

Duer Builder 0.80 用户手册

一、文件概述

Duer Builder（以下简称“本产品”，原名“C-- builder”）是一个类 C 语言解释器，姑且称之为 Duer 语言（原名“C--”）。Duer 语言的语法充分参考了 C 语言。

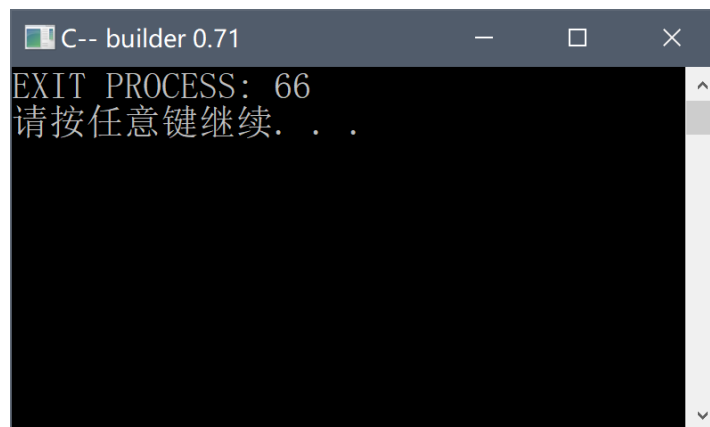
官方网站 <http://duer.org/>，本程序完全开源，且使用者享有任何权利。

本文件针对的是 0.80 版本的 Duer Builder。

二、某些解释

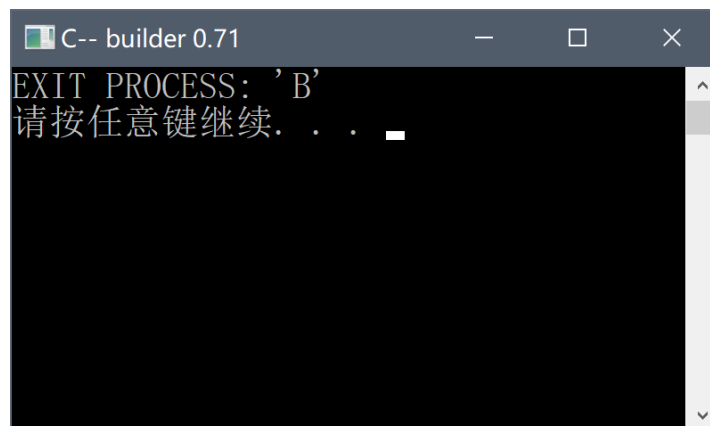
1、本产品会自动读取 D://1.txt 作为代码文件，如果找不到文件，则会寻找当前目录下的 1.txt；

2、本产品不支持预编译，因而导致运行库缺失，为了提供输出功能，特使 main 函数的返回值输出，例如：



```
C-- builder 0.71
EXIT PROCESS: 66
请按任意键继续. . .
```

EXIT PROCESS: 后接 main 函数的返回值，相应的，main 函数的返回值类型可以自由调整以输出不同格式的数据，例如：



```
C-- builder 0.71
EXIT PROCESS: 'B'
请按任意键继续. . .
```

是将 main 函数的返回类型修改为 char 的结果。

三、语法简介

1、变量

变量名的规则与 C 语言一致，此为强类型语言，定义时需要指定类型名，例如：

```
int iVal = 0;
bool bVal = 1;
```

定义时对变量进行赋值的行称为**初始化**，初始化与**赋值**有所区别，例如在定义 const 变量时，如果不对其进行初始化，则永远无法对其赋值，即该 const 变量无意义。每一条命令语句的结尾都需要以分号隔开，换行符、空格、制表符是无效的命令分隔符。

定义在所有函数外的变量称为**全局变量**，与 main 函数平行的函数称为**全局函数**，全局变量可以被任何函数在任何时候被调用，全局函数和全局变量在进入 main 函数之前被扫描完毕，所以不需要定义在 main 函数的前面（仅限全局函数和全局变量）。

基本类型为以下几种：

```
char bool int double
```

其可存放的数据宽度可以参考 c 语言。

任何自定义常整数的返回值均为 int，自定义常小数为 double，例如

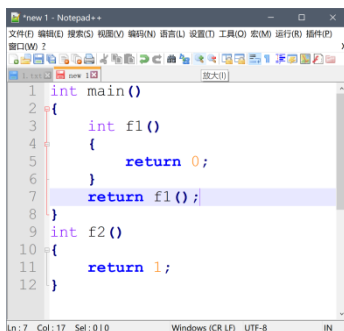
```
bool bVal = 2;
```

定义了一个 bool 类型的变量，右边的 2 返回的是一个 int 类型，之后通过**类型隐式转换**为 bVal 初始化，得到 bVal 的实际值为 1。

分别支持前置、后置的自增自减运算符，效果参考 C 语言。

2、函数

函数在本语言中可以嵌套出现，例如：



```
1 int main()
2 {
3     int f1()
4     {
5         return 0;
6     }
7     return f1();
8 }
9 int f2()
10 {
11     return 1;
12 }
```

