C 语言基础

5.2.1 数据类型

关于“#define”等预编译部分应用的脚本示例。

```
#define COUNT 100    //这里定义人数合计 COUNT，其值为 100
#define SALARY 4000  //每个人的薪水平均值 SALARY，其值为 4000

Action()
{
  lr_output_message("100 人合计薪资支出为：%d", COUNT *SALARY);
  return 0;
}
```

整型数据变量在 LoadRunner 中应用的脚本示例如下：

```
#define COUNT 100    //这里定义人数合计 COUNT，其值为 100
#define SALARY 4000  //每个人的薪水平均值 SALARY，其值为 4000

Action()
{
  int total;
  total = COUNT * SALARY;
  lr_output_message("100 人合计薪资支出为：%d", total);
  return 0;
}
```

计算出的合计薪资支出通过格式化输出分别表示为十进制、八进制以及十六进制，参见在 LoadRunner 中应用的脚本示例：

```
#define COUNT 100    //这里定义人数合计 COUNT，其值为 100
#define SALARY 4000  //每个人的薪水平均值 SALARY，其值为 4000

Action()
{
  int total,i, j;
  total = COUNT * SALARY;
  lr_output_message("100 人合计薪资支出为（十进制）：%d", total);
  lr_output_message("100 人合计薪资支出为(八进制）：%o", total);
  lr_output_message("100 人合计薪资支出为（十六进制）：%x", total);

  return 0;
}
```

格式化输出符号及其含义表

格 式 字 符	意　　义
d	以十进制形式输出带符号整数（正数不输出符号）
o	以八进制形式输出无符号整数（不输出前缀 o）
x，X	以十六进制形式输出无符号整数（不输出前缀 Ox）
u	以十进制形式输出无符号整数
f	以小数形式输出单、双精度实数
e，E	以指数形式输出单、双精度实数
g，G	以%f 或%e 中较短的输出宽度输出单、双精度实数
c	输出单个字符
s	输出字符串

参见在 LoadRunner 中应用的脚本示例：

```
#define PI 3.14

Action()
{
  float r = 5.5, s;
  double r1 = 22.36, s1;
  long double r2 = 876.99, s2;

  s = PI * r * r;
  s1 = PI * r1 * r1;
  s2 = PI * r2 * r2;

  lr_output_message("半径为%.2f 的面积为：%f .", r, s);
  lr_output_message("半径为%.2f 的面积为：%f .", r1, s1);
  lr_output_message("半径为%.2f 的面积为：%f .", r2, s2);
  return 0;
}
```

关于字符类型，参见下面在 LoadRunner 中应用的脚本示例代码：

```
#define CHAR 'x'

Action()
{
  char x = 'y';
  int num = 121;

  lr_output_message("常量 CHAR 用字符表示为：%c", CHAR);
  lr_output_message("常量 CHAR 用整数表示为：%d", CHAR);

  lr_output_message("----------------------------------------");

  lr_output_message("整型变量 num 用整型表示为：%d", num);
  lr_output_message("整型变量 num 用字符表示为：%c", num);
```

```
  lr_output_message("----------------------------------------");

  lr_output_message("字符型变量 x 用整型表示为：%d", x);
  lr_output_message("字符型变量 x 用字符表示为：%c", x);

  return 0;
}
```

关于字符串类型，参见下面在 LoadRunner 中应用的脚本示例代码：

```
#define STR "A"  //定义字符串常量 STR，其值为"A"

Action()
{
  char CHAR = 'A'; //定义字符变量 CHAR，其值为'A'

  lr_output_message("字符\'A\'占的空间大小为%d！", sizeof(CHAR));
  lr_output_message("字符串\"A\"占的空间大小为%d！", sizeof(STR));

  return 0;
}
```

5.2.2 C 语言语句分类

程序的功能也是由执行语句实现的，C 语句可分为以下 5 类。

- 表达式语句。

```
Action()
{
  int x, y, z;
  x = 20;
  y = 40;
  z = x + y;
  lr_output_message("%d+%d=%d", x, y, z);
  return 0;
}
```

- 函数调用语句。

在 LoadRunner 中应用函数调用语句的示例代码如下：

```
double sqrt(double x);

Action()
{
  double x;
  sqrt(100);
  lr_output_message("%f", sqrt(100));
  return 0;
}
```

