

JUlamono

a monospaced programming font  
with reasonable Unicode support

<https://github.com/cormullion/juliamono>

# JuliaMono Light

abcdefghijklmnoprstuvwxyz 12345  
ABCDEFGHIJKLMNOPQRSTUVWXYZ 67890  
{ } [ ] ( ) <> \$ \* - + = / # \_ % ^ @ \ & | ~ ? ' " ` ! , . ; :

# JuliaMono Regular

abcdefghijklmnoprstuvwxyz 12345  
ABCDEFGHIJKLMNOPQRSTUVWXYZ 67890  
{ } [ ] ( ) <> \$ \* - + = / # \_ % ^ @ \ & | ~ ? ' " ` ! , . ; :

# JuliaMono Medium

abcdefghijklmnoprstuvwxyz 12345  
ABCDEFGHIJKLMNOPQRSTUVWXYZ 67890  
{ } [ ] ( ) <> \$ \* - + = / # \_ % ^ @ \ & | ~ ? ' " ` ! , . ; :

# JuliaMono Bold

abcdefghijklmnoprstuvwxyz 12345  
ABCDEFGHIJKLMNOPQRSTUVWXYZ 67890  
{ } [ ] ( ) <> \$ \* - + = / # \_ % ^ @ \ & | ~ ? ' " ` ! , . ; :

# JuliaMono ExtraBold

abcdefghijklmnoprstuvwxyz 12345  
ABCDEFGHIJKLMNOPQRSTUVWXYZ 67890  
{ } [ ] ( ) <> \$ \* - + = / # \_ % ^ @ \ & | ~ ? ' " ` ! , . ; :

# JuliaMono Black

abcdefghijklmnoprstuvwxyz 12345  
ABCDEFGHIJKLMNOPQRSTUVWXYZ 67890  
{ } [ ] ( ) <> \$ \* - + = / # \_ % ^ @ \ & | ~ ? ' " ` ! , . ; :

```
function _nloops(N::Int, itersym::Symbol, rangeexpr::Expr, args::Expr...)
    if rangeexpr.head !== :→
        throw(ArgumentError("second argument must be an anonymous function expression to compute the range"))
    end
    if !(1 ≤ length(args) ≤ 3)
        throw(ArgumentError("number of arguments must be 1 ≤ length(args) ≤ 3, got $args"))
    end
    body = args[end]
    ex = Expr(:escape, body)
    for dim = 1:N
        itervar = inlineanonymous(itersym, dim)
        rng = inlineanonymous(rangeexpr, dim)
        preexpr = length(args) > 1 ? inlineanonymous(args[1], dim) : (:(:nothing))
        postexpr = length(args) > 2 ? inlineanonymous(args[2], dim) : (:(:nothing))
        ex = quote
            for $(esc(itervar)) = $(esc(rng))
                $(esc(preexpr))
                $ex
                $(esc(postexpr))
            end
        end
    end
    ex
end

function _nloops(N::Int, itersym::Symbol, rangeexpr::Expr, args::Expr...)
    if rangeexpr.head !== :→
        throw(ArgumentError("second argument must be an anonymous function expression to compute the range"))
    end
    if !(1 ≤ length(args) ≤ 3)
        throw(ArgumentError("number of arguments must be 1 ≤ length(args) ≤ 3, got $args"))
    end
    body = args[end]
    ex = Expr(:escape, body)
    for dim = 1:N
        itervar = inlineanonymous(itersym, dim)
        rng = inlineanonymous(rangeexpr, dim)
        preexpr = length(args) > 1 ? inlineanonymous(args[1], dim) : (:(:nothing))
        postexpr = length(args) > 2 ? inlineanonymous(args[2], dim) : (:(:nothing))
        ex = quote
            for $(esc(itervar)) = $(esc(rng))
                $(esc(preexpr))
                $ex
                $(esc(postexpr))
            end
        end
    end
    ex
end

function _nloops(N::Int, itersym::Symbol, rangeexpr::Expr, args::Expr...)
    if rangeexpr.head !== :→
        throw(ArgumentError("second argument must be an anonymous function expression to compute the range"))
    end
    if !(1 ≤ length(args) ≤ 3)
        throw(ArgumentError("number of arguments must be 1 ≤ length(args) ≤ 3, got $args"))
    end
    body = args[end]
    ex = Expr(:escape, body)
    for dim = 1:N
        itervar = inlineanonymous(itersym, dim)
        rng = inlineanonymous(rangeexpr, dim)
        preexpr = length(args) > 1 ? inlineanonymous(args[1], dim) : (:(:nothing))
        postexpr = length(args) > 2 ? inlineanonymous(args[2], dim) : (:(:nothing))
        ex = quote
            for $(esc(itervar)) = $(esc(rng))
                $(esc(preexpr))
                $ex
                $(esc(postexpr))
            end
        end
    end
    ex
end
```



