Amirkabir University of Technology
Computer Engineering and Information Technology Department
Compiler Course Winter 2006

This Document is the specification of 3KP language. It might evolve over the course of the semester. The date of each definition released is attached, so please refer to the latest version available.

Lexical Consideration

1. The following terms are keywords. They are *reserved* which means they cannot be used as identifiers. ***char, int, double, program, bool, for, to, down, while, if, then, else, switch, of, break, repeat, continue, return, write, read, case, true, false, in, end***.
2. Every ID and keyword in this language is case-sensitive. Just like C++, so all keywords are lowercase.
3. Whitespaces (i.e. spaces, tabs, new line) are utilized to separate tokens.
4. An Identifier is a sequence of letters, digits and underscores, starting with an underscore or a letter.
5. All numbers are in base 10. There are two types of number: integers or decimals. all numbers may have a sign symbol prefix (+ , -). decimal numbers can be in form of .12 ,0.12, 12. 12E2 12e-2 and so on.
6. We have comments in this language. comments starts with /* and ends with */. any symbol is allowed in comment section except */, which ends the current comment. Comments do not nest in each other.
7. STRING is a constant string enclosed in double quotation mark.
8. CHAR is a single character enclosed in a single quotation mark.

Semantics

1. All parameters should be called by value. We don't have any pointer or reference in this language.
2. int type is a signed 32-bit number. bool is an 8-bit number. double is 32-bit floating point number.
3. We don't have any void type in this language. so all function should have a return value.

4. It is not allowed to use arrays as function parameters. If we declare an array with n cell, (i.e. bool temp[100] where n would be 100) then all indices to this array should be in range [0,99].
5. All array sizes should be positive integer and constant. So we don't have any array declarations like int temp[10*2+1].
6. Logical "and" and "or" should be interpreted using short-circuit evaluation.

IN THIS PROJECT YOU CAN ALWAYS ASSUME THAT THE INPUT SOURCE CODE IS CORRECT. We don't want you to have error detection and error recovery in your source codes.

Grammar Definition

<Program> → program ID ';' <DecList> '{' '}' '.' OK
<DecList> → ε | <Dec> <DecList> OK
<Dec> → <VarDecs> | <FuncDecs> OK
<FuncDecs> → ε | <FuncDec> <FuncDecs> OK
<VarDecs> → <VarDec> | <VarDec> <VarDecs> OK
<VarDec> → <Type> <IDDList> ';' OK
<Type> → int | double | bool | char OK
<IDDim>→ ID| <IDDim> '['<IntNumber>']' OK
<IDDList> → <IDDim> | <IDDim> ',' <IDDList> OK
<IDList> → ID | ID ',' <IDList> OK
<FuncDec> →<Type> ID '(' <ArgsList> ')' '{' <SList> '}' ';' OK
<ArgsList> → ε | <ArgList> OK
<ArgList> → <Arg> | <Arg> ';' <ArgList> OK
<Arg> → <Type> <IDList> OK
<SList> → ε | <Stmt> ';' <Slist> OK
<Stmt> → ε |
<Exp> |
<VarDecs> |
for <lvalue> '=' <Exp> '(' to | down to ')' <Exp> do <Block> |
while <Exp> do <Block> |
if <Exp> then <Block> |
if <Exp> then <Block> else <Block> |
switch <Exp> of '{' <Cases> '}' |
break |
repeat <Block> until <Exp> |
continue |
return <Exp> |
write <ExpPlus> |
read '(' <lvalue> ')'
<Range> →<Exp> '..' <Exp>
<Cases> →<Case> | <Case> <Cases> '.' end

<Case> → case <Exp> ‘:’ <Block> | case <Range> ‘:’ <Block>
<Logic> →’&&’ | ‘||’ | ‘<’ | ‘>’ | ‘>=’ | ‘<=’ | ‘==’ | ‘!=’
<Aop> → ‘+’ | ‘-‘ | ‘*’ | ‘/’ | ‘%’
<ExpList> → ε | <ExpPlus>
<ExpPlus> →<Exp>| <Exp> ',' <ExpPlus>
<IDD> →ID | <IDD> ‘[‘ <Exp> ‘]’
<lvalue> → ID | <IDD>
<Exp> →IntNumber |
<lvalue>
RealNumber |
CHAR |
true |
false |
<Exp> <Aop> <Exp> |
<Exp> <Logic> <Exp>|
‘-‘ <Exp> |
STRING |
‘(‘<Exp>’)’ |
<Exp> in <Range> |
<lvalue> ‘=’ <Exp> |
ID’(‘<ExpList>’)’
<Block> → ‘{‘ <Slist> ‘}’ | <Stmt>

Operators Precedence

Operator precedence is as follows from highest to lowest

1. [ ] (array indexing)
2. (unary minus)
3. / % (multiply, divide, mod)
4. + - (addition, subtraction)
5. < <= >= > (relational)
6. == != (equality)
7. && (logical and)
8. || (logical or)
9. = (assignment)

Type Casting

Everything can cast to int (e.g. char cast to int with its value). If you want to cast a Boolean expression into an integers as you know true value is 1 and false is 0, so expression like 1+(3<20) will be evaluated to 2.

Special Commands

write and **read** are special commands. **read** can read a single value from input, you should implement it with scanf command. But **write** command can have many arguments and should be implemented with printf command. Also **in** is another special command in which return true if its expression value is between the ranges inclusive.

Control Structures

- an **else** clause always joins the closest unclosed if statement.
- **break** statement can only appear within a **while, for, repeat** loop.

all operators are left-associative except "=" which is right associative.

Scope

This language supports several levels of scoping. A declaration at top level is placed in the global scope. Each function declaration has its own local scope. And each variable defined in each block will be seen in inner blocks.

Sample Programs

```
program Dfs;
int n,m,mat[100][100],mark[100];
int dfs(int a)
{
        int i,j;
        mark[a]=1;
        for i=0 to n do
                if (mark[i]==0 && mat[a][i]) then
                        dfs(i);
        return mark[a];
};
{
        int n,m,i,j,c;
        c=0;
        read(n);
        for i=0 to n do
                for j=0 to i-1 do
                        read(mat[i][j]);
        for i=0 to n do
                mark[i]=0;
        for i=0 to n do
                if (mark[i]==0) then
                {
                        dfs(i);
                        c=c+1;
                }
        print(c);
}.
program gcdDemonestration;
int gcd(int a,b)
{
        if(a*b==0) return a+b;
        return gcd(b,a%b);
```

```
};
{
        int a,b;
        while(1) do
        {
                read(a);
                read(b);
                print(gcd(a,b));
        }
}.
```

If you think this grammar has vagueness or confliction, please tell me.