学号:031602523姓名:刘宏岩 学院:数计学院 专业:计算机类 班级:实验班

**《Linux 操作系统设计实践》**

**实验五:图形界面**

**实验环境:Ubuntu16.04**

**实验内容: 四则运算自动出题器**

1. **相关API学习**

**opgtk\_window\_new(GTK\_WINDOW\_TOPLEVEL)**

* 函数创建一个窗口并返回这个窗口的控件指针。
* 参数 GTK\_WINDOW\_TOPLEVEL 指明窗口的类型为最上层的主窗口，它最常 用。还可以取另一个值 GTK\_WINDOW\_POPUP 指明窗口的类型为弹出式的无边框 的窗口。

**g\_signal\_connect()**

* 使用这个宏为窗口或控件加回调函数。  
  g\_signal\_connect 宏有 4 个参数，分别是: 连接的对象，就是要连接信号的控件的指针(**注意:必须是已创建完的控件的指针**)，需要用 G\_OBJECT 宏来转换; 信号名称，就是要连接的信号名称，为字符串形式，用双引号引起来。不 同的控件拥有的信号名称是不一样的;
* 回调函数，指信号发生时调用的函数，这里只用到函数名称，需要用G\_CALLBACK 宏来转换一下;传递给回调函数的参数，它的值类型应该为 gpointer。如果不是这一类型 可以强制转换，如果没有参数则为 NULL。这里只能传递一个参数，如果有多 个参数，可以先将多个参数定义为一个结构，再将此结构作为参数传递过来

**gtk\_window\_set\_title(window,const gchar\* title)**

* 设定窗口的标题:

**gtk\_window\_set\_default\_size(window,int width,int height)**

* 设定窗口的默认宽高

**gtk\_window\_set\_position(window,GtkWindowPosition**

**position);**

* 设定窗口的位置:
* 其中 position 可以取如下值

GTK\_WIN\_POS\_NONE 不固定  
GTK\_WIN\_POS\_CENTER 居中  
GTK\_WIN\_POS\_MOUSE 出现在鼠标位置 GTK\_WIN\_POS\_CENTER\_ALWAYS 窗口改变尺寸仍居中 GTK\_WIN\_POS\_CENTER\_ON\_PARENT 居于父窗口的中部

**gtk\_container\_add ()**

* 功能是将另一控件加入到容器中来。
* 它的第一参数是 GtkContainer 型的指针，这就需要将窗口控件指针用宏 GTK\_CONTAINER 转换一下，即 GTK\_CONTAINER(window)。它的第二参数是要容 纳的控件的指针，即 button。

**gtk\_widget\_show\_all()**

* 原本每一个控件都要用函数 gtk\_widget\_show 来显示，而这个函数显示容

器中所有控件。

* 它的参数是一个容器控件的指针，例如本次实验中用到的:

gtk\_widget\_show\_all(window);

1. **实验思路**

编写一个用于小学生学习四则运算的小程序，编写一个函数用于随机生成算式，和一个计算函数用于计算随机算式的结果。根据用户输入的答案进行对比，统计正确题目和错误题目的数目。

确认按钮：用于切换下一题，并判断结果对错。

清空按钮：清空计数器，还原程序。

1. **本次实验代码**

#include <gtk/gtk.h>

#include <iostream>

#include <fstream>

#include <cstdlib>

#include <ctime>

#include <string.h>

#include <stdlib.h>

#include <string>

#include <vector>

#include <stdlib.h>

#include <stdio.h>

#include <sstream>

#include <cstring>

#include <unordered\_map>

#include <math.h>

#define random() (rand()%100000)

using namespace std;

string combine(string str1, string str2, char k)

{

string combination;

combination = str1 + k + str2;

return combination;

}

string int\_string(int number)

{

int temp = abs(number);

std::stringstream ss;

std::string str;

ss<<temp;

ss>>str;

return str;

}

int randomNunber()

{

int a = random() % 10 + 1;

return a;

}

char randomOperation()

{

int j;

char symbol[1];

j = random() % 3;

switch (j) {

case 0:

symbol[0] = '+';

break;

case 1:

symbol[0] = '-';

break;

case 2:

symbol[0] = '\*';

break;