**Ancient Greek** Ἀδμηθ', ὁρᾶς γὰρ τάμα πράγμαθ', ώς ἔχει, λέξαι θέλω σοι πρὶν θανεῖν ἢ βούλομαι.

**Bulgarian** Я, пазачът Валъо уж бди, а скришом хапва кюфтенца зад щайгите.

**Catalan** «Dóna amor que seràs feliç!». Això, il·lus company geniüt, ja és un lliüt rētol blavís d'onze kWh.

**Czech** Zvlášť zákeřný učeň s doličky běží podél zóny úlu

**Danish** Quizdeltagerne spiste jordbær med fløde, mens cirkusklovnen Walther spillede på xylofon.

**English** Sphinx of black quartz, judge my vow.

**Estonian** Põdur Zagrebi tšellomängija-fölljetonist Ciqo külmetas kehvas garaažis

**Finnish** Charles Darwin jammaili Åken hevixylofonilla Qatarin yöpub Zeligissä.

**French** Voix ambiguë d'un cœur qui au zéphyr préfère les jattes de kiwi.

**Georgian** სწრაფი ყავისფერი მელა ახტება ზარმაც ძაღლს.

**German** Victor jagt zwölf Boxkämpfer quer über den großen Sylter Deich.

**Greek** Ταχίστη αλώπηξ βαφής ψημένη γη, δρασκελίζει υπέρ νωθρού κυνός.

**Guarani** H̄ilandiagua kuñanguéra oho peteī sa'uju ypa'ūme Gavōme ombo'e hāguá ingyleñe'ē mitānguérale ne'ēndy'ŷ.

**Hungarian** Jó foxim és don Quijote húszwattos lámpánál ülve egy pár bűvös cipőt készít.

**IPA** [ɛwɔ?.nas.do:ŋ.kʰlja]  
[ɛjan.ɛzliwo.çiuɛn.χwa]

**Icelandic** Kæmi ný öxi hér, ykist þjófum nú bæði víl og ádrepa.

**Irish** Čuaiġ bé mórsáċ le dlúitspád fiorfinn trí hata mo dea-þorcáin big.

**Latvian** Muļķa hipiji mēģina brīvi nogaršot celofāna žņaudzējčūsku.

**Lithuanian** Įlinkdama fechtuotojo špaga sublykčiojusi pragrėžę apvalų arbūzą.

**Macedonian** Сидарски пејзаж: шугав билмез со чудење џвака ќофте и кељ на туѓ цех.

**Norwegian** Jeg begynte å fortære en sandwich mens jeg kjørte taxi på vei til quiz

**Polish** Pchnąć w tę łódź jeża lub ósm skrzyń fig.

**Portuguese** Luís argüia à Júlia que «brações, fé, chá, óxido, pôr, zângão» eram palavras do português.

**Romanian** Înjurând pițigăiat, zoofobul comandă vexat whisky și tequila.

**Russian** Широкая электрификация южных губерний даст мощный толчок подъёму сельского хозяйства.

**Scottish** Mus d'fhàg Cèit-Ùna ròp ì le ob.

**Serbian** Ајшо, лепото и чежњо, за љубав срца мога дођи у Хаџиће на кафу.

**Spanish** Benjamín pidió una bebida de kiwi y fresa; Noé, sin vergüenza, la más champaña del menú.

**Swedish** Flygande bäckasiner söka hwila på mjuka tuvor.

**Turkish** Pijamalı hasta yağız şoföre çabucak güvendi.

**Ukrainian** Чуєш іх, доцю, га? Кумедна жти, прощайся без ґольфів!

yotta	Y	$10^{24}$	1 000 000 000 000 000 000 000 000	septillion
zetta	Z	$10^{21}$	1 000 000 000 000 000 000 000 000	sextrillion
exa	E	$10^{18}$	1 000 000 000 000 000 000 000 000	quintillion
peta	P	$10^{15}$	1 000 000 000 000 000 000 000 000	quadrillion
tera	T	$10^{12}$	1 000 000 000 000	trillion
giga	G	$10^9$	1 000 000 000	billion
mega	M	$10^6$	1 000 000	million
kilo	k	$10^3$	1 000	thousand
hecto	h	$10^2$	100	hundred
deca	da	$10^1$	10	ten
deci	d	$10^{-1}$	0.1	tenth
centi	c	$10^{-2}$	0.01	hundredth
milli	m	$10^{-3}$	0.001	thousandth
micro	$\mu$	$10^{-6}$	0.000 001	millionth
nano	n	$10^{-9}$	0.000 000 001	billionth
pico	p	$10^{-12}$	0.000 000 000 001	trillionth
femto	f	$10^{-15}$	0.000 000 000 000 001	quadrillionth
atto	a	$10^{-18}$	0.000 000 000 000 000 001	quintillionth
zepto	z	$10^{-21}$	0.000 000 000 000 000 000 001	sextillionth
yocto	y	$10^{-24}$	0.000 000 000 000 000 000 000 001	septillionth

classical electron radius	$r_e$	$2.817940285 \times 10^{-15}$
Compton wavelength of the electron	$\lambda_c$	$2.426310215 \times 10^{-12}$
reduced Compton wavelength of the electron	$\tilde{\lambda}_c$	$3.8615926764 \times 10^{-13}$
Bohr radius of the hydrogen atom	$a_0$	$5.291772083 \times 10^{-11}$
natural units based on the electronvolt	$1 \text{ eV}^{-1}$	$1.97 \times 10^{-7}$
reduced wavelength of hydrogen radiation	$1/R_\infty$	$9.112670505509 \times 10^{-8}$
Planck length	$\ell_p$	$1.616199 \times 10^{-35}$
Stoney unit of length	$l_s$	$1.381 \times 10^{-35}$
quantum chromodynamics (QCD) unit of length	$l_{\text{qcd}}$	$2.103 \times 10^{-16}$

	mm		inches		points	
	w	h	w	h	w	h
A0	841	1189	33.11	46.81	2384	3370
A1	594	841	23.39	33.11	1684	2384
A2	420	594	16.54	23.39	1190	1684
A3	297	420	11.69	16.54	842	1190
A4	210	297	8.27	11.69	595	842
A5	148	210	5.83	8.27	420	595
A6	105	148	4.13	5.83	298	420
A7	74	105	2.91	4.13	210	298
A8	52	74	2.05	2.91	148	210
Letter (ANSI A)	215.9	279.4	8.5	11	612	792
Legal	215.9	355.6	8.5	14	612	1008
Ledger (ANSI B)	279.4	431.8	11	17	792	1224
Tabloid (ANSI B)	431.8	279.4	17	11	1224	792
Executive	184.1	266.7	7.25	10.55	522	756
ANSI C	559	432	22	17	1584	1224
ANSI D	864	559	34	22	2448	1584
ANSI E	1118	864	44	34	3168	2448
Foolscap	336	419	13.25	16.5	954	1188
Small Post	368	469	14.5	18.5	1044	1332
Sheet and 1/3 cap	336	588	13.25	22	954	1584
Sheet and 1/2 cap	336	628	13.25	24.75	954	1782
Demy	394	507	15.5	20	1116	1440
Large Post	419	533	16.5	21	1188	1512
Small medium	444	558	17.5	22	1260	1584
Medium	457	584	18	23	1296	1656
Small Royal	482	609	19	24	1368	1728
Royal	507	634	20	25	1440	1800
Imperial	559	761	22	30	1584	2160



v cat

u+e838



Braille 2345 (t)

u+281e



blackboard S

u+1d54a



Julia dots

u+e800



editorial coronis

u+2e0e



triple integral

u+222d



alembic

u+2697



APL dot tack jot

u+234e



left arrow subset

u+297a



transversal intersect

u+2adb



weierstrass

u+2118



Pluto

u+2bd4



circled asterisk

u+229b



heavy right arrow

u+1f882



G circled

u+24bc



Mayan 14

u+1d2ee



yod

u+5d9



numero

u+2116



heavy 12pt Pinwheel Star

u+1f7d4



subscript b

u+e805



fist

u+270a



division times

u+22c7



amalgamation Or Coproduct

u+2a3f



ace of spades

u+1f0a1



Sharp S

u+1e9e



a



percent sign



ogham nion

u+1685



segno

u+1d10b



tetragram divergence

u+1d310



white upperleft square

u+25f0



recycled paper

u+267c



Roman 50K

u+2187



eternity

u+58e



subset with +

u+2abf



psi

u+3c8



umbrella

u+2602



triple dagger

u+2e4b



blackboard a

u+1d552



alef

u+2135



devangari 5

u+96b



swash ampersand

u+1f675



coptic 800

u+102fa



cube root

u+221b



git PR

u+e726



hacker

u+e826



zNot Rel Comp

u+2a3e



R fraktur

u+211c



speech bubble

u+1f4ac



hbar

u+210f



earth ground

u+23da



trans Pluto

u+2bd7



palm branch

u+e819



per

u+214c



hexagram completion

u+4df6



DifyQ

u+e830



smash product

u+2a33

random Selection of Unicode characters

## contextual alternates

calt off	calt on
->	→
=>	⇒
>	▷
<	△
::	::