其中，main 函数为主函数，f1 函数在 main 函数内部，称为 f1 属于 main 函数。返回值为 f1 返回的值。值得一提的是，任何非全局函数在使用前必须先定义。

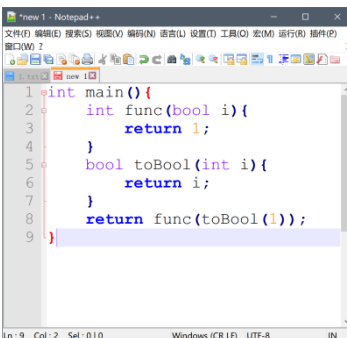
函数的查找通过函数签名，即函数参数的变量类型和个数，例如

```
int f(int value) 与
```

```
int f(bool value)
```

是不同的函数，需要不同的定义。

值得一提的是，若将非 int 作为参数，（例如 `int func(bool i)`），调用 `func(1)` 时会找不到函数，因为**参数不进行任何隐式类型转换**，但是可以作如图写法：



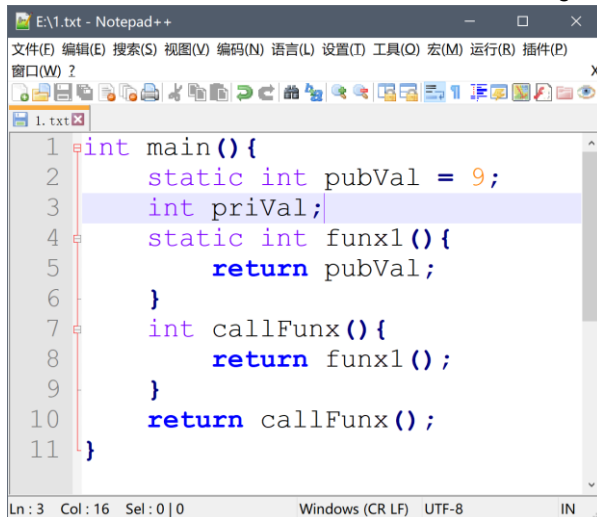
```
1 int main() {
2     int func(bool i) {
3         return 1;
4     }
5     bool toBool(int i) {
6         return i;
7     }
8     return func(toBool(1));
9 }
```

即，返回值不作限制。

另外，可以将**单个字符**作为 int 常量使用，例如 'A' 代表 65，'\n' 代表 10 等。

另外，平行函数不能相互调用，如左图，func 函数不能调用 toBool 函数，因为它们没有从属关系，函数作为一种特殊的作用域是相对独立的，函数只能使用属于它的函数或变量，以及全局函数或变量。

在函数内定义全局函数或变量，加上 static 即可。如：



```

1 int main() {
2     static int pubVal = 9;
3     int priVal;
4     static int funx1() {
5         return pubVal;
6     }
7     int callFunx() {
8         return funx1();
9     }
10    return callFunx();
11 }

```

funx1 是一个静态函数，pubVal 是一个全局变量，它们可以在任何地方被使用。

在私有变量（函数）的名称（或函数签名）与全局变量（函数）**重复**时，可以使用全局作用符号“::”代表全局环境，例如：

```
return ::funx1();
```

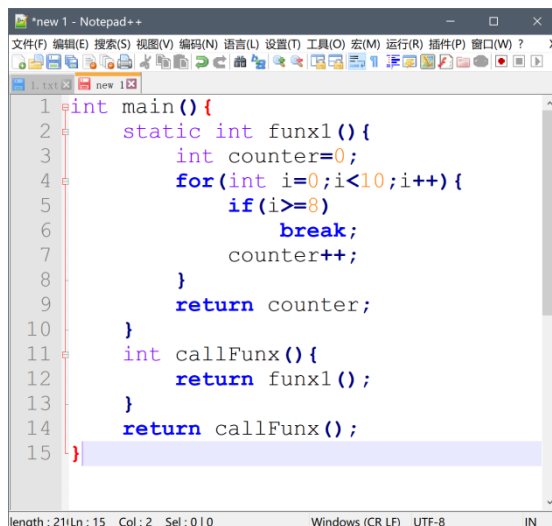
函数退出时，其内部的变量被销毁。

静态函数在调用前必须先定义，静态函数的地位与全局函数一致，其函数签名不可与全局函数重复。

3、流程语句

本语言提供 if 和 for 两种特殊语句，并对 for 提供 break 和 continue 控制语句，if 和 for 会创建一种异于函数的普通环境，即，**可以在函数范围内调用环境外的变量（函数），也可以使用全局变量（函数）。**

