

STATEMENTS

BCPL is not case sensitive. The reserved words of the language (*if*, *unless*, *while*, etc) may be typed in any combination of capital and lowercase letters.

Assignment

destination := *expression*

destination operator expression for any dyadic operator, equivalent to
destination := (*destination*) *operator expression*

Destinations for assignment, called L-values, may be:

The name of any variable

! *E*, where *E* is any expression - see the section on expressions

E ! F, where *E* and *F* are any expressions - see the section on expressions

E from *F* - see the section on expressions

E of *F* - see the section on expressions

Conditional

if expression then statement *do* may be used in place of *then*

unless expression do statement \equiv *if not (expression) do statement*

test expression then statement or statement
do may be used in place of *then*
else may be used in place of *or*

In conditional and loop expressions, 0 is taken as false, all other values are taken as true.

Loops

while expression do statement

until expression do statement \equiv *while not (expression) do statement*

statement repeat \equiv *while true do statement*

statement repeatwhile expression the statement is always executed once before the condition is ever tested.

statement repeatuntil expression \equiv *statement repeatwhile not (expression)*

`for variable = expression to expression do statement`

`variable` is declared as local to the loop.

程序代写代做 CS编程辅导

The `to expression` is evaluated only once, before the loop begins.

If the loop is to count down, “`by -1`” must be stated explicitly

`for variable = expression by expression do statement`

the `by expression` must be a compile-time constant

`break`

same as in C++

`loop`

\equiv C++’s `continue`



Function call

`expression ()`

`expression (expression)`

`expression (expression , expression)`

`expression (expression , expression , expression)`

`expression (expression , expression , expression , expression)`

etc, etc, etc.

`expression () := expression`

`expression (expression) := expression`

`expression (expression , expression) := expression`

`expression (expression , expression , expression) := expression`

etc, etc, etc.

The expressions are all parameters, including the one following `:=`.

Any number of parameters may be provided, regardless of the number specified in the function’s declaration. No check is performed.

If the function returns a result (i.e. uses `resultis` instead of `return`), there is no error, the result is ignored.

If the `:=` versions are used, the expression `lhs ()` will be true inside the function.

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

https://tutorcs.com

Exiting

`return`

exits from “void” function

`resultis expression`

exits from non-void function or a `valof`

`finish`

terminates the whole program

Blocks

```
{ } // do nothing
{ statement }
{ statement ; statement }
{ statement ; statement }
{ statement ; statement ; statement } // etc, etc, etc.
```

All labels must appear before the first regular statement.
Separate statements, they are not parts of those statements.
They are required immediately after a } symbol.

程序代写代做 CS编程辅导



Jumps

`goto expression` // *expression* should evaluate to a label

`label : statement` // or

`label : }` // attaches a label to a goto destination
labels are just names, same rule as variables
labels are local to the block they appear in

WeChat: cstutorcs

Assignment Project Exam Help

Multi-way jump

Email: tutorcs@163.com

`switchon expression into statement` // *statement* should be a block containing case statements and optionally a default

`case expression : statement`

`case expression ... expression : statement`

`default : statement`

`default expression ... expression : statement`

https://tutorcs.com

in case and default statements the expressions must be compile-time constants.

`endcase`

≡ C++'s `break` in a `switch`

On executing `switchon E into S`, the expression *E* is evaluated and a jump is made directly to the statement inside *S* (which should be a block) that has the matching case label.

If there is no case statement matching the expression, the jump is to the statement with the `default` label. If there is no default statement, the whole `switchon` has no effect. If the default statement has a range of expressions and the `switchon` expression is outside that range, again the whole `switchon` has no effect.

Once a case or default has been jumped to, the following statements are also executed until the end of the `switchon` is reached, even if other case statements are reached.

Miscellaneous

`debug expression`

程序代写代做 CS编程辅导

causes an interruption to program execution and transfer of control to the assembly language debugger.



DECLARATIONS

Names chosen for variables and functions must begin with a letter and may contain any combination of letters, numbers, and dots. Capital and lower case letters are the same: Number, number, and underscore are the same thing.

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
a b c d e f g h i j k l m n o p q r s t u v w x y z
0 1 2 3 4 5 6 7 8 9

WeChat: cstutorcs

Local Variables

inside any block, before the first regular statement.

Assignment Project Exam Help

`let item , item , item , item ...`

items may be:

Email: tutorcs@163.com

variable = *expression*

variable = *vec expression*

The third form creates an array of the indicated size, and sets the value of the new variable to be the address of the first element of the array. The expression used to indicate the array's size must be a compile-time constant. The array has the same lifetime as the variable.