## stylistic sets

zero	0	0	slashed zero
ss01	g	g	alternate g
ss02	@	@	alternate @
ss03	j	j	alternate j
ss04	ø	ø	alternate ø
ss05	*	*	lighter asterisk
ss06	a	a	simple a
ss07	`	`	smaller grave
ss08	->	→	distinct ligatures†
ss09	f	f	alternate f
ss10	r	r	alternate r
ss11	'	'	thinner grave

+ with calt off

```
45 @inline function ntuple(f::F, ::Val{N}) where {F,N}
46     N::Int
47     (N >= 0) || throw(ArgumentError(string("tuple length should be ≥ 0, got ", N)))
48     if @generated
49         quote
50             @nexprs $N i → t_i = f(i)
51             @ncall $N tuple t
52         end
53     else
54         Tuple(f(i) for i = 1:N)
55     end
56 end
57
58 @inline function fill_to_length(t::Tuple, val, ::Val{_N}) where {_N}
59     M = length(t)
60     N = _N::Int
61     M > N && throw(ArgumentError("input tuple of length $M, requested $N"))
62     if @generated
63         quote
64             (t..., $(fill(:val, (_N::Int) - length(t.parameters))...))
65         end
66     else
67         (t..., fill(val, N-M)...)
```

```
@inline function _foldoneto(op, acc, ::Val{N}) where N
    @assert N::Integer > 0
    if @generated
        quote
            acc_0 = acc
            Base.Cartesian.@nexprs $N i ->
                acc_{i} = op(acc_{i-1},
            return $(Symbol(:acc_, N)))
        end
    else
        quote
            for i in 1:N
                acc = op(acc, i)
            end
        end
    end
end
```