在 LoadRunner 中应用 if 语句的示例代码如下：

```
Action()
{
  int randomnumber; //随机数变量
  randomnumber = rand() % 3+1; //生成一个随机数字

  if (randomnumber == 1)
  {
    web_url("www.google.com""URL=http：//www.google.com/", "Resource=0",
      "RecContentType=text/html", "Referer=", "Snapshot=t1.inf", "Mode=HTML",
      EXTRARES, "Url=http：//www.google.cn/images/nav_logo3.png",
      "Referer=http：//www.google.cn/", ENDITEM, LAST);
  }
  else if (randomnumber == 2)
  {
    web_url("www.sohu.com", "URL=http：//www.sohu.com/", "Resource=0",
      "RecContentType=text/html", "Referer=", "Snapshot=t1.inf", "Mode=HTML",
      EXTRARES, "Url=http：//images.sohu.com/uiue/sohu_logo/2005/juzhen_bg.gif",
      "Referer=http：//www.sohu.com/", ENDITEM,
      "Url=http：//images.sohu.com/cs/button/market/volunteer/760320815.swf",
      "Referer=http：//www.sohu.com/", ENDITEM,
      "Url=http：//images.sohu.com/chat_online/else/sogou/20070814/450X105.swf",
      "Referer=http：//www.sohu.com/", ENDITEM,
      "Url=http：//images.sohu.com/cs/button/market/sogou/chuxiao/7601000801.swf",
      "Referer=http：//www.sohu.com/", ENDITEM,
      "Url=http：//images.sohu.com/cs/button/huaxiashuanglong/2007/7601000816.swf",
      "Referer=http：//www.sohu.com/", ENDITEM,
      "Url=http：//images.sohu.com/cs/button/lianxiang/125-15/5901050815.swf",
      "Referer=http：//www.sohu.com/", ENDITEM, LAST);
  }
  else
  {
    web_url("www.baidu.com", "URL=http：//www.baidu.com/", "Resource=0",
      "RecContentType=text/html", "Referer=", "Snapshot=t1.inf", "Mode=HTML",
      LAST);
  }

  return 0;
}
```

在 LoadRunner 中应用 switch 语句的示例代码如下：

```
Action()
{

  int randomnumber; //随机数变量
  randomnumber = rand() % 3+1; //生成一个随机数字

  switch (randomnumber)
  {
```

```
    case 1:
      {
       web_url("www.google.com""URL=http://www.google.com/", "Resource=0",
         "RecContentType=text/html", "Referer=", "Snapshot=t1.inf",
         "Mode=HTML", EXTRARES,
         "Url=http://www.google.cn/images/nav_logo3.png",
         "Referer=http://www.google.cn/", ENDITEM, LAST);
       break;
      }
    case 2:
      {
       web_url("www.sohu.com", "URL=http://www.sohu.com/", "Resource=0",
         "RecContentType=text/html", "Referer=", "Snapshot=t1.inf",
         "Mode=HTML", EXTRARES,
         "Url=http://images.sohu.com/uiue/sohu_logo/2005/juzhen_bg.gif",
         "Referer=http://www.sohu.com/", ENDITEM,
         "Url=http://images.sohu.com/cs/button/market/volunteer/760320815.swf",
         "Referer=http://www.sohu.com/", ENDITEM,
         "Url=http://images.sohu.com/chat_online/else/sogou/20070814/450X105.swf",
         "Referer=http://www.sohu.com/", ENDITEM,
         "Url=http://images.sohu.com/cs/button/market/sogou/chuxiao/7601000801.swf",
         "Referer=http://www.sohu.com/", ENDITEM,
         "Url=http://images.sohu.com/cs/button/huaxiashuanglong/2007/7601000816.swf",
         "Referer=http://www.sohu.com/", ENDITEM,
         "Url=http://images.sohu.com/cs/button/lianxiang/125-15/5901050815.swf",
         "Referer=http://www.sohu.com/", ENDITEM, LAST);
       break;
      }
    default:
      {
       web_url("www.baidu.com", "URL=http://www.baidu.com/", "Resource=0",
         "RecContentType=text/html", "Referer=", "Snapshot=t1.inf",
         "Mode=HTML", LAST);
      }
  }

  return 0;
}
```

在 LoadRunner 中应用 do while 语句的示例代码如下：

```
Action()
{
  int i = 1;
  int sum = 0;

  do
  {
    sum = sum + i;
    i++;
  }
```

```
  while (i <= 100);

  lr_output_message("1～100 之和是：%d", sum);
  return 0;
}
```

在 LoadRunner 中应用 while 语句的示例如下：

```
Action()
{
     int i=1;
     int sum=0;

     while (i<=100) {
          sum=sum+i;
          i++;
     }

     lr_output_message("1～100 之和是：%d",sum);
     return 0;
}
```

在 LoadRunner 中应用 for 语句的示例代码如下：

```
Action()
{
     int i;
     int sum=0;

     for (i=1;i<=100;i++) sum=sum+i;

     lr_output_message("1～100 之和是：%d",sum);
     return 0;
}
```