例如：



```

1 int main() {
2     static int funx1() {
3         int counter=0;
4         for(int i=0;i<10;i++){
5             if(i>=8)
6                 break;
7             counter++;
8         }
9         return counter;
10    }
11    int callFunx() {
12        return funx1();
13    }
14    return callFunx();
15 }

```

对于 if 和 for 语句来说，如果后面只跟一条代码，即空悬语句，则不需要大括号，左图中的 for 和 if 语法含义与 C 语言相同。If 语句可以使用 else if，语法与 C 语言相同。

目前不提供 while 语句和 do while 语句，也不提供 switch 语句。

4、特殊变量

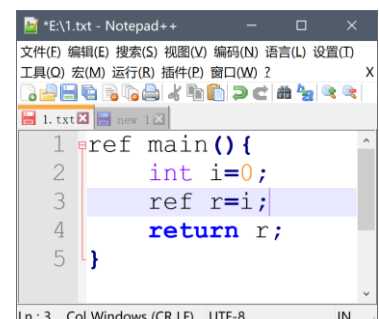
本语言不提供指针，替代品为引用，引用的定义语法如下：

```
ref <refname> = <valname>;
```

值得注意的是，引用并非绝对安全，**在函数返回时引用函数内的临时变量时不可行的**。引用不存在的变量会导致未知问题。

例如右图，这是一种典型的逻辑错误，将导致不可预料的问题。

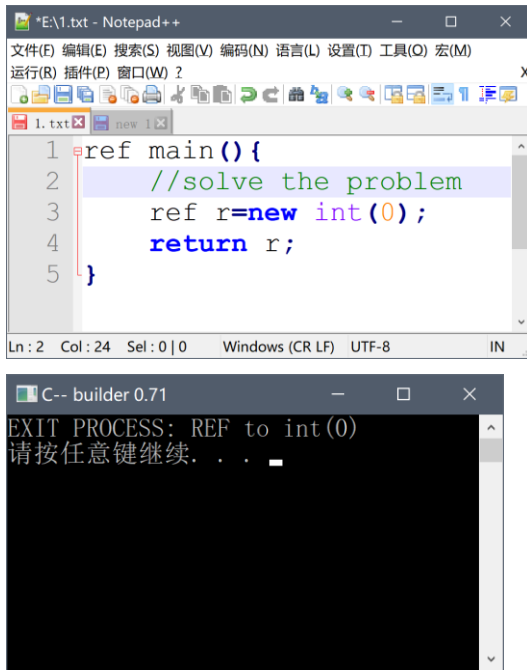
可以使用动态内存分配来解决这个问题，例如：



```

1 ref main() {
2     int i=0;
3     ref r=i;
4     return r;
5 }

```



这样程序就可以正常运行了。

r 指向堆上的空间，值得一提的是，我们不需要手动释放内存空间，堆上的空间会在程序退出时被自动释放。

另外，如果在 new 时不提供初始值（例如提供空括号或者不提供括号），则该变量会被自动初始化为 0。

上图程序的返回值如左图（main 函数的返回值为 ref）。

这代表返回了一个 ref，指向 int 类型，其值为 0。

5、特殊语句（0.75 新增）

我们有 require_once 和 require 语句，示例如下：

```
require_once "lib/typeConverts.hmm";
```

被包含的文件将被安装入当前环境。

6、String 变量（0.80 新增）

在 0.82 中引入 String 变量的部分功能，包括：

1) new 一个 string 变量：

```
ref r = new string("123");
```

2) String 变量与普通变量的转换和运算：

```
string p = "abc" + 3 + 3.14 + "abd" + 'a';
```

```
//output: abc33.14abda
```

```
string p = "abc" + (3 + 3.14) + "abd" + 'a';
```

```
//output: abc6.14abda
```

3) String 变量的逻辑运算（相等、不等、大于、大于等于，...）

7、标准库（0.80 新增）

1) lib/typeConverts.hmm

用于各种类型转换。

2) lib/stdio.hmm

用于输入输出，包含多种输入输出函数如 gets(), getline(), print(), println()等。

- 3) lib/devUtils.hmm
包含 pause 和 clear 函数。

四、结尾

以上就是对 Duer Builder 的简单描述。

官网 <http://duer.org/>。

本程序的作者为 Alvazu，任何问题或讨论请联系邮箱 me@mailwhat.com。