

MiniC 相关说明

1 概述

MiniC 是 C 语言的一个子集，对 C 语言语法进行了大量删减，以产生一种适用于编译实习课程的语言。

MiniC 是为了取代原先编译实习课程使用的 MiniJava 语言而设计的，目的是更好地配合编译原理课程进度，在一定程度上减轻任务量。MiniC 基础语法高度精炼，使同学们无论能力如何，都能完成编译器编写的过程。

同时，对于能力较强的同学，实习课程可以提供一些 MiniC 语法扩展内容，把从 C 语言中删除的一些语法加回 MiniC，来提升 MiniC 语言的易用程度和表达能力，对完成扩展内容的同学提供一定程度的加分。

2 语法描述

- MiniC 取消了 C 语言中的宏。
- MiniC 中的变量有两种类型，int 和一维 int 数组。
- MiniC 中函数返回值只有 int，参数可以是 int 或 int 数组，程序从 main 函数开始执行。同时，MiniC 不会给函数默认返回值，如果执行完一个函数而没有 return，会导致未知行为。
- 单目运算符有 '!' 和 '-'
- 双目运算符有 '+', '-', '*', '/', '%', '&&', '||', '<', '>', '==', '!='
- 合法的表达式参考 BNF。
- 允许使用函数前置声明（参见样例中 `getint` 函数）。
- MiniC 程序中允许 C 风格的单行注释。
- MiniC 只保留 `if-else` 条件分支语句和 `while` 循环语句。
- 为了使 MiniC 更容易实现，限定 `if, while` 后面的括号里和逻辑运算符两边的运算分量只会出现如下两种形式：
 `x > y || (a + b) != c` 这样的逻辑表达式；
 `x` 或 `f(x)` 或 `a[x]` 这样的单个变量或函数。
 总之，不会出现类似 `if (a + b)` 或 `(b + c) || d` 这样的语句。

3 BNF

```
 $\langle Goal \rangle ::= (\text{VarDefn} \mid \text{FuncDefn} \mid \text{FuncDecl})^* \text{MainFunc}$   
 $\langle VarDefn \rangle ::= \text{Type Identifier ';'}$   
                   $\mid \text{Type Identifier '[' } \langle INTEGER \rangle \text{' ;'}$   
 $\langle VarDecl \rangle ::= \text{Type Identifier}$   
                   $\mid \text{Type Identifier '[' } \langle INTEGER \rangle \text{' ?'}$   
 $\langle FuncDefn \rangle ::= \text{Type Identifier '(' ( VarDecl ( ',' VarDecl )^* )? ') ' '{' (FuncDecl}$   
                   $\mid \text{Statement})^* \text{'}$   
 $\langle FuncDecl \rangle ::= \text{Type Identifier '(' ( VarDecl ( ',' VarDecl )^* )? ') ';'}$   
 $\langle MainFunc \rangle ::= \text{'int' 'main' '(' ') ' '{' (FuncDecl } \mid \text{Statement})^* \text{'}$   
 $\langle Type \rangle ::= \text{'int'}$   
 $\langle Statement \rangle ::= \text{'{' (Statement)^* \text{'}}$   
                   $\mid \text{'if' '(' Expression ') ' Statement ('else' Statement)?}$   
                   $\mid \text{'while' '(' Expression ') ' Statement}$   
                   $\mid \text{Identifier '=' Expression ';'}$   
                   $\mid \text{Identifier '[' Expression ']' '=' Expression ';'}$   
                   $\mid \text{VarDefn}$   
                   $\mid \text{'return' Expression ';'}$   
 $\langle Expression \rangle ::= \text{Expression ( '+' } \mid \text{'-' } \mid \text{'*' } \mid \text{'/' } \mid \text{'%' ) Expression}$   
                   $\mid \text{Expression ( '\&\&' } \mid \text{'||' } \mid \text{'<' } \mid \text{'==' } \mid \text{'>' } \mid \text{'!=' ) Expression}$   
                   $\mid \text{Expression '[' Expression '] '}$   
                   $\mid \langle INTEGER \rangle$   
                   $\mid \text{Identifier}$   
                   $\mid \text{( '!' } \mid \text{'-' ) Expression}$   
                   $\mid \text{Identifier '(' (Identifier ( ',' Identifier )^* )? ') '}$   
                   $\mid \text{'(' Expression ') '}$   
 $\langle Identifier \rangle ::= \langle IDENTIFIER \rangle$ 
```

4 示例

```
1  int getint(); // 前置函数声明, getint 函数是 MiniC 内置函数, 返回一个读入的整数
2  int putchar(int c); // 内置函数, 用于输出字符 (参数为 ascii 码), 返回值无意义
3      // 注意! base 语法集不包括形如 int putchar(int); 这种参数名没有具体给出的函数声明
4  int putint(int i); // 内置函数, 用于输出整数, 返回值无意义
5  int getchar(); // 内置函数, 返回一个读入的字符的 ascii 码 (此程序未使用到该函数)
6  int f(int x) /* 该函数以递归方式计算 Fibonacci 数 */
7  {
8      if (x < 2) /* if-else 语句 */
9          return 1;
10     else
11         return f(x - 1) + f(x - 2); /* 递归函数调用 */
12 }
13 int g(int x) /* 该函数以数组和循环语句计算 Fibonacci 数 */
14 {
15     int a[40]; /* int 数组声明
16         // 注意! base 中数组大小必须是常数, 不可写成 int a[x]; 或 int a[10+30]; 这样 */
17     a[0] = a[1] = 1;
18     int i;
19     i = 2; /* 注意! base 语法集不包括初始化赋值语句 int i = 2; */
20     while (i < x + 1) /* while 循环是 base 语法集唯一的循环语句 */
21     {
22         a[i] = a[i - 1] + a[i - 2];
23         ++i;
24     }
25     return a[x];
26 }
27 int n; // 声明了一个全局变量
28 int main() {
29     n = getint();
30     if (n < 0 || n > 30) /* 不带 else 的 if 语句 */
31         return 1;
32     putint(f(n));
33     putchar(10); // 输出换行符
34     putint(g(n));
35     putchar(10);
36     return 0;
37 }
```

5 MiniC 语法扩展

MiniC 设计者们亲身实践了 MiniC 大部分语法扩展，深切体会到实现一些复杂语法扩展的不宜。

对 MiniC 语法进行扩展时，应尽量遵循“细致”的原则，避免涉及范围过大的扩展。

比如，整数数据类型扩展：加入 8 位，16 位，32 位，64 位的有符号和无符号整数。这个扩展涉及的范围就有些大，可以考虑分解成多个扩展：带符号整数扩展、不同长度的整数运算扩展、char 字符串扩展（8 位有符号整数组成的数组）、字符与字符串表示扩展（加入 "abc", '\n' 这样的表达式）。

具体可以扩展内容，按教学通知为准。