acoth cumprod keys ismount getproperty retry  $U$  @view include\_dependency AbstractVector showerror seek @cmd propertynames splitdir ispath denominator @generated include\_string strides AbstractIrrational in process\_running Cushort & <: Inf mapslices cld rounding InsertionSort indexin NaN showable iterate flush acotd @label ENV nameof Pipe Cstring tanh deg2rad cosd sortperm! eps hasmethod maximum! mark foldr g RoundNearestTiesAway KeyError BitVector partialsortperm! trues @inbounds rpad cosc enumerate code\_lowered symlink union atexit isuppercase ExponentialBackOff NaN64 ascii findmin! fill issubset log10 oper M Complex ispunct coth fldmod1  $\triangleright$  zip put! < ~ findmax isdigit Threads numerator oneunit valtype Inf16 DimensionMismatch StridedVector Libc circshift asecd VecOrMat  $\triangleright$  factorial leading\_zeros isetuid >> set.z split @s\_str NaN32 IOContext asinh @macroexpand sech partialsortperm truncate / SubstitutionString seed = @r\_str BitSet CompositeException RoundNearestTiesUp Docs cospi task\_local\_storage isfifo OrdinalRan widemul uppercase cscd redisplay istaskdone isopen prevfloat time TextDisplay findlast AbstractDict fieldoffset & reset run log1p broadcast! filter foldl ifelse isprimitivetype SubArray lstat eachindex > @mac last stat atanb broadcast skipmissing AbstractDisplay abs eltype istaskstarted simpi Cchar similar popat! tempdir ismissing UnitRange min rem2pi partialsort eachline Dims @async StridedArray @inline setdim AbstractVecOrMat sortslices diff! filesize length argmin hcat setrounding ndims symdiff unsafe\_read foreach nextprod acos RoundNearest sinc Cintmax\_t significand | hypot sleep all! countlines yieldto ista homedir pointer\_from\_objref relpath parent map! bytesavailable typejoin isreal unsafe\_write typeintersect chown ischardev uppercasefirst > sortperm SubString firstindex RoundingMode findall modf Csize\_t non repr count isascii pi isempty string Cfloat extrema QuickSort clamp maxintfloat im isinteger redirect\_stdin Iterators schedule Dict accumulate ComplexF32 insert! isspace tryparse @doc searchsorted @views str RoundDown kill splitpath Pair readbytes! AbstractChannel StepRange prevpow muladd digits pop! @timed walkdir IdDict  $\ni$  partialsort! skip C\_NULL @raw\_str cumsum! Rational digits! bitstring exponent fullname isprint all signbit hasfield get\_zero\_subnormals DenseVecOrMat BitMatrix abs2 Clonglong Irrational unescape\_string chomp invperm getindex  $\leq$  VersionNumber isdispatchtuple / IOBuffer isblockdev @specialize h circcopy! isready sqrt minimum mod1 ReentrantLock BigFloat view % gensym eof count\_ones big @threadcall replace! display gcd @int128.str bytes2hex withenv append! mv vec argmax temname ComplexF16 xor e c isbits dirname  $\pi$  rand yield copy! iswritable reverse! println @goto supertype @nospecialize \ CartesianIndex acot Cshort checkindex issubnormal @\_DIR\_ joinpath merge @deprecated Array reduce reim floor We cmp RawFD rem AbstractSet pairs endswith stderr pathof floatmin timedwait sign DenseVector isassigned : @\_MODULE\_ frexp isabstracttype powermod fd normpath Cmd redirect\_stdout signed Cptrdiff\_t tand copy range ones fd csch invmod isdirpath log lpad open htn RoundToZero GC mapfoldr ProcessFailedException Cwchart LinearIndices IndexStyle pipeline transpose parentmodule selectdim @time ^ div rstrip sort LO read ismutable ignorestatus findmax! redirect\_stderr ARGs notify readline step fieldcount empty! eachcol kton mtime isinteractive sizehint! splitdrive TaskFailedException download @eval CapturedException ax permutedims! inv values Missing size isnan StackTraces show sizeof cosh bitrotate findfirst hash unmark readlines stdout acos delete! isone NaN16 Cint StridedMatrix nextpow isfile acsch fma cp @text\_str Clon # sincos prod hvcat VERSION StringIndexError allunique lastindex ctime haskey Inf64 contains isinf IOStream invpermute! falses count! complex isabspath occurin one pointer PartialQuickSort Set merge! flip IndexCartesian ispow2 isbits type convert @isdefined sec escape\_string filter! RoundFromZero BigInt v hex2bytes empty IndexLinear cumsum ComplexF64 \* filemode get! fetch popfirst! @MIME\_str @\_dot\_ isdir Cwst @big\_str float basename searchsortedfirst circshift! unsafe\_wrap pkgdir issorted asyncmap atan startswith @ccall isequal rotr90 BitArray asyncmap mapfoldl ptring seed = @on RoundToZero GC mapfoldr ProcessFailedException Cwchart LinearIndices IndexStyle pipeline transp cd intersect! seekstart stepfieldcount empty! eachcol kton mtime isinteractive sizehint! splitdrive TaskFailedException download @eval CapturedException ax prod! Cuint @simd isapprox @warn option Cwchart LinearIndices IndexStyle pipeline transp cd intersect! seekstart stepfieldcount empty! eachcol kton mtime isinteractive sizehint! splitdrive TaskFailedException download @eval CapturedException ax acoth cumprod keys ismount getproperty retry  $U$  @view include\_dependency AbstractVector showerror seek @cmd propertynames splitdir ispath denominator @generated include\_string strides AbstractIrrational in process\_running Cushort & <: Inf mapslices cld rounding InsertionSort indexin NaN showable iterate flush acotd @label ENV nameof Pipe Cstring tanh deg2rad cosd sortperm! eps hasmethod maximum! mark foldr g RoundNearestTiesAway KeyError BitVector partialsortperm! trues @inbounds rpad cosc enumerate code\_lowered symlink union atexit isuppercase ExponentialBackOff NaN64 ascii findmin! fill issubset log10 oper M Complex ispunct coth fldmod1  $\triangleright$  zip put! < ~ findmax isdigit Threads numerator oneunit valtype Inf16 DimensionMismatch StridedVector Libc circshift asecd VecOrMat  $\triangleright$  factorial leading\_zeros isetuid >> set.z split @s\_str NaN32 IOContext asinh @macroexpand sech partialsortperm truncate / SubstitutionString seed = @r\_str BitSet CompositeException RoundNearestTiesUp Docs cospi task\_local\_storage isfifo OrdinalRan widemul uppercase cscd redisplay istaskdone isopen prevfloat time TextDisplay findlast AbstractDict fieldoffset & reset run log1p broadcast! filter foldl ifelse isprimitivetype SubArray lstat eachindex > @mac last stat atanb broadcast skipmissing AbstractDisplay abs eltype istaskstarted simpi Cchar similar popat! tempdir ismissing UnitRange min rem2pi partialsort eachline Dims @async StridedArray @inline setdim AbstractVecOrMat sortslices diff! filesize length argmin hcat setrounding ndims symdiff unsafe\_read foreach nextprod acos RoundNearest sinc Cintmax\_t significand | hypot sleep all! countlines yieldto ista homedir pointer\_from\_objref relpath parent map! bytesavailable typejoin isreal unsafe\_write typeintersect chown ischardev uppercasefirst > sortperm SubString firstindex RoundingMode findall modf Csize\_t non repr count isascii pi isempty string Cfloat extrema QuickSort clamp maxintfloat im isinteger redirect\_stdin Iterators schedule Dict accumulate ComplexF32 insert! isspace tryparse @doc searchsorted @views str RoundDown kill splitpath Pair readbytes! AbstractChannel StepRange prevpow muladd digits pop! @timed walkdir IdDict  $\ni$  partialsort! skip C\_NULL @raw\_str cumsum! Rational digits! bitstring exponent fullname isprint all signbit hasfield get\_zero\_subnormals DenseVecOrMat BitMatrix abs2 Clonglong Irrational unescape\_string chomp invperm getindex  $\leq$  VersionNumber isdispatchtuple / IOBuffer isblockdev @specialize h circcopy! isready sqrt minimum mod1 ReentrantLock BigFloat view % gensym eof count\_ones big @threadcall replace! display gcd @int128.str bytes2hex withenv append! mv vec argmax temname ComplexF16 xor e c isbits dirname  $\pi$  rand yield copy! iswritable reverse! println @goto supertype @nospecialize \ CartesianIndex acot Cshort checkindex issubnormal @\_DIR\_ joinpath merge @deprecated Array reduce reim floor We cmp RawFD rem AbstractSet pairs endswith stderr pathof floatmin timedwait sign DenseVector isassigned : @\_MODULE\_ frexp isabstracttype powermod fd normpath Cmd redirect\_stdout signed Cptrdiff\_t tand copy range ones fd csch invmod isdirpath log lpad open htn RoundToZero GC mapfoldr ProcessFailedException Cwchart LinearIndices IndexStyle pipeline transp cd intersect! seekstart stepfieldcount empty! eachcol kton mtime isinteractive sizehint! splitdrive TaskFailedException download @eval CapturedException ax permutedims! inv values Missing size isnan StackTraces show sizeof cosh bitrotate findfirst hash unmark readlines stdout acos delete! isone NaN16 Cint StridedMatrix nextpow isfile acsch fma cp @text\_str Clon # sincos prod hvcat VERSION StringIndexError allreplace! display gcd @int128.str bytes2hex withenv append! mv vec argmax temname ComplexF16 xor e c isbits dirname  $\pi$  rand yield copy! iswritable reverse! println @goto supertype @nospecialize \ CartesianIndex acot Cshort checkindex issubnormal @\_DIR\_ joinpath merge @deprecated Array reduce reim floor We cmp RawFD rem AbstractSet pairs endswith stderr pathof floatmin timedwait sign DenseVector isassigned : @\_MODULE\_ frexp isabstracttype powermod fd normpath Cmd redirect\_stdout signed Cptrdiff\_t tand copy range ones fd csch invmod isdirpath log lpad open htn RoundToZero GC mapfoldr ProcessFailedException Cwchart LinearIndices IndexStyle pipeline transp cd intersect! seekstart stepfieldcount empty! eachcol kton mtime isinteractive sizehint! splitdrive TaskFailedException download @eval CapturedException ax prod! Cuint @simd isapprox @warn option Cwchart LinearIndices IndexStyle pipeline transp cd intersect! seekstart stepfieldcount empty! eachcol kton mtime isinteractive sizehint! splitdrive TaskFailedException download @eval CapturedException ax acoth cumprod keys ismount getproperty retry  $U$  @view include\_dependency AbstractVector showerror seek @cmd propertynames splitdir ispath denominator @generated include\_string strides AbstractIrrational in process\_running Cushort & <: Inf mapslices cld rounding InsertionSort indexin NaN showable iterate flush acotd @label ENV nameof Pipe Cstring tanh deg2rad cosd sortperm! eps hasmethod maximum! mark foldr g RoundNearestTiesAway KeyError BitVector partialsortperm! trues @inbounds rpad cosc enumerate code\_lowered symlink union atexit isuppercase ExponentialBackOff NaN64 ascii findmin! fill issubset log10 oper M Complex ispunct coth fldmod1  $\triangleright$  zip put! < ~ findmax isdigit Threads numerator oneunit valtype Inf16 DimensionMismatch StridedVector Libc circshift asecd VecOrMat  $\triangleright$  factorial leading\_zeros isetuid >> set.z split @s\_str NaN32 IOContext asinh @macroexpand sech partialsortperm truncate / SubstitutionString seed = @r\_str BitSet CompositeException RoundNearestTiesUp Docs cospi task\_local\_storage isfifo OrdinalRan widemul uppercase cscd redisplay istaskdone isopen prevfloat time TextDisplay findlast AbstractDict fieldoffset & reset run log1p broadcast! filter foldl ifelse isprimitivetype SubArray lstat eachindex > @mac last stat atanb broadcast skipmissing AbstractDisplay abs eltype istaskstarted simpi Cchar similar popat! tempdir ismissing UnitRange min rem2pi partialsort eachline Dims @async StridedArray @inline setdim AbstractVecOrMat sortslices diff! filesize length argmin hcat setrounding ndims symdiff unsafe\_read foreach nextprod acos RoundNearest sinc Cintmax\_t significand | hypot sleep all! countlines yieldto ista names in Julia Base istaskdone isopen prevonly rotr90 @textDisplay findlast AbstractDict fieldoffset & reset run log1p broadcast! filter foldl ifelse isprimitivetype SubArray lstat eachindex > @mac last stat atanb broadcast skipmissing AbstractDisplay abs eltype istaskstarted simpi Cchar similar popat! tempdir ismissing UnitRange min rem2pi partialsort eachline Dims @async StridedArray @inline setdim AbstractVecOrMat sortslices diff! filesize length argmin hcat setrounding ndims symdiff unsafe\_read foreach nextprod acos RoundNearest sinc Cintmax\_t significand | hypot sleep all! countlines yieldto ista