QQ: 749389476

https://tutorcs.com

Very Local Variables

statement where *item* , *item* , *item* , *item* ...

Any statement may be followed by a *where* clause, which introduces new variables that are local to that single statement, for example:

`y := t * (t+1) where t = 3*x+5`

Global Variables

May be declared using the same syntax as local variables but outside of any function.

Static Variables

```
static { item item item, ... }
```

May be global (in which case they are exactly the same as global variables) or local to any block. A local static variable has the same lifetime as a global variable, it will retain its value from one execution to the next.



Named Constants

```
manifest { name = value, name = value, ... }
```

May be global or local. The values must be compile-time constants.

WeChat: cstutorcs

Functions

```
let name ( parameters ) be statement for “Void” functions
```

```
let name ( parameters ) = expression for functions that return a value
```

parameters is a list of names separated by commas. If there are no parameters, an empty pair of parentheses is still required. Normally the *statement* will be a block enclosed in { }, but that is not necessary.

The function may be called with any number of parameters, regardless of the number appearing in the declaration. The standard library function `numargs()` may be used inside the function at run time to find out how many were actually passed in.

Functions may be global as in C++, but they may also be declared local to another function or block.

Simultaneous Declaration of Functions

```
let name ( parameters ) be statement  
and name ( parameters ) be statement  
and name ( parameters ) be statement ...
```

The “= *expression*” form may be used instead of the “be *statement*” form shown. All of the functions are declared before any of the statements are processed by the compiler, so the later functions may refer to the earlier ones without violating the declare-before-use rule. This is necessary to allow mutually recursive functions.

COMMENTS

Comments are exactly the same as in C.
after `//` ignore everything up to the end of the line,
after `/*` ignore everything up to and including the next `*/`.



PROGRAMS

A BCPL program consists of declarations of functions, named constants, and global variables. To be a valid program must have a function called `start`, which serves the same purpose as `main` in C.

Amongst the declarations in a program, there may also be directives to make use of pre-defined libraries, these have the form

```
import "name"
```

The standard library of input/output functions is accessed through `import "io"`

EXPRESSIONS, in order of decreasing priority

1. Atomic expressions

identifier

The name of any variable, manifest constant, or function.

The value of a function name is the address of the memory location that contains its first executable instruction.

integer constants

may be written in:

decimal, using only the digits 0 to 9

binary, being prefixed by `0b` and using only the digits 0 to 1

example: `0b1001` is equivalent to 9.

octal, being prefixed by `0o` and using only the digits 0 to 7

example: `0o123` is equivalent to 83.

hexadecimal, being prefixed by `0x` and using only the digits

0 to 9, A to F, and a to f

example: `0x1A2` is equivalent to 418.

floating point constants

may be written only in decimal, using the digits 0 to 9, and must contain either a decimal point, or use the E notation to represent “times ten to the power of”, or both. Examples:

1.0

123.4

0.157

.98765

1e3 is equivalent to 1000.0

程序代写代做 CS编程辅导

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

https://tutorcs.com

1.23E3 is equivalent to 1230.0

1.234567e+1 is equivalent to 12.34567

1.23e-12 is equivalent to 0.00000000000123

Beware, a preceding + or - sign is not considered to be part of the number, but the use of a separate monadic operator. To produce a valid negative floating point number, the floating point #- operator must be

string constants



number of characters except line breaks, surrounded by Special characters are represented in the C style, using notations beginning with \:

\a represents a single \, ascii code 92.

\' represents a single ', ascii code 39.

\\" represents a single ", ascii code 34.

\n represents a new-line character, ascii code 10.

\r represents a carriage return, ascii code 13.

\t represents a tab, ascii code 9.

\b represents a backspace character, ascii code 8.

\s represents a normal space, it is just a way to make them visible.

\nnn represents the character with decimal ascii code nnn

The value of a string constant is the address in memory where the characters are stored, terminated with a NUL character (ascii code 0), and packed four to each word.

character constants

consist of one to four characters surrounded by single 'quotes'. The same \ notation described for strings is used for special characters. The value of a single quoted character (e.g. 'x') is its ascii code. Multiple quoted characters are packed into the four 8 bit bytes that fit in a 32 bit word.

parenthesised expressions

Any valid expression may be surrounded by brackets in the usual way. Round brackets (2+3) and square brackets [2+3] may both be used, and have the same meaning, but they must be correctly matched.

valof block

Any normal block of code (declarations and statements surrounded by {}) may be converted into an expression by valof. The result is statement is used to indicate the final value. Example:

```
x := y + valof { let f = 1;
                  while n > 0 do
                  { f := n;
                    if f>1000 then resultis f;
                    n := 1 }
                  resultis f/2 } * 7
```

true

Equivalent to -1, a 32 bit value in which every bit is set to 1.

false

Equivalent to 0.

nil

Equivalent to 0. Intended to represent a null pointer.

