Asymptote Reference Card

Program structure/functions

import "filename"
import "filename" as name
include "filename"

```
type f(type arg,
    ...)
    {
        statements
        return value;
    }
```

Data types/declarations

<table>
<thead>
<tr>
<th>Data types/declarations</th>
<th>Data types/declarations</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean (true or false)</td>
<td>bool</td>
</tr>
<tr>
<td>tri-state boolean (true, default, or false)</td>
<td>bool3</td>
</tr>
<tr>
<td>integer</td>
<td>int</td>
</tr>
<tr>
<td>float (double precision)</td>
<td>real</td>
</tr>
<tr>
<td>ordered pair (complex number)</td>
<td>pair</td>
</tr>
<tr>
<td>character string</td>
<td>string</td>
</tr>
<tr>
<td>fixed piecewise cubic Bezier spline</td>
<td>path</td>
</tr>
<tr>
<td>unresolved piecewise cubic Bezier spline</td>
<td>guide</td>
</tr>
<tr>
<td>color, line type/width/cap, font, fill rule</td>
<td>pen</td>
</tr>
<tr>
<td>label with position, alignment, pen attributes</td>
<td>Label</td>
</tr>
<tr>
<td>drawing canvas</td>
<td>picture</td>
</tr>
<tr>
<td>affine transform</td>
<td>transform</td>
</tr>
<tr>
<td>constant (unchanging) value</td>
<td>const</td>
</tr>
<tr>
<td>allocate in higher scope</td>
<td>static</td>
</tr>
<tr>
<td>no value</td>
<td>void</td>
</tr>
<tr>
<td>inhibit implicit argument casting</td>
<td>explicit</td>
</tr>
<tr>
<td>structure</td>
<td>struct</td>
</tr>
<tr>
<td>create name by data type</td>
<td>typedef type name</td>
</tr>
</tbody>
</table>

3D data types (import three)

ordered triple
3D path
3D guide
3D affine transform

Constants

<table>
<thead>
<tr>
<th>Constants</th>
<th>6.02e23</th>
</tr>
</thead>
<tbody>
<tr>
<td>exponential form</td>
<td>&quot;abc\ldots\de&quot;</td>
</tr>
<tr>
<td>\TeX{} string constant</td>
<td>&lt;,&gt;</td>
</tr>
<tr>
<td>\TeX{} strings: special characters</td>
<td>'abc\ldots\de'</td>
</tr>
<tr>
<td>C strings: constant</td>
<td>&lt;,&gt; ^' ?</td>
</tr>
<tr>
<td>C strings: special characters</td>
<td>&lt;,&gt; ^' ?</td>
</tr>
<tr>
<td>C strings: newline, cr, tab, backspace</td>
<td>\n \r \t \b</td>
</tr>
<tr>
<td>C strings: octal, hexadecimal bytes</td>
<td>0\ldots377 \x0\ldots\xFF</td>
</tr>
</tbody>
</table>

Operators

<table>
<thead>
<tr>
<th>Operators</th>
<th>+ - * /</th>
</tr>
</thead>
<tbody>
<tr>
<td>arithmetic operations</td>
<td>%</td>
</tr>
<tr>
<td>modulus (remainder)</td>
<td>== != &gt; &gt;= &lt; &lt;=</td>
</tr>
<tr>
<td>comparisons</td>
<td>!</td>
</tr>
<tr>
<td>not and or (conditional evaluation of RHS)</td>
<td>&amp;&amp;</td>
</tr>
<tr>
<td>and or xor</td>
<td>(type) expr</td>
</tr>
<tr>
<td>cast expression to type</td>
<td>++ --</td>
</tr>
<tr>
<td>increment decrement prefix operators</td>
<td>expr1 ? expr2 : expr3</td>
</tr>
<tr>
<td>assignment operators</td>
<td>name.member</td>
</tr>
</tbody>
</table>

Flow control

<table>
<thead>
<tr>
<th>Flow control</th>
<th>;</th>
</tr>
</thead>
<tbody>
<tr>
<td>statement terminator</td>
<td>(</td>
</tr>
<tr>
<td>block delimiters</td>
<td>*/ <em>/</em></td>
</tr>
<tr>
<td>comment delimiters</td>
<td>//</td>
</tr>
<tr>
<td>comment to end of line delimiter</td>
<td>break;</td>
</tr>
<tr>
<td>exit from while/do/for</td>
<td>continue;</td>
</tr>
<tr>
<td>next iteration of while/do/for</td>
<td>return expr;</td>
</tr>
<tr>
<td>return value from function</td>
<td>exit();</td>
</tr>
<tr>
<td>terminate execution</td>
<td>abort(string);</td>
</tr>
</tbody>
</table>

Flow constructions (if/while/for/do)

```
    if(expr) statement
        else if(expr) statement
        else statement
    while(expr)
        statement
    for(expr1; expr2; expr3)
        statement
    for(type var : array)
        statement
    do statement
        while(expr);
```
Arrays

array
array element \( i \)
array indexed by elements of int array \( A \)
anonymous array
array containing \( n \) deep copies of \( x \) length
cyclic flag
pop element \( x \)
push element \( x \)
append array \( a \)
insert rest arguments at index \( i \)
delete element at index \( i \)
delete elements with indices in \([1,3]\)
delete all elements
test whether element \( n \) is initialized
array of indices of initialized elements
complement of int array in \([0,\ldots,n-1]\)
deep copy of array \( a \)
array \([0,1,\ldots,n-1]\)
array \([n, n+1, \ldots, m]\)
array \([n-1, n-2, \ldots, 0]\)
array \( f(0), f(1), \ldots, f(n-1) \)
array obtained by applying \( f \) to array \( a \)
uniform partition of \([a, b]\) into \( n \) intervals
concat specified 1D arrays
return sorted array
return array sorted using \( \text{ordering} \)
return sorted array
concat specified 1D arrays
uniform partition of \([a, b]\) into \( n \) intervals
concat specified 1D arrays

Initialization

initialize variable
initialize array

path connectors

straight segment
Bez\( \iota \) segment with implicit control points
Bez\( \iota \) segment with explicit control points
concatenate
lift pen
..tension atleast..1..
..tension atleast infinity..