The title of the painting, which is known in English as *Mona Lisa*, comes from a description by Renaissance art historian Giorgio Vasari, who wrote "Leonardo undertook to paint, for Francesco del Giocondo, the portrait of *Mona Lisa*, his wife." *Mona* in Italian is a polite form of address originating as *ma donna* – similar to *Ma'am*, *Madam*, or *my lady* in English. This became *madonna*, and its contraction *mona*. The title of the painting, though traditionally spelled *Mona* (as used by Vasari), is also commonly spelled in modern Italian as *Monna Lisa* (*mona* being a vulgarity in some Italian dialects), but this is rare in English. Vasari's account of the *Mona Lisa* comes from his biography of Leonardo published in 1550, 31 years after the artist's death. It has long been the best-known source of information on the provenance of the work and identity of the sitter. Leonardo's assistant Salai, at his death in 1524, owned a portrait which in his personal papers was named *la Gioconda*, a painting bequeathed to him by Leonardo. That Leonardo painted such a work, and its date, were confirmed in 2005 when a scholar at Heidelberg University discovered a marginal note in a 1477 printing of a volume by ancient Roman philosopher Cicero. Dated October 1503, the note was written by Leonardo's contemporary Agostino Vespucci. This note likens Leonardo to renowned Greek painter Apelles, who is mentioned in the text, and states that Leonardo was at that time working on a painting of *Lisa del Giocondo*. In response to the announcement of the discovery of this document, Vincent Delieuvin, the Louvre representative, stated "Leonardo da Vinci was painting, in 1503, the portrait of a Florentine lady by the name of *Lisa del Giocondo*. About this we are now certain. Unfortunately, we cannot be absolutely certain that this portrait of *Lisa del Giocondo* is the painting of the Louvre." The model, *Lisa del Giocondo*, was a member of the Gherardini family of Florence and Tuscany, and the wife of wealthy Florentine silk merchant Francesco del Giocondo. The painting is thought to have been commissioned for their new home, and to celebrate the birth of their second son, Andrea. The Italian name for the painting, *La Gioconda*, means 'jocund' ('happy' or 'joyful') or, literally, 'the jocund one', a pun on the feminine form of *Lisa*'s married name, *Giocondo*. In French, the title *La Joconde* has the same meaning. Before that discovery, scholars had developed several alternative views as to the subject of the painting. Some argued that *Lisa del Giocondo* was the subject of a different portrait, identifying at least four other paintings as the *Mona Lisa* referred to by Vasari. Several other women have been proposed as the subject of the painting. Isabella of Aragon, Cecilia Gallerani, Costanza d'Avalos, Duchess of Francavilla, Isabella d'Este, Pacifica Brandano or Brandino, Isabela Gualanda, Caterina Sforza, Bianca Giovanna Sforza—even Salai and Leonardo himself—are all among the list of posited models portrayed in the painting. The consensus of art historians in the 21st century maintains the long-held traditional opinion that the painting depicts *Lisa del Giocondo*. History: Leonardo da Vinci had begun working on a portrait of *Lisa del Giocondo*, the model of the *Mona Lisa*, by October 1503. It is believed by some that the *Mona Lisa* was begun in 1503 or 1504 in Florence. Although the Louvre states that it was "doubtless painted between 1503 and 1506", art historian Martin Kemp says that there are some difficulties in confirming the dates with certainty. In addition, many Leonardo experts, such as Carlo Pedretti and Alessandro Vezzosi, are of the opinion that the painting is characteristic of Leonardo's style in the final years of his life, post-1513. Other academics argue that, given the historical documentation, Leonardo would have

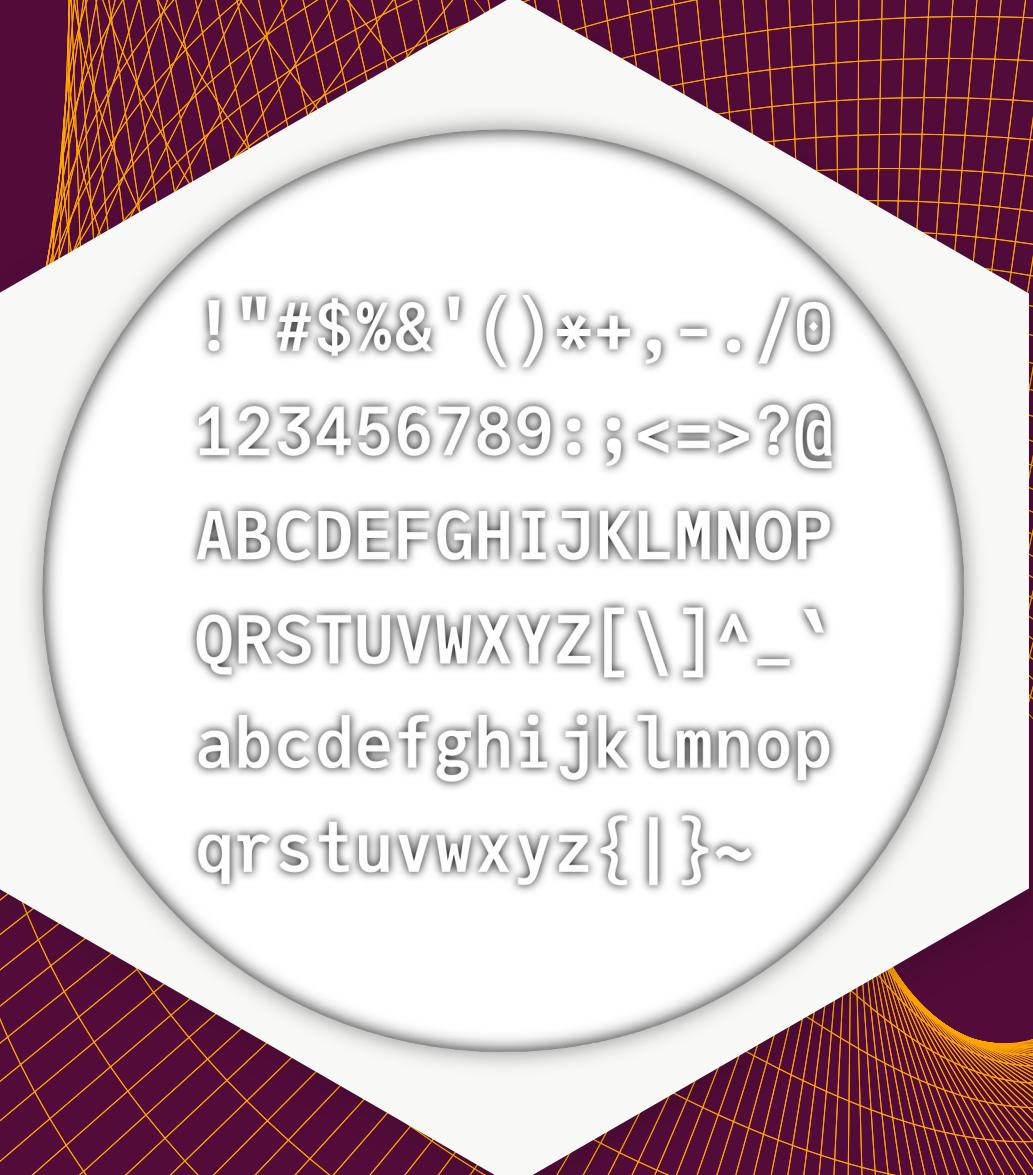
painted the work from 1513. According to Vasari, "after he had lingered over it four years, left it unfinished". In 1516, Leonardo was invited by King Francis I to work at the Clos Lucé near the Château d'Amboise; it is believed that he took the *Mona Lisa* with him and continued to work on it after he moved to France. Art historian Carmen C. Bambach has concluded that Leonardo probably continued refining the work until 1516 or 1517. Leonardo's right hand was paralytic circa 1517, which