2. Function calls

程序代写代做 CS编程辅导

expression ()

expression (*expression*)

expression (*exp*  *n*)

expression (*exp* *n* , *expression*)

expression (*exp* *n* , *expression* , *expression*)

etc, etc, etc.

If the function *expression* is not a function (i.e. uses `return` instead of `result`), there is no error, *expression* is indeterminate.

3. Monadic operators

E represents any valid expression

+ *E*

Has no effect

- *E*

Integer negation. The value is $0 - E$.

#- *E*

Floating point negation. The value is $0.0 - E$.

not *E*

~ *E*

Logical negation. If $E=0$ the result is -1 ; if $E \neq 0$, the result is 0 .

bitnot *E*

Bit by-bit negation. Each of the 32 bits of *E* is switched over 0 to 1 and 1 to 0.

! *E*

Pointer following. The value of the expression is whatever is stored in memory location *E*.

This form is also allowed as an L-value in an assignment statement:

! E := F

changes the value stored in memory location *E* to become *F*.

abs *E*

Integer absolute value. If *E* is positive, the value is *E*. If *E* is negative, the value is $-E$.

#abs *E*

Floating point absolute value. If *E* is positive, the value is *E*. If *E* is negative, the value is $\#-E$.

fix *E*

Conversion from floating point to integer. The result is truncated towards zero.

float *E*

Conversion from integer to floating point.

@ *E*

Address of. *E* must be a variable, the result is the numeric address of the memory location used to store *E*'s value. *! @ E* is the same as just *E*.

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

https://tutorcs.com

程序代写代做 CS编程辅导

4. Infix function call

A, B represent any valid expression

$A \% F B$



F is a function.

\times is equivalent to `fun(x, 24)`

5. Array and structure

A, B represent any valid expression

$A ! B$

$A ! B$ is exactly equivalent to `!(A+B)`

Either A or B should be the address of the first memory location in a vector (array or data structure), and the other should be an integer index into that vector. The result is the value stored at that position in the vector.

This form is also allowed as an L-value in an assignment statement:

$A ! B = C$ changes the value stored at that position in the vector to C .

6. To the power of

Email: tutores@163.com

$A ** B$

Integer to integer power, integer result.

$A \# ** B$

Floating point to integer power, floating point result.

7. Multiplication-like operators

A, B represent any valid expression

$A * B$

Integer multiplication, meaningless results if either operand is floating point.

$A \# * B$

Floating point multiplication, meaningless results if either operand is integer.

$A \# \# * B$

Unsigned integer multiplication.

A / B

Integer division, meaningless results if either operand is floating point.

$A \# / B$

Floating point division, meaningless results if either operand is integer.

$A \# \# / B$

Unsigned integer division.

$A \text{ rem } B$

Both operands should be integer values. Remainder after division.

$A \# \# \text{ rem } B$

Remainder after unsigned integer division.

8. Addition-like operators

A, B represent any valid expression

$A + B$

程序代写代做 CS编程辅导

Integer addition, meaningless results if either operand is floating point.

$A \# + B$

Fl meaningless results if either operand is integer.

$A - B$

Int meaningless results if either operand is floating point.

$A \# - B$

Fl on, meaningless results if either operand is integer.



9. Selectors, and

10. Bit range selection

A, B, C, D, E represent any valid expression

selector $A : B$

Describes the group of A consecutive bits that have B bits to the right of them within any 32 bit word

C from D

If C is a selector value and D is any expression, is the group of bits described by C extracted from the value of D , shifted right. Example:

selector 10 : 4 from 0x1B4693A5

0x1B4693A5 in binary is

00011011010001101001001110100101
^^^^^^^^^^

selector 10 : 4 describes these ten bits, so

selector 10 : 4 from 0x1B4693A5 is 0100111010 in binary, or 0x13A, or 314 in decimal.

