**哈尔滨工程大学**

**《程序设计基础》实验报告**

**综合实践**

小组成员：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **姓名** | **班级** | **学号** | **分工** | **成绩** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

实验时间: 年 月 日

**哈尔滨工程大学计算机基础课程教学中心**

实验题目：单词学习程序

设计思想：

成员任务分工：

运行环境：glibc 2.29，gcc 8.3.0，Linux 5.1.4

编程语言：C

成员1的工作：

姓名：

学号：

任务分工：

设计思想：用List储存数据，在main里通过循环来判断操作的类型，最后调用不同的函数来使程序工作。

代码行数：209

实验代码及注释：

#**include** <**malloc**.**h**>

#**include** <**stdio**.**h**>

#**include** <**string**.**h**>

#**define** **CHANGE** 960979113899

#**define** **INSERT** 961221679226

#**define** **DELETE** 961015077464

#**define** **QUERY** 29137653787

#**define** **PRINT** 29136364274

#**define** **LODE** 882772937

#**define** **SAVE** 883009844

#**define** **EXIT** 882531359

#**define** **HELP** 882618574

#**define** **EMPTY** 741

*//* *采用List来储存数据*

**typedef** **struct** \_List List;

**struct** **\_List** {

**int** **id**;

**int** **status**;

**char** **word**[48];

List\* **next**;

};

List\* head;

**int** id\_count;

**void** **help**()

{

printf(**"insert** **word\n"**

**"delete** **id\n"**

**"change** **id** **word\n"**

**"query** **word\n"**

**"print\n"**

**"help\n"**);

}

*//* *将字符串进行散列化得到唯一整数输出，避免main函数频繁出现if*

**unsigned** **long** **hash**(**const** **char**\* str)

{

**unsigned** **long** hash = 741;

**int** c;

**while** ((c = \*str++))

hash = ((hash << 5) + hash) + c;

**return** hash;

}

**void** **insert**(**const** **char**\* s)

{

List\* p;

**if** (!head) {

head = malloc(**sizeof**(List));

p = head;

} **else** {

**for** (p = head; p->**next**; p = p->**next**)

;

p->**next** = malloc(**sizeof**(List));

p = p->**next**;

}

p->**next** = **NULL**;

p->**id** = id\_count++;

p->**status** = 0;

strcpy(p->**word**, s);

}

**int** **delete** (**int** id)

{

List \*p, \*n;

**if** (!head)

**return** -1;

**for** (p = n = head; n && n->**id** != id; p = n, n = n->**next**)

;

**if** (!n)

**return** -1;

**if** (n == head)

head = n->**next**;

**else**

p->**next** = n->**next**;

free(n);

**return** 0;

}

**int** **query**(**const** **char**\* s)

{

**for** (List\* p = head; p; p = p->**next**)

**if** (!strcmp(s, p->**word**)) {

p->**status**++;

printf(**"id=%d** **status=%d** **word=%s\n"**,

p->**id**, p->**status**, p->**word**);

**return** 0;

}

**return** -1;

}

**int** **change**(**int** id, **const** **char**\* s)

{

**for** (List\* p = head; p; p = p->**next**)

**if** (p->**id** == id) {

strcpy(p->**word**, s);

**return** 0;

}

**return** -1;

}

**void** **print**()

{

**for** (List\* p = head; p; p = p->**next**)

printf(**"id=%d** **status=%d** **word=%s\n"**,

p->**id**, p->**status**, p->**word**);

}

*//* *save和load函数用来进行数据持久化*

**void** **save**()

{

FILE\* fd = fopen(**"data"**, **"w"**);

**for** (List\* p = head; p; p = p->**next**)

fprintf(fd, **"%d** **%s\n"**, p->**status**, p->**word**);

fclose(fd);

}

**void** **load**()

{

FILE\* fd = fopen(**"data"**, **"r"**);

**if** (!fd)

**return**;

**char** buf[256];

**char** word[64];

List \*p, \*n;

**int** status;

**while** (fgets(buf, 256, fd) != **NULL**) {

sscanf(buf, **"%d** **%s"**, &status, word);

n = malloc(**sizeof**(List));

**if** (!head)

head = n;

**else**

p->**next** = n;

p = n;

n->**id** = id\_count++;

n->**status** = status;

n->**next** = **NULL**;

strcpy(n->**word**, word);

}

fclose(fd);

}

**int** **main**(**int** argc, **char**\* argv[])

{

**char** buf[256] = { **'\0'** };

**char** command[128] = { **'\0'** };

**char** word[64] = { **'\0'** };

**char**\* arg = **NULL**;

**int** id = -1;

**int** index = 0;

load();

*//* *关于错误处理*

*//* *我们需要处理以下几种情况的错误*

*//* *空命令，无效命令，有效命令错误参数*

*//* *首先通过初始化和循环结束的语句确保在sscanf即使失败，*

*//* *command，word，id着三个变量也为空字符串和-1，因为空字符串和-1，不会出现在List里。*

*//* *这样即使parse失败把着些变量传递给函数，函数也会正确处理这些变量，返回错误。*

**for** (;;) {

printf(**"prompt>** **"**);

fflush(**stdout**); *//stdin是行缓冲的流，所以这里要进行显式flush*

**if** (fgets(buf, 256, **stdin**) == **NULL**)

**break**;

sscanf(buf, **"%s%n"**, command, &index);

arg = buf + index; *//* *shift参数*

**switch** (hash(command)) {

**case** **EMPTY**:

**break**;

**case** **HELP**:

help();

**break**;

**case** **INSERT**:

sscanf(arg, **"%s"**, word);

**if** (\*word == **'\0'**)

printf(**"insert** **error\n"**);

**else**

insert(word);

**break**;

**case** **QUERY**:

sscanf(arg, **"%s"**, word);

**if** (query(word) != 0)

printf(**"query** **error\n"**);

**break**;

**case** **CHANGE**:

sscanf(arg, **"%d** **%s"**, &id, word);

**if** (change(id, word) != 0)

printf(**"change** **error\n"**);

**break**;

**case** **DELETE**:

sscanf(arg, **"%d"**, &id);

**if** (delete (id) != 0)

printf(**"delete** **error\n"**);

**break**;

**case** **PRINT**:

print();

**break**;

**case** **EXIT**:

save();

**return** 0;

**default**:

printf(**"command** **not** **found:** **%s\n"**, command);

}

id = -1;

command[0] = **'\0'**;

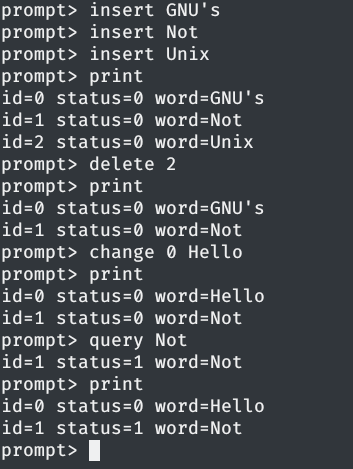
word[0] = **'\0'**;

}

**return** 0;

}

验证与结论：



成员2的工作：

姓名：

学号：

任务分工：

设计思想：

代码行数：

实验代码及注释：

验证与结论：