}

return symbol[0];

}

string generateExpression()

{

int num1, num2, change, count;

char symbol;

string str\_num1, str\_num2, Equation, t;

num1 = randomNunber();

num2 = randomNunber();

count = random() % 6 + 2;

symbol = randomOperation();

str\_num1 = int\_string(num1);

str\_num2 = int\_string(num2);

Equation = combine(str\_num1, str\_num2, symbol); //随机数与随机的符号结合

for (count; count>2; count--) {

symbol = randomOperation();

str\_num1 = Equation;

change = random() % 2;

if (change == 0) {

str\_num1 = '(' + str\_num1 + ')';

}

num2 = random() % 10 + 1;

str\_num2 = int\_string(num2);

change = random() % 2;

if (change == 0) {

t = str\_num1;

str\_num1 = str\_num2;

str\_num2 = t;

}

Equation = combine(str\_num1, str\_num2, symbol);

}

// cout << Equation << "=" << endl;

//string Equation2 = Equation + '=';

//formula = Equation2;

return Equation;

}

char priority(char pre, char post)

{

if (pre == '+')

{

if (post == '+') return '>';

else if (post == '-') return '>';

else if (post == '\*') return '<';

else if (post == '/') return '<';

else if (post == '(') return '<';

else if (post == ')') return '>';

}

else if (pre == '-')

{

if (post == '-') return '>';

else if (post == '+') return '>';

else if (post == '\*') return '<';

else if (post == '/') return '<';

else if (post == '(') return '<';

else if (post == ')') return '>';

}

else if (pre == '\*')

{

if (post == '\*') return '>';

else if (post == '/') return '>';

else if (post == '+') return '>';

else if (post == '-') return '>';

else if (post == '(') return '<';

else if (post == ')') return '>';

}

else if (pre == '/')

{

if (post == '/') return '>';

else if (post == '\*') return '>';

else if (post == '+') return '>';

else if (post == '-') return '>';

else if (post == '(') return '<';

else if (post == ')') return '>';

}

else if (pre == '(')

{

if (post == '\*') return '<';

else if (post == '/') return '<';

else if (post == '+') return '<';

else if (post == '-') return '<';

else if (post == '(') return '<';

else if (post == ')') return '=';

}

}

int caculate(int Operand1, int Operand2, char Operator) ;

int calculateResult(string str) { //用于计算计算生成算式的值

vector< int > Operands; //运算数栈

vector< char > Operators; //运算符栈

int OperandTemp = 0;

char LastOperator = 0; //记录上一次所遇到的符号

for (int i = 0; i < str.size(); i++) { //此循环用于去括号

char ch = str[i];

if ('0' <= ch && ch <= '9') {

OperandTemp = OperandTemp \* 10 + ch - '0';

}

else if (ch == '+' || ch == '-' || ch == '\*' || ch == '/' || ch == '(' || ch == ')') {

if (ch != '(' && LastOperator != ')') { //结合本次和上次所遇见的符号来判断是否需要将当前存储的运算数压入栈

Operands.push\_back(OperandTemp);

OperandTemp = 0;

}

char Opt2 = ch;

for (; Operators.size() > 0;) {

char Opt1 = Operators.back();

char CompareRet = priority(Opt1,Opt2); //用当前符号与栈顶符号来对算式简化

if (CompareRet == '>') { //当前的符号的优先级小于栈顶符号时就可以将栈顶符号计算掉并将结果压入栈

int Operand2 = Operands.back();

Operands.pop\_back();

int Operand1 = Operands.back();

Operands.pop\_back();

Operators.pop\_back();

int Ret = caculate(Operand1, Operand2, Opt1);

Operands.push\_back(Ret);

}

else if (CompareRet == '<') { //当前的符号优先级大于栈顶符号不能进行运算所以跳出循环来存储当前符号

break;

}

else if (CompareRet == '=') { //这个是“（”，“） ”结合的情况 所以移除“（”并退出循环

Operators.pop\_back();

break;

}

}

if (Opt2 != ')') {

Operators.push\_back(Opt2);

}

LastOperator = Opt2;