From expressions also allowed as L-values in assignment statements:

selector 16 : 8 from $x := 0$

sets the middle 16 bits of the variable x all to zero.

selector $A : B : C$

Describes the group of A consecutive bits that have B bits to the right of them in the C^{th} word of any vector.

D of E

If D is a selector value and E is a pointer to a vector, is the group of bits described by D extracted from the indicated element of E . Example:

manifest { those = selector 16 : 16 : 2 }

let them = vec 4;

$v ! 0 := 0x13578642;$

$v ! 1 := 0xBEEFFACE;$

$v ! 2 := 0x1A2B3C4D;$

$v ! 3 := 0xE8500C2A;$

those of them := 0xAAAA;

changes $v!2$ to 0xAAAA3C4D.

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutors@163.com

QQ: 749389476

https://tutors.com

byte A

Is a selector for the A^{th} 8-bit byte in a vector an abbreviated form to simplify string processing.

byte $A \equiv \text{selector } 8 : (A \text{ rem } 4) * 8 : A / 4$

by "FGHIJ" has the value 'F'
by 'F' changes the i^{th} character of string s .



11. Shift operators

, B represent any valid expression

$A \ll B$

Left shift. The 32 bits of A are shifted B positions to the left.
The B most significant bits are lost, and B zero bits are added at the right.

$A \gg B$

Right shift. The 32 bits of A are shifted B positions to the right.
The B least significant bits are lost, and B zero bits are added at the left.

$A \text{ alshift } B$

Arithmetic left shift. Exactly the same as $A \ll B$.

$A \text{ arshift } B$

Arithmetic right shift. The 32 bits of A are shifted B positions to the right.
The B least significant bits are lost, but the sign of A is preserved.
If A was positive, then B zero bits are added at the left.
If A was negative, then B one bits are added at the left.
 $A \text{ arshift } B$ has the value of A divided by two to the power of B .

$A \text{ rotl } B$

Rotate left. The 32 bits of A are shifted B positions to the left.
The B most significant bits of A are removed from the left, but fed in from the right, so no bits are lost.

$A \text{ rotr } B$

Rotate right. The 32 bits of A are shifted B positions to the right.
The B least significant bits of A are removed from the right, but fed in from the left, so no bits are lost.

$0 \times 12345678 \text{ rotl } 16 = 0 \times 56781234$

12. Relational operators

A, B represent any valid expression

Chains of comparisons have the mathematically expected meaning, so for example

$A < B < C < D$ is the same as $A < B \wedge B < C \wedge C < D$

Integer comparisons - results are indeterminate if any operand is floating point.

$A < B$

True if A is less than B , false otherwise

$A \leq B$

True if A is less than or equal to B , false otherwise

$A > B$

True if A is greater than B , false otherwise

$A >= B$

True if A is greater than or equal to B , false otherwise

$A = B$

True if A is equal to B , false otherwise

$A <> B$

$A \neq B$

$A \backslash= B$

True if A is not equal to B , false otherwise



Floating point comparisons are indeterminate if any operand is not floating point.

$A \#< B$

True if A is less than B , false otherwise

$A \#<= B$

True if A is less than or equal to B , false otherwise

$A \#> B$

True if A is greater than B , false otherwise

$A \#>= B$

True if A is greater than or equal to B , false otherwise

$A \#= B$

True if A is equal to B , false otherwise

$A \#<> B$

$A \# \neq B$

$A \# \backslash= B$

True if A is not equal to B , false otherwise

Unsigned integer comparisons / results are indeterminate if any operand is floating point.

$A \#\#< B$

True if A is less than B , false otherwise

$A \#\#<= B$

True if A is less than or equal to B , false otherwise

$A \#\#> B$

True if A is greater than B , false otherwise

$A \#\#>= B$

True if A is greater than or equal to B , false otherwise

$A \#\#= B$

True if A is equal to B , false otherwise

$A \#\#<> B$

$A \#\# \neq B$

$A \#\# \backslash= B$

True if A is not equal to B , false otherwise

$A \wedge B$

Logical and, with “short circuit” evaluation.

If A is zero (i.e. false), then B is not evaluated, the result is 0.

Otherwise, if B is zero (i.e. false), the result is 0.

Otherwise, if A and B are both non-zero, the result is true (i.e. -1).

$A \text{ bitand } B$

Bit-by-bit and. The 32 bits of A and B are individually anded together, producing a 32 bit result.



13. Disjunctions

A, B represent any valid expression

$A \vee B$

Logical or, with “short circuit” evaluation.