在 LoadRunner 中应用 break 语句的示例代码如下：

```
Action()
{
   int count, total = 0;
   char buffer[1000];
   long file_stream;
   char * filename = "c：\\test.txt";

   //判断是否可以读取“c：\test.txt”文件
   if ((file_stream = fopen(filename, "r")) == NULL )
   {
     //不能读取文件，则输出错误信息“不能打开 c：\test.txt 文件！”信息
       lr_error_message ("不能打开%s 文件！", filename);

       return -1;
   }
```

```
    while (!feof(file_stream)) //如果没有读取到文件结束符，则执行循环体内容
    {
        //从文件中读取 1000 个字符
        count = fread(buffer,sizeof(char),1000,file_stream);
        // 累加，这里意义不是很大，因为第一次读取后 count=1000，一定符合后面的 if 语句
        total += count;
        if (total>=1000) //条件判断语句，这里一定会满足条件
        {
            fclose(file_stream);//关闭文件
            //输出文件的前 1000 个字符
            lr_output_message("文件的前 1000 个字符内容为：%s",buffer);
            break;//退出循环
        }
    }
    return 0;
}
```

下面，给大家再举一个 for 语句双重循环的例子：

```
Action()
{
     int i,j;//声明 2 个整型变量

     //for 语句双重循环
     for (i=1;i<=5;i++)//第 1 重循环，循环 5 次
     {
          if (i==3) break;//当 i 等于 3 时，跳出本重循环
          else lr_output_message("i=%d",i);//否则，输出 i 值

          for (j=1;j<=5;j++)//第 2 重循环，循环 5 次
          {
               if (j==2) break;//当 j 等于 2 时，跳出本重循环
               lr_output_message("j=%d",j);//输出 j 值
          }

     }
}
```

在 LoadRunner 中应用 continue 语句的示例代码如下：

```
Action()
{
     //
    int i;
    for (i=1; i<=20; i++)
    {
        if ((i%5)==0) continue; //输出 1～20 中除 5 的倍数外的所有整数
        lr_output_message ("%d",i);
    }

    return 0;
}
```

在 LoadRunner 中应用 goto 语句的示例代码如下：

```
Action()
{
  int i;

  for (i = 1; i <= 3; i++)
  {
    if (i == 2)
      goto prtname;
    else
      lr_output_message("i=%d", i);
  }

  prtname: lr_output_message("Your Name is tony");

  return 0;
}
```

在 LoadRunner 中应用 return 语句的示例代码如下：

```
Action()
{
  LPCSTR user1 = "悟空";
  LPCSTR user2 = "八戒";

  if ((user1 == "悟空") || (user1 == "猴哥"))
  {
    lr_output_message("悟空和猴哥是同一个人！");
    return 0;
  }
  else
  {
    lr_output_message("我是八戒不是悟空！");
    return  - 1;
  }
  lr_output_message("这句话永远不会被执行！");
}
```

- 复合语句。

在 LoadRunner 中应用复合语句的示例代码如下：

```
Action()
{
  double x;
  int i, j = 0;
  for (i = 0; i < 5; i++)
  {
    j++;
    lr_output_message("j=%d", j);
  }
```

```
  return 0;
}
```

- 空语句。

在 LoadRunner 中应用空语句的示例代码如下：

```
Action()
{
    int i=0;
    for (;;)
 {
       i++;
       if (i=100) break;
  }
    lr_output_message("%d",i);
}
```

5.2.3 基础知识

在 LoadRunner 中应用的示例代码如下：

```
Action()
{
  int a = 1, b = 2, c = 3, d = 4, e = 5;
  int x, y;
  LPCSTR exp1 = "-a*b%c+d+e="; //表达式"-a*b%c+d+e="
  LPCSTR exp2 = "((((-a)*b)%c)+(d+e))="; //表达式"((((-a)*b)%c)+(d+e))="

  x =-a*b%c+d+e;//取的表达式的值将结果赋值给 x
  y = ((((-a)*b)%c)+(d+e)); //取的表达式的值将结果赋值给 y

  lr_output_message("%s%d", exp1, x); //输出表达式和 x 值
  lr_output_message("%s%d", exp2, y); //输出表达式和 y 值
  return 0;
}
```

在 LoadRunner 中应用强制类型转换运算符的示例脚本代码如下：

```
Action()
{
    int x,y;                          //定义 2 个整形变量
    double pi,z;                      //定义 2 个双精度浮点变量

    pi=3.14;                          //对浮点变量 pi 赋值
    x=10;                             //对整型变量 x 赋值
    y=(int)pi;                        //将浮点数值强制转换为整型数值，赋值给 y
    z=(double)x;                      //将整型数值强制转换为浮点数值，赋值给 z

    lr_output_message("y=%d",y);      //输出 y 值
    lr_output_message("z=%f",z);      //输出 z 值
}
```