}

}

if (LastOperator != ')') { //接下来就是计算一个不含括号的算式了

Operands.push\_back(OperandTemp);

}

for (; Operators.size() > 0;) {

int Operand2 = Operands.back();

Operands.pop\_back();

int Operand1 = Operands.back();

Operands.pop\_back();

char Opt = Operators.back();

Operators.pop\_back();

int Ret = caculate(Operand1, Operand2, Opt);

Operands.push\_back(Ret);

}

return Operands.back(); //返回结果

}

int caculate(int Operand1, int Operand2, char Operator) { //计算函数

int result = 0;

if (Operator == '+') {

result = Operand1 + Operand2;

}

if (Operator == '-') {

result = Operand1 - Operand2;

}

if (Operator == '\*') {

result = Operand1\*Operand2;

}

if (Operator == '/') {

result = Operand1 / Operand2;

}

return result;

}

GtkWidget \*lb\_problem;

GtkWidget \*lb\_correct;

GtkWidget \*lb\_wrong;

GtkWidget \*lb\_ans;

int correctNum = 0;

int wrongNum = 0;

string lastProb = "";

void refreshProblem()

{

}

void onBtnSubmit(GtkWidget \*widget, gpointer data)

{

if(lastProb == "")

{

lastProb = generateExpression();

cout << lastProb << endl;

gtk\_label\_set\_text((GtkLabel\*)lb\_problem,lastProb.c\_str());

//ans = calculateResult(s);

gtk\_entry\_set\_text((GtkEntry \*)lb\_ans,"");

}

else

{

const gchar \*myans = gtk\_entry\_get\_text((GtkEntry \*)lb\_ans);

if(atoi(myans) == calculateResult(lastProb))

{

gtk\_label\_set\_text((GtkLabel\*)lb\_correct,int\_string(++correctNum).c\_str());

cout << "Yes" << endl;

}

else

{

gtk\_label\_set\_text((GtkLabel\*)lb\_wrong,int\_string(++wrongNum).c\_str());

cout << "No" << endl;

}

gtk\_entry\_set\_text((GtkEntry \*)lb\_ans,"");

lastProb = generateExpression();

cout << lastProb << endl;

gtk\_label\_set\_text((GtkLabel\*)lb\_problem,lastProb.c\_str());

const char \*myAns = gtk\_entry\_get\_text((GtkEntry \*)lb\_ans);

}

}

void onBtnClear(GtkWidget \*widget, gpointer data)

{

lastProb = "";

gtk\_entry\_set\_text((GtkEntry \*)lb\_ans,"");

gtk\_label\_set\_text((GtkLabel\*)lb\_problem,"按提交按钮获得最新题目！");

correctNum = 0;

wrongNum = 0;

gtk\_label\_set\_text((GtkLabel\*)lb\_correct,"0");

gtk\_label\_set\_text((GtkLabel\*)lb\_wrong,"0");

}

void UI(int argc, char \*argv[])

{

gtk\_init(&argc,&argv);

GtkWidget \*window=gtk\_window\_new(GTK\_WINDOW\_TOPLEVEL);

gtk\_window\_set\_default\_size(GTK\_WINDOW(window),200,50);

g\_signal\_connect(G\_OBJECT(window),"delete\_event",G\_CALLBACK(gtk\_main\_quit),NULL);

gtk\_window\_set\_title(GTK\_WINDOW(window),"口算心算天天练");

gtk\_window\_set\_position(GTK\_WINDOW(window),GTK\_WIN\_POS\_CENTER);

gtk\_container\_set\_border\_width(GTK\_CONTAINER(window),10);

//-----------------------------------------------------

GtkWidget \*hb = gtk\_hbox\_new(0,0);

gtk\_container\_add(GTK\_CONTAINER(window),hb);

//-----------------------------------------------------

GtkWidget \*vbox = gtk\_vbox\_new(0,0);

gtk\_container\_add(GTK\_CONTAINER(hb),vbox);

//-----------------------------------------------------

GtkWidget \*hbox1 = gtk\_hbox\_new(0,0);

gtk\_box\_pack\_start(GTK\_BOX(vbox),hbox1,FALSE,FALSE,5);