If A is not zero (i.e. true), then B is not evaluated, the result is true (i.e. -1).

Otherwise, if B is not zero, the result is true.

Otherwise, if A and B are both zero, the result is false (i.e. 0).

$A \text{ bitor } B$

Bit-by-bit or. The 32 bits of A and B are individually orred together, producing a 32 bit result.

14. Equivalence and exclusive or A, B represent any valid expression

$A \text{ eqv } B$

The 32 bits of A and B are compared one-by-one to produce a 32 bit result.

Where a bit in A has the same value as the corresponding bit in B , there is a 1 in the result, where the bits are not the same there is a 0.

$A \text{ neqv } B$

Exclusive or. $A \text{ neqv } B \equiv \text{not } (A \text{ eqv } B)$

The 32 bits of A and B are compared one-by-one to produce a 32 bit result.

Where a bit in A has a different value as the corresponding bit in B , there is a 1 in the result, where the bits are equal there is a 0.

15. Conditional expressions

A, B, C represent any valid expression

$A \rightarrow B, C$

A is evaluated first. If the value of A is zero (i.e. false), C is evaluated and is the result. In this case B is never evaluated.

If the value of A is true (i.e. not zero), B is evaluated and is the result. In this case C is never evaluated.

16. Tables

A, B, C, D, E represent any valid expression

table $A, B, C, D, E, F \dots$

All of the expressions must be compile-time constants or strings or other table expressions. A table expression is equivalent to a global variable whose value is `vec N` (where N is the number of expressions), and the contents of the vector are set to the values of the expressions before program execution begins. For example:

```
let days = table "Mon", "Tue", "Wed", "Thur",  
                "Fri", "Sat", "Sun";
```

```
let  
gives the value "Thur"
```



THE STANDARD LIBRARY

`outch (N)`

print a single character

`outno (N)`

WeChat: cstutorcs

print an integer in decimal

`outhex (N)`

Assignment Project Exam Help

print an integer in hexadecimal

`outbin (N)`

Email: tutores@163.com

print an integer in binary

`outf (N)`

QQ: 749389476

print a floating point number

`outs (N)`

print a string

`outsv (N)`

https://tutorcs.com

print a string so that even invisible characters are visible

`out (F, A, B, C, ...)`

formatted output

The first parameter should be a string. The string is printed character by character until a '%' is encountered, then the next unused parameter is taken and printed using the function indicated by the character immediately following the %, as follows:

- d - `outno`
- f - `outf`
- s - `outs`
- v - `outsv`
- c - `outch`
- x - `outhex`
- b - `outbin`

example: `out("int %d float %f char %c\n", 12, 3.5, 65)`

prints int 12 float 3.5 char A

`inch()`

read a single character from the user, return its ascii code.

`inno()`

read a decimal integer and return its value

`init(V, N)`



is its length in words.
memory allocation system.

`newvec(N)`

initial words of memory from the vector given to init,
to the first

`freevec(V)`

de-allocate memory previously obtained from newvec.

`numbargs()`

returns the number of parameters that the surrounding function was
called with.

`lhs()`

true if the surrounded function was called from the left hand side of
an assignment statement.

`thiscall()`

returns a reference to the currently executing function call,
for use with `returnto`.

`returnto(R)`

`returnto(R, V)`

return to R, a function call reference previously obtained from `thiscall`,
terminating all intervening function calls still executing.
V is the value returned to that context.

`seconds()`

returns the number of seconds elapsed since midnight on 1st January
2000, local time, as an integer.

`datetime(T, V)`

T should be a time value as returned by `seconds`, V should be a vector
of at least 7 words. The time value is decoded and stored in V thus:

V ! 0 := year
V ! 1 := month, 1-12
V ! 2 := day, 1-31
V ! 3 := day of week, 0 = Sunday
V ! 4 := hour, 0-23
V ! 5 := minute, 0-59
V ! 6 := second, 0-59

程序代写代做 CS编程辅导

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutores@163.com

QQ: 749389476

https://tutorcs.com

`datetime2(V)`

程序代写代做 CS编程辅导
Similar to `datetime`, but a more compressed representation, and more precision in the time. `V` should be a vector of at least 2 words.

`V ! 0` : most significant 13 bits = year
next 4 bits = month
next 5 bits = day
next 3 bits = day of week
least significant 7 bits not used
most significant 5 bits = hour
next 6 bits = minute
next 6 bits = seconds
next 10 bits = milliseconds
least significant 5 bits not used



WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

<https://tutorcs.com>