }  else{  return 0;  }  break;  default:  break;  }  i++;  }  return stack\_cnt == -1;  } |
| **四、数据分析**  **(包括算法测试数据的分析等等)**   1. 程序输入一般数据时能够正确输出结果，如文件内容“（）{}”判断括号匹配，而“（{）}”判断括号不匹配 2. 但发现当数据中出现类似‘（’的字符时判断失误，原因是程序无法识别出括号是正常的括号还是字符括号，故在is\_matching\_brackets函数循环开始前加上特判代码，判断每一个括号字符左右是否被引号包围，如果是则应跳过这次循环，即continue。具体代码如下：   if(i>0){  if(content[i-1] == '\'' && content[i+1] == '\''){  i++;  continue;  }  }  于是程序判断正确 |
| **五、总结**  **（完成的心得和其他，主要是自己碰到的问题，以及解决问题的方法等）**   1. 程序设计之初，不会C语言读取文件的操作，故在互联网上查询并学会了使用File指针和fopen函数读取文件内容 2. 通过这次实验，我懂得了独立自主学习的重要性，对程序设计流程有了更清晰的认识，有了更强的工程意识。 |