GtkWidget \*lb4 = gtk\_label\_new("题目:");

gtk\_box\_pack\_start(GTK\_BOX(hbox1),lb4,1,0,5);

lb\_problem = gtk\_label\_new("按提交按钮获得最新题目！");

gtk\_box\_pack\_start(GTK\_BOX(hbox1),lb\_problem,1,0,5);

GtkWidget \*hbox11 = gtk\_hbox\_new(0,0);

gtk\_box\_pack\_start(GTK\_BOX(vbox),hbox11,FALSE,FALSE,5);

GtkWidget \*lb2 = gtk\_label\_new("正确数:");

gtk\_box\_pack\_start(GTK\_BOX(hbox11),lb2,1,0,5);

lb\_correct = gtk\_label\_new("0");

gtk\_box\_pack\_start(GTK\_BOX(hbox11),lb\_correct,1,0,5);

//-----------------------------------------------------

GtkWidget \*vb3 = gtk\_vbox\_new(0,0);

gtk\_container\_add(GTK\_CONTAINER(hb),vb3);

GtkWidget \*hb4 = gtk\_hbox\_new(0,0);

gtk\_box\_pack\_start(GTK\_BOX(vb3),hb4,FALSE,FALSE,5);

GtkWidget \*lb = gtk\_label\_new("\t您的答案:");

gtk\_box\_pack\_start(GTK\_BOX(hb4),lb,1,0,5);

lb\_ans = gtk\_entry\_new();

gtk\_box\_pack\_start(GTK\_BOX(hb4),lb\_ans,1,0,5);

GtkWidget \*hbox2 = gtk\_hbox\_new(0,0);

gtk\_box\_pack\_start(GTK\_BOX(vb3),hbox2,FALSE,FALSE,5);

GtkWidget \*lb3 = gtk\_label\_new("错误数:");

gtk\_box\_pack\_start(GTK\_BOX(hbox2),lb3,1,0,5);

lb\_wrong = gtk\_label\_new("0");

gtk\_box\_pack\_start(GTK\_BOX(hbox2),lb\_wrong,1,0,5);

//-----------------------------------------------------

GtkWidget \*vb = gtk\_vbox\_new(0,0);

gtk\_container\_add(GTK\_CONTAINER(hb),vb);

GtkWidget \*btn\_submit = gtk\_button\_new\_with\_label("提交");

gtk\_box\_pack\_start(GTK\_BOX(vb),btn\_submit,1,0,5);

g\_signal\_connect(G\_OBJECT(btn\_submit),"clicked",G\_CALLBACK(onBtnSubmit),NULL);

GtkWidget \*btn\_clear = gtk\_button\_new\_with\_label("清屏");

gtk\_box\_pack\_start(GTK\_BOX(vb),btn\_clear,1,0,5);

g\_signal\_connect(G\_OBJECT(btn\_clear),"clicked",G\_CALLBACK(onBtnClear),NULL);

//-----------------------------------------------------

gtk\_widget\_show\_all(window);//显示所有窗体

gtk\_main();

}

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

{

srand(time(0));

UI(argc,argv);

double ans = 0;

string s = "";

s = generateExpression();

cout << "所产生的算式为" << s << endl;

ans = calculateResult(s);

cout << "结果为" << ans << endl;

return 0;