Labels

implicit cast of string \( s \) to Label
Label \( a \) with relative position and alignment
Label \( a \) with absolute position and alignment
Label \( a \) with specified pen

draw commands

draw path with current pen
draw path with pen
draw labeled path
draw arrow with pen
draw path on picture
draw visible portion of line through two pairs
draw line(pair, pair)
draw(picture, path)
draw(command)
draw(commands)
draw(set)
draw(set, path)
draw(set, command)

fill commands

fill path with current pen
fill path with pen
fill path on picture

label commands

label a pair with optional alignment \( z \)
label a path with optional alignment \( z \)
add label to picture

clip commands

clip to path
clip to path with fill rule
clip picture to path

pens

Grayscale pen from value in \([0,1]\)
RGB pen from values in \([0,1]\)
CMYK pen from values in \([0,1]\)
RGB pen from heximdecimal string
heximdecimal string from rgb pen
hsb pen from values in \([0,1]\)
invisible pen
default pen
current pen
solid pen
dotted pen
wide dotted current pen
wide dotted pen
dashed pen
long dashed pen
dash dotted pen
long dash dotted pen
PostScript butt line cap
PostScript round line cap
PostScript projecting square line cap
miter join
round join
bevel join
pen with miter limit
zero-winding fill rule
even-odd fill rule
align to character bounding box (default)
align to \( \TeX \) baseline
pen with font size (pt)
La\( \TeX \) pen from encoding, family, series, shape
\( \TeX \) pen
scaled \( \TeX \) pen
PostScript font from strings
pen with opacity in \([0,1]\)
construct pen nib from polygonal path
pen mixing operator
path operations

number of segments in path \( p \)
number of nodes in path \( p \)
is path \( p \) cyclic?
is segment \( i \) of path \( p \) straight?
is path \( p \) straight?
coordinates of path \( p \) at time \( t \)
direction of path \( p \) at time \( t \)
direction of path \( p \) at length(\( p \))
unit(dir(\( p \)))+dir(\( q \))
acceleration of path \( p \) at time \( t \)
radius of curvature of path \( p \) at time \( t \)
precontrol point of path \( p \) at time \( t \)
postcontrol point of path \( p \) at time \( t \)
arclength of path \( p \)
time at which arclength(\( p \))=\( L \)
point on path \( p \) at arclength \( L \)
first value \( t \) at which dir(\( p \),\( t \))=\( z \)
time \( t \) at relative fraction \( i \) of arclength(\( p \))
point at relative fraction \( i \) of arclength(\( p \))
point midway along arclength of \( p \)
path running backwards along \( p \)
subpath of \( p \) between times \( a \) and \( b \)
times for one intersection of paths \( p \) and \( q \)
times at which \( p \) reaches minimal extents
times at which \( p \) reaches maximal extents
intersection times of paths \( p \) and \( q \)
intersection times of path \( p \) with "---a--b--''
intersection times of path \( p \) crossing \( x =x \)
intersection times of path \( p \) crossing \( y =y \)
intersection point of paths \( p \) and \( q \)
intersection points of \( p \) and \( q \)
intersection of extension of \( P--Q \) and \( p--q \)
lower left point of bounding box of path \( p \)
upper right point of bounding box of path \( p \)
subpaths of \( p \) split by \( n \)th cut of \( \text{knife} \)
path surrounding region bounded by paths
path filled by \( \text{draw}(g, p) \)
unit square with lower-left vertex at origin
unit circle centered at origin
circle of radius \( r \) about \( c \)
arcc of radius \( r \) about \( c \) from angle \( a \) to \( b \)
unit \( n \)-sided polygon
unit \( n \)-point cyclic cross

pictures

add picture \( \text{pic} \) to \( \text{currentpicture} \)
add picture \( \text{pic} \) about path \( p \)

affine transforms

identity transform
shift by values
shift by pair
scale by \( x \) in the \( x \) direction
scale by \( y \) in the \( y \) direction
scale by \( x \) in both directions
scale by real values \( x \) and \( y \)
map \( (x, y) \to (x+ay, y) \)
rotate by real angle in degrees about path \( p \)
reflect about line from \( P--Q \)

string operations

concatenate operator
string length
time \( \geq \) pos of first occurrence of \( t \) in \( s \)
position \( \leq \) pos of last occurrence of \( t \) in \( s \)
string with \( t \) inserted in \( s \) at pos
string \( s \) with \( \text{before} \) changed to \( \text{after} \)
string \( s \) translated via \({\{\text{before},\text{after}]\ldots} \)
format \( x \) using C-style format string \( s \)
casts hexadecimal string to an integer
casts \( x \) to string using precision digits
current time formatted by format
second of string \( t \) using format
string corresponding to seconds using format
split \( s \) into strings separated by delimiter

identity transform
shift by values
shift by pair
scale by \( x \) in the \( x \) direction
scale by \( y \) in the \( y \) direction
scale by real values \( x \) and \( y \)
rotate by real angle in degrees about path \( p \)
reflect about line from \( P--Q \)

May 2014 v1.1. Copyright © 2014 John C. Bowman
Permission is granted to make and distribute copies of this card, with or without modifications, provided the copyright notice and this permission notice are preserved on all copies.