2004. In view of this, Vincent Delieuvin, curator of 16th-century Italian painting at the Louvre, states that the sketch and these other copies must have been inspired by another version, while Zöllner states that the sketch may be after another Leonardo portrait of the same subject. The record of an October 1517 visit by Louis d'Aragon states that the *Mona Lisa* was executed for the deceased Giuliano de' Medici, Leonardo's steward at the Belvedere Palace between 1513 and

and assistant Salai's possession until his death in 1524. The second, commissioned by Giuliano de' Medici circa 1513, would have been sold by Salai to Francis I in 1518 and is the one in the Louvre today. Others believe that there was only one true *Mona Lisa*, but are divided as to the two aforementioned fates. At some point in the 16th century, a varnish was applied to the painting. It was kept at the Palace of Fontainebleau until Louis XIV moved it to the Palace of Versailles, where it remained until the French Revolution. In 1797, it was put on to permanent display at the Louvre. In the early 21st century, French scientist Pascal Cotte hypothesized a hidden portrait underneath the surface of the painting, circumstantial evidence for which was produced using reflective light technology. The underlying portrait appears to be of a model looking to the side, but lacks the flanking columns drawn by Raphael. Having been given access to the painting by the Louvre in 2004, Cotte spent ten years studying the painting with layer-amplification methods. However, the alleged portrait does not fit with historical descriptions of the painting: both Vasari and Gian Paolo Lomazzo describe the subject as smiling, unlike the subject in Cotte's portrait. Cotte admits that his reconstitution had been carried out only in support of his hypotheses and should not be considered as objective proof of an underlying portrait. Refuge, theft and vandalism After the French Revolution, the painting was moved to the Louvre, but spent a brief period in the bedroom of Napoleon (d. 1821) in the Tuileries Palace. The *Mona Lisa* was not widely known outside the art world, but in the 1800s, a portion of the French intelligentsia began to hail it as a masterpiece of Renaissance painting. During the Franco-Prussian War (1870–1871), the painting was moved from the Louvre to the Brest Arsenal. Vacant wall in the Louvre's Salon Carré after the painting was stolen in 1911 "La Joconde est Retrouvée" ("Mona Lisa is Found"), Le Petit Parisien, 13 December 1913. In 1911, the painting was still not popular among the lay-public. On 21 August 1911, the painting was stolen from the Louvre. The missing painting was first noticed the next day by painter Louis Béroud. After some confusion as to whether the painting was being photographed somewhere, the Louvre was closed for a week for investigation. French poet Guillaume Apollinaire came under suspicion and was arrested and imprisoned. Apollinaire implicated his friend Pablo Picasso, who was brought in for questioning. Both were later exonerated. The real culprit was Louvre employee Vincenzo Peruggia, who had helped construct the painting's glass case. He carried out the theft by entering the building during regular hours, hiding in a broom closet, and walking out with the painting hidden under his coat after the museum had closed. Peruggia was an Italian patriot who believed that Leonardo's painting should have been returned to an Italian museum. Peruggia may have been motivated by an associate whose copies of the original would significantly rise in value after the painting's theft. After having kept the *Mona Lisa* in his apartment for two years, Peruggia grew impatient and was caught when he attempted to sell it to Giovanni Poggi, director of the Uffizi Gallery in Florence. It was exhibited in the Uffizi Gallery for over two weeks and returned to the Louvre on 4 January 1914. Peruggia served six months in prison for the crime and was hailed for his patriotism in Italy. A year after the theft, Saturday Evening Post journalist Karl Decker met an alleged accomplice named Eduardo de Valfierro, who claimed to have masterminded the theft. Forger Yves Chaudron was to have created ... [text by Wikipedia contributors. "Mona Lisa." Wikipedia, The Free Encyclopedia.]

may indicate why he left the *Mona Lisa* unfinished. Circa 1505, Raphael executed a pen-and-ink sketch, in which the columns flanking the subject are more apparent. Experts universally agree that it is based on Leonardo's portrait. Other later copies of the *Mona Lisa*, such as those in the National Museum of Art, Architecture and Design and The Walters Art Museum, also display large flanking columns. As a result, it was thought that the *Mona Lisa* had been trimmed. However, by 1993, Frank Zöllner observed that the painting surface had never been trimmed; this was confirmed through a series of tests in

1516—but this was likely an error. According to Vasari, the painting was created for the model's husband, Francesco del Giocondo. A number of experts have argued that Leonardo made two versions (because of the uncertainty concerning its dating and commissioner, as well as its fate following Leonardo's death in 1519, and the difference of details in Raphael's sketch—which may be explained by the possibility that he made the sketch from memory). The hypothetical first portrait, displaying prominent columns, would have been commissioned by Giocondo circa 1503, and left unfinished in Leonardo's pupil



!"#\$%&' ()\*+, -./0  
123456789: ;<=>?@  
ABCDEFGHIJKLMNP  
QRSTUVWXYZ[\]^\_`  
abcdefghijklmnop  
qrstuvwxyz{|}~