}

1. **实验截图**

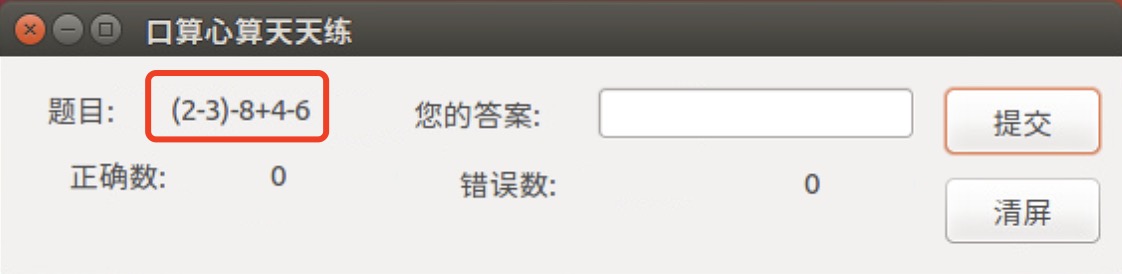
* 程序初始化

****

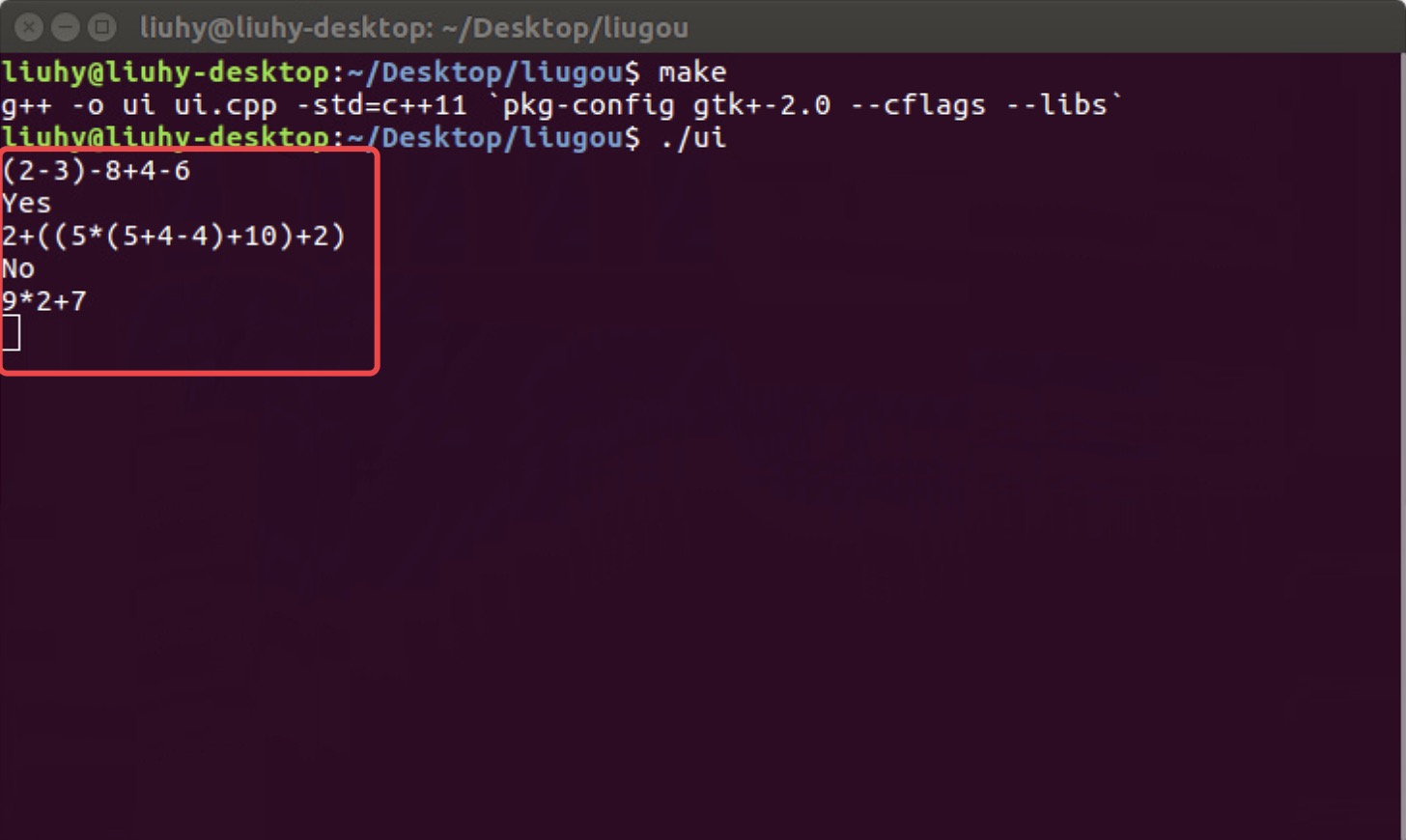
* 出题

****

* 输入正确结果后提交，正确数加一

****

* 在终端中统计题目

****