Asymptote Reference Card

Program structure/functions

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

include "filename"
include verbatim text from file

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

Data types/declarations

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

3D data types (import three;)

ordered triple triple
3D path path3
3D guide guide3
3D affine transform transform3

Constants

exponential form 6.02e23
\TeX string constant "abc\ldots de"
\TeX strings: special characters \, \, \"\, \`
C strings: constant 'abc\ldots de'
C strings: special characters \, \, \"\, \\
C strings: newline, cr, tab, backspace \n \r \t \b
C strings: octal, hexadecimal bytes 0-377 \x0-\x7F

Operators

arithmetic operations + - * /
modulo (remainder) %
comparisons == != > >= < <=
not !
and or (conditional evaluation of RHS) && ||
cast expression to type (type) expr
prefix operators ++ --
suffix operators += -= *= /= %=
conditional expression expr1 ? expr2 : expr3
structure member operator name.member
expression evaluation separator ,

Flow control

statement terminator ;
block delimiters {
comment delimiters /* */
comment to end of line delimiter //
exit from while/do/for break;
next iteration of while/do/for continue;
return value from function
terminate execution
abort execution with error message

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);

1
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,j]
delete all elements
test whether element n is initialized
array of indices of initialized elements
complement of int array in {0,...,n-1}
deep copy of array a
array {0,1,...,n-1}
array {n,n+1,...,m}
array {n-1,n-2,...,0}
array {f(0),f(1),...,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 ordering
return sorted array
concat specified 1D arrays
uniform partition of [a,b] into n intervals
deep copy of array a
deep copies of x array(n,x)

dim (a)
dim (a,n)
dim (a,0)
dim (a,n-1)
dim (a,n)
dim (a,n,m)
dim (a,n+1)
dim (a,n-1)
dim (a,n+1,m)
dim (a,n)
dim (a,0,m)
dim (a,0,0)
dim (a,0,n)
dim (a,0,0)
dim (a,0,0)

Initialization
initialize variable
initialize array
path connectors
straight segment
Bezier segment with implicit control points
Bezier segment with explicit control points
concatenate
lift pen
..tension at least 1..
..tension at least infinity..

Labels
implicit cast of string s to Label
Label s with relative position and alignment
Label s with absolute position and alignment
Label s 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

type[] name;
name[i];
name[A];
new type[dim]
array(n,x)
name.length
name.cyclic
name.pop()
name.push(x)
nname.append(a)
nname.insert(i,1,...)
nname.delete(i)
nname.delete(i,j)
nname.delete()
nname.initialized(n)
nname.keys
complement(a,n)
copy(a)
sequence(n)
reverse(n)
sequence(f,n)
map(f,a)
uniform(a,b,n)
concat(a,b,...)
sort(a)
sort(s,less)
search(a,key)
find(a)
find(a,n)

drawline(pair,pair)
draw path on picture
draw(picture,path)
draw(path)
draw(path,pen)
draw(path,pen,Arrow)
drawline(pair,pair)

draw commands
fill path with current pen
draw path with pen
draw 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
hsv pen from values in [0,1]
invisible pen
default pen
current pen
solid pen
dotted pen
wide dotted pen
wide dotted pen
dashed pen
long dashed pen
dash dotted pen
long dash dotted pen
square cap
circle cap
dash cap
dash cap
dash cap
dash cap
dash cap
dash cap
dash cap
dash cap
dash cap
dash cap
dash cap

fill commands
fill path with current pen
fill path with pen
fill path on picture
label commands
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
hsv pen from values in [0,1]
invisible pen
default pen
current pen
solid pen
dotted pen
wide dotted pen
wide dotted pen
dashed pen
long dashed pen
dash dotted pen
long dash dotted pen
square cap
circle cap
dash cap
dash cap
dash cap
dash cap
dash cap
dash cap
dash cap
dash cap
dash cap
dash cap
dash cap
dash cap
dash cap
dash cap

fill commands
fill path with current pen
fill path with pen
fill path on picture
label commands
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
hsv pen from values in [0,1]
invisible pen
default pen
current pen
solid pen
dotted pen
wide dotted pen
wide dotted pen
dashed pen
long dashed pen
dash dotted pen
long dash dotted pen
square cap
circle cap
dash cap
dash cap
dash cap
dash cap
dash cap
dash cap
dash cap
dash cap
dash cap
dash cap

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 l of arclength(p)
point at relative fraction l 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 =z
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 nth cut of knife
pair z lies within path p
pair z lies within or on path p?
path surrounding region bounded by paths path filled by draw(g,p)
unit square with lower-left vertex at origin
unit circle centered at origin
circle of radius r about c
arc of radius r about c from angle a to b
unit n-sided polygon
unit n-point cyclic cross

pictures

add picture pic to currentpicture
add picture pic about pair z

length(p)
size(p)
cyclic(p)
straight(p,i)
piecewisestraight(p)
point(p,t)
dir(p,t)
dir(p,q)
accel(p,t)
radius(p,t)
precontrol(p,t)
postcontrol(p,t)
arc_length(p)
arc_time(p,L)
arc_point(p,L)
dir_time(p,z)
reltime(p,l)
rel_point(p,l)
midpoint(p)
reverse(p)
subpath(p,a,b)
intersect(p,q)
mintimes(p,q)
maxtimes(p)
intersections(p,q)
times(p,x)
times(p,z)
times_point(p,q)
times_points(p,q)
extension(P,Q,p,q)
min(p)
max(p)
cut(p,knife,n)
interior(p,z)
inside(p,z)
buildcycle(...)
strokepath(g,p)
unitsquare
unitcircle
circle(c,r)
arc(c,r,a,b)
polygon(n)
cross(n)

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) → (x+a, y)
rotate by real angle in degrees about pair z
reflect about line from P--Q

string operations

concatenate operator
string length
position ≥ pos of first occurrence of t in s
position ≤ pos of last occurrence of t in s
string with t inserted in s at pos
string s with n characters at pos erased
substring of string s of length n at pos
string s reversed
string s with before changed to after
string s translated via {{before,after},...}
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
time in seconds of string t using format
string corresponding to seconds using format
split s into strings separated by delimiter

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.