JuliaMono Light

JuliaMono Regular

JuliaMono Medium

JuliaMono Bold

JuliaMono ExtraBold

JuliaMono Black

```
n == 10 ? (†(1), †(2), †(3), †(4), †(5), †(6), †(7), †(8), †(9), †(10)) :
```

```
+ 15
+ 16 ...
+ 17 """
+ 18 function ntuple(f::F, n::Integer) where F
+ 19     t = n == 0 ? () :
+ 20     n == 1 ? (f(1),) :
+ 21     n == 2 ? (f(1), f(2)) :
+ 22     n == 3 ? (f(1), f(2), f(3)) :
+ 23     n == 4 ? (f(1), f(2), f(3), f(4)) :
+ 24     n == 5 ? (f(1), f(2), f(3), f(4), f(5)) :
+ 25     n == 6 ? (f(1), f(2), f(3), f(4), f(5), f(6)) :
+ 26     n == 7 ? (f(1), f(2), f(3), f(4), f(5), f(6), f(7)) :
+ 27
+ 28     n == 8 ? (f(1), f(2), f(3), f(4), f(5), f(6), f(7), f(8)) :
+ 29     n == 9 ? (f(1), f(2), f(3), f(4), f(5), f(6), f(7), f(8), f(9)) :
+ 30     n == 10 ? (f(1), f(2), f(3), f(4), f(5), f(6), f(7), f(8), f(9), f(10)) :
+ 31     _ntuple(f, n)
+ 32     return t
+ 33 end
+ 34
+ 35 function _ntuple(f, n)
+ 36     @_noinline_meta
+ 37     (n >= 0) || throw(ArgumentError(string("tuple length should be >= 0, got ", n)))
+ 38     ([f(i) for i = 1:n]...)
+ 39 end
+ 40
+ 41 # inferrable ntuple (enough for bootstrapping)
+ 42 ntuple(f, ::Val{0}) = ()
+ 43 ntuple(f, ::Val{1}) = (@_inline_meta; (f(1),))
+ 44 ntuple(f, ::Val{2}) = (@_inline_meta; (f(1), f(2)))
+ 45 ntuple(f, ::Val{3}) = (@_inline_meta; (f(1), f(2), f(3)))
+ 46 #
+ 47 @_inline function ntuple(f::F, ::Val{N}) where {F,N}
+ 48     N::Int
+ 49     (N >= 0) || throw(ArgumentError(string("tuple length should be >= 0, got ", N)))
+ 50     if @_generated
+ 51         quote
+ 52             @nexprs $N i → t.i = f(i)
+ 53             @ncall $N tuple t
+ 54         end
+ 55     else
+ 56         Tuple(f(i) for i = 1:N)
+ 57     end
+ 58 end
+ 59
+ 60 @_inline function fill_to_length(t::Tuple, val, ::Val{_N}) where {_N}
+ 61     M = length(t)
+ 62     N = _N::Int
+ 63     if M < N
+ 64         quote
+ 65             @_generated
+ 66             quote
+ 67                 @_noinline_meta
+ 68                 @nexprs $N i → t.i = val
+ 69                 @ncall $N tuple t
+ 70             end
+ 71         end
+ 72     end
+ 73 end
+ 74
+ 75 @_inline function tuple(f::F, ::Val{_N}) where {_N}
+ 76     N::Int
+ 77     (N >= 0) || throw(ArgumentError(string("tuple length should be >= 0, got ", N)))
+ 78     @_generated
+ 79     quote
+ 80         @_noinline_meta
+ 81         @nexprs $N i → t.i = f(i)
+ 82         @ncall $N tuple t
+ 83     end
+ 84 end
+ 85
+ 86 @_inline function _tuple(f, n)
+ 87     @_noinline_meta
+ 88     (n >= 0) || throw(ArgumentError(string("tuple length should be >= 0, got ", n)))
+ 89 end
+ 90
+ 91 # inferrable ntuple (enough for bootstrapping)
+ 92 ntuple(f, ::Val{0}) = ()
+ 93 ntuple(f, ::Val{1}) = (@_inline_meta; (f(1),))
+ 94 ntuple(f, ::Val{2}) = (@_inline_meta; (f(1), f(2)))
+ 95 ntuple(f, ::Val{3}) = (@_inline_meta; (f(1), f(2), f(3)))
+ 96 #
+ 97 @_inline function ntuple(f::F, ::Val{N}) where {F,N}
+ 98     N::Int
+ 99     (N >= 0) || throw(ArgumentError(string("tuple length should be >= 0, got ", N)))
+ 100    if @_generated
+ 101        quote
+ 102            @_nexprs $N i → t.i = f(i)
+ 103            @_ncall $N tuple t
+ 104        end
+ 105    else
+ 106        Tuple(f(i) for i = 1:N)
+ 107    end
+ 108 end
+ 109
+ 110 @_inline function fill_to_length(t::Tuple, val, ::Val{_N}) where {_N}
+ 111     M = length(t)
+ 112     N = _N::Int
+ 113     if M < N
+ 114         quote
+ 115             @_generated
+ 116             quote
+ 117                 @_noinline_meta
+ 118                 @_nexprs $N i → t.i = val
+ 119                 @_ncall $N tuple t
+ 120             end
+ 121         end
+ 122     end
+ 123 end
+ 124
+ 125 @_inline function tuple(f::F, ::Val{_N}) where {_N}
+ 126     N::Int
+ 127     (N >= 0) || throw(ArgumentError(string("tuple length should be >= 0, got ", N)))
+ 128     @_generated
+ 129     quote
+ 130         @_noinline_meta
+ 131         @_nexprs $N i → t.i = f(i)
+ 132         @_ncall $N tuple t
+ 133     end
+ 134 end
+ 135
+ 136 @_inline function _tuple(f, n)
+ 137     @_noinline_meta
+ 138     (n >= 0) || throw(ArgumentError(string("tuple length should be >= 0, got ", n)))
+ 139 end
+ 140
+ 141 # inferrable ntuple (enough for bootstrapping)
+ 142 ntuple(f, ::Val{0}) = ()
+ 143 ntuple(f, ::Val{1}) = (@_inline_meta; (f(1),))
+ 144 ntuple(f, ::Val{2}) = (@_inline_meta; (f(1), f(2)))
+ 145 ntuple(f, ::Val{3}) = (@_inline_meta; (f(1), f(2), f(3)))
+ 146 #
+ 147 @_inline function ntuple(f::F, ::Val{N}) where {F,N}
+ 148     N::Int
+ 149     (N >= 0) || throw(ArgumentError(string("tuple length should be >= 0, got ", N)))
+ 150     if @_generated
+ 151         quote
+ 152             @_nexprs $N i → t.i = f(i)
+ 153             @_ncall $N tuple t
+ 154         end
+ 155     else
+ 156         Tuple(f(i) for i = 1:N)
+ 157     end
+ 158 end
+ 159
+ 160 @_inline function fill_to_length(t::Tuple, val, ::Val{_N}) where {_N}
+ 161     M = length(t)
+ 162     N = _N::Int
+ 163     if M < N
+ 164         quote
+ 165             @_generated
+ 166             quote
+ 167                 @_noinline_meta
+ 168                 @_nexprs $N i → t.i = val
+ 169                 @_ncall $N tuple t
+ 170             end
+ 171         end
+ 172     end