（4）自增、自减运算符。

在 LoadRunner 中应用自增、自减运算符的示例脚本代码如下：

```
Action()
{
    int i=10,j=10,k=10,l=10,m=10,n=10,g=10,h=10;

    lr_output_message("i=%d,i++=%d,i=%d",i,i++,i);
    lr_output_message("j=%d,++j=%d,j=%d",j,++j,j);
    lr_output_message("k=%d,k--=%d,k=%d",k,k--,k);
    lr_output_message("l=%d,--l=%d,l=%d",l,--l,l);
    lr_output_message("++m+5=%d",++m+5);
    lr_output_message("n+++5=%d",n+++5);
    lr_output_message("--g+5=%d",--g+5);
    lr_output_message("h--+5=%d",h--+5);
    return 0;
```

```
}
```

在 LoadRunner 中应用赋值运算符的示例脚本代码如下：

```
Action()
{
  int a = 10;           //整型变量赋值
  float x = 10.5;       //单精度浮点变量赋值
  double y = 3.14159;   //双精度浮点变量赋值
  char c = 'a';         //字符类型变量赋值
  int z = 20;           //整型变量赋值

  z += 2;               //复合赋值表达式

  lr_output_message("整型数字 a=%d", a);
  lr_output_message("单精度浮点类型数字 x=%.1f", x);
  lr_output_message("双精度浮点类型数字 y=%.5f", y);
  lr_output_message("字符类型数值 c=%c", c);
  lr_output_message("复合赋值表达式 z+=2 的运算结果为%d", z);
  return 0;
}
```

3．预处理

在 LoadRunner 中，大家经常会看到类似于下面的语句：

```
#include "web_api.h"

Action()
{
     return 0;
}
```

在 LoadRunner 中应用预处理的示例脚本代码如下：

```
#define PI 3.14159              //定义 PI 常量
#define MAX(a,b) (a>b)?a:b      //定义取两数较大值的函数

int min(int x,int y)            //取两个整数较小值的函数
{
   if (x<=y) return x;
   else return y;
}

Action()
{
    int a=10;
    int b=20;
    int z=MAX(10,20);
    int cc=min(10,20);

    lr_output_message("PI=%.5f",PI);
    lr_output_message("z=%d",z);
    lr_output_message("x=%d",cc);
```

```
    return 0;
}
```

“myfunccomm.h”文件的内容如下：

```
#define PI 3.14159                  //定义 PI 常量
#define MAX(a,b) (a>b)?a:b          //定义取两数较大值的函数

int min(int x,int y)                //取两个整数较小值的函数
{
  if (x<=y) return x;
  else return y;
}
```

为了实现和前面的脚本同样的功能，需要将刚才定义的“myfunccomm.h”引用到脚本中，在 LoadRunner 中应用预处理，引用文件的示例脚本代码如下：

```
#include <myfunccomm.h>

Action()
{
    int a=10;
    int b=20;
    int z=MAX(10,20);
    int cc=min(10,20);

    lr_output_message("PI=%.5f",PI);
    lr_output_message("z=%d",z);
    lr_output_message("x=%d",cc);

     return 0;
}
```

4．函数及其函数的参数

```
void SayHello() //打招呼的函数
{
  lr_output_message("Hello %s", lr_get_host_name());
}

int GetBigger(int x, int y) //得到较大值函数
{
  if (x > y)
  {
    return x;
  }
  else
  {
    return y;
  }
```

```
}

Action()
{
  int x = 10, y = 20, result; //声明部分

  SayHello(); //无形参，无返回值函数
  result = GetBigger(x, y); //带形参，带返回值函数
  lr_output_message("GetBigger(%d,%d)=%d", x, y, result);

  return 0;
}
```

5. 局部变量和全局变量

请参见 globals.h 头文件内容：

```
#ifndef _GLOBALS_H
#define _GLOBALS_H

//---------------------------------------------------------------------
// Include Files
#include "lrun.h"
#include "web_api.h"
#include "lrw_custom_body.h"

//---------------------------------------------------------------------
// Global Variables

int icount=10;//全局变量

#endif // _GLOBALS_H
```

Action 部分代码如下：

```
int a=8,b=15;     /*a,b 为全局变量*/

int max(int a,int b) /*a,b 为局部变量*/
{
     return a>b?a:b ;
}

int incb()
{
     ++b;
     lr_output_message("incb()函数中的 a=%d,b=%d",a,b);
}

Action()
{
     int a=10;//局部变量
     lr_output_message("a=%d,b=%d",a,b);
```

```
        incb();
        lr_output_message("a=%d,b=%d",a,b);
        lr_output_message("max(%d,%d)=%d",a,b,max(a,b));
        lr_output_message("globals.h 中的全局变量 icount=%d", icount);
        return 0;
    }
```