+ 173 end
+ 174
+ 175 @_inline function tuple(f::F, ::Val{_N}) where {_N}
+ 176     N::Int
+ 177     (N >= 0) || throw(ArgumentError(string("tuple length should be >= 0, got ", N)))
+ 178     @_generated
+ 179     quote
+ 180         @_noinline_meta
+ 181         @_nexprs $N i → t.i = f(i)
+ 182         @_ncall $N tuple t
+ 183     end
+ 184 end
+ 185
+ 186 @_inline function _tuple(f, n)
+ 187     @_noinline_meta
+ 188     (n >= 0) || throw(ArgumentError(string("tuple length should be >= 0, got ", n)))
+ 189 end
+ 190
+ 191 # inferrable ntuple (enough for bootstrapping)
+ 192 ntuple(f, ::Val{0}) = ()
+ 193 ntuple(f, ::Val{1}) = (@_inline_meta; (f(1),))
+ 194 ntuple(f, ::Val{2}) = (@_inline_meta; (f(1), f(2)))
+ 195 ntuple(f, ::Val{3}) = (@_inline_meta; (f(1), f(2), f(3)))
+ 196 #
+ 197 @_inline function ntuple(f::F, ::Val{N}) where {F,N}
+ 198     N::Int
+ 199     (N >= 0) || throw(ArgumentError(string("tuple length should be >= 0, got ", N)))
+ 200     if @_generated
+ 201         quote
+ 202             @_nexprs $N i → t.i = f(i)
+ 203             @_ncall $N tuple t
+ 204         end
+ 205     else
+ 206         Tuple(f(i) for i = 1:N)
+ 207     end
+ 208 end
+ 209
+ 210 @_inline function fill_to_length(t::Tuple, val, ::Val{_N}) where {_N}
+ 211     M = length(t)
+ 212     N = _N::Int
+ 213     if M < N
+ 214         quote
+ 215             @_generated
+ 216             quote
+ 217                 @_noinline_meta
+ 218                 @_nexprs $N i → t.i = val
+ 219                 @_ncall $N tuple t
+ 220             end
+ 221         end
+ 222     end
+ 223 end
+ 224
+ 225 @_inline function tuple(f::F, ::Val{_N}) where {_N}
+ 226     N::Int
+ 227     (N >= 0) || throw(ArgumentError(string("tuple length should be >= 0, got ", N)))
+ 228     @_generated
+ 229     quote
+ 230         @_noinline_meta
+ 231         @_nexprs $N i → t.i = f(i)
+ 232         @_ncall $N tuple t
+ 233     end
+ 234 end
+ 235
+ 236 @_inline function _tuple(f, n)
+ 237     @_noinline_meta
+ 238     (n >= 0) || throw(ArgumentError(string("tuple length should be >= 0, got ", n)))
+ 239 end
+ 240
+ 241 # inferrable ntuple (enough for bootstrapping)
+ 242 ntuple(f, ::Val{0}) = ()
+ 243 ntuple(f, ::Val{1}) = (@_inline_meta; (f(1),))
+ 244 ntuple(f, ::Val{2}) = (@_inline_meta; (f(1), f(2)))
+ 245 ntuple(f, ::Val{3}) = (@_inline_meta; (f(1), f(2), f(3)))
+ 246 #
+ 247 @_inline function ntuple(f::F, ::Val{N}) where {F,N}
+ 248     N::Int
+ 249     (N >= 0) || throw(ArgumentError(string("tuple length should be >= 0, got ", N)))
+ 250     if @_generated
+ 251         quote
+ 252             @_nexprs $N i → t.i = f(i)
+ 253             @_ncall $N tuple t
+ 254         end
+ 255     else
+ 256         Tuple(f(i) for i = 1:N)
+ 257     end
+ 258 end
+ 259
+ 260 @_inline function fill_to_length(t::Tuple, val, ::Val{_N}) where {_N}
+ 261     M = length(t)
+ 262     N = _N::Int
+ 263     if M < N
+ 264         quote
+ 265             @_generated
+ 266             quote
+ 267                 @_noinline_meta
+ 268                 @_nexprs $N i → t.i = val
+ 269                 @_ncall $N tuple t
+ 270             end
+ 271         end
+ 272     end
+ 273 end
+ 274
+ 275 @_inline function tuple(f::F, ::Val{_N}) where {_N}
+ 276     N::Int
+ 277     (N >= 0) || throw(ArgumentError(string("tuple length should be >= 0, got ", N)))
+ 278     @_generated
+ 279     quote
+ 280         @_noinline_meta
+ 281         @_nexprs $N i → t.i = f(i)
+ 282         @_ncall $N tuple t
+ 283     end
+ 284 end
+ 285
+ 286 @_inline function _tuple(f, n)
+ 287     @_noinline_meta
+ 288     (n >= 0) || throw(ArgumentError(string("tuple length should be >= 0, got ", n)))
+ 289 end
+ 290
+ 291 # inferrable ntuple (enough for bootstrapping)
+ 292 ntuple(f, ::Val{0}) = ()
+ 293 ntuple(f, ::Val{1}) = (@_inline_meta; (f(1),))
+ 294 ntuple(f, ::Val{2}) = (@_inline_meta; (f(1), f(2)))
+ 295 ntuple(f, ::Val{3}) = (@_inline_meta; (f(1), f(2), f(3)))
+ 296 #
+ 297 @_inline function ntuple(f::F, ::Val{N}) where {F,N}
+ 298     N::Int
+ 299     (N >= 0) || throw(ArgumentError(string("tuple length should be >= 0, got ", N)))
+ 300     if @_generated
+ 311
```

```

# inferable ntuple (enough for bootstrapping)
ntuple(f, ::Val{0}) = ()
ntuple(f, ::Val{1}) = (@_inline_meta; (f(1),))
ntuple(f, ::Val{2}) = (@_inline_meta; (f(1), f(2)))
ntuple(f, ::Val{3}) = (@_inline_meta; (f(1), f(2), f(3)))

@inline function ntuple(f)::F, ::Val{N}) where {F,N}
    N::Int
    (N >= 0) || throw(ArgumentError(string("tuple length should be ≥ 0, got ", N)))
    if @generated
        quote
            @nexprs $N i → t_i = f(i)
            @ncall $N tuple t
        end
    else
        Tuple(f(i) for i = 1:N)
    end
end

```

All text set in JuliaMono.

JuliaMono is licensed with the  
SIL OpenFont licence.



Copyright Creative Commons  
CC BY-SA 3.0  
(Attribution-ShareAlike 3.0)

```

@inline function fill_to_length(t::Tuple, val, ::Val{_N}) where {_N}
    M = length(t)
    N = _N::Int
    M > N && throw(ArgumentError("input tuple of length $M, requested $N"))
    if @generated
        quote
            (t..., $(fill(:val, (_N::Int) - length(t.parameters))...))
        end
    else
        Tuple(val for i = 1:N)
    end
end

```