Default Project Name

1.0

Generated by Doxygen 1.12.0

Chapter 1

24-Game Solver

An enhanced version of the 24-game solver using recursive search and pruning.

This program allows users to solve the 24-game using the following features:

- Recursive search with pruning
- Progress bar to monitor search progress
- OpenMP parallelization for multi-core systems
- Optimization for commutative operations (addition and multiplication)

Users can either use default settings or customize them according to their needs. The program outputs include:

- The first valid solution for 24
- Detailed recursive steps and expressions

Additionally, the program supports solving the game with varying number of inputs and halts upon finding a valid solution.

1.0.1 Key Features

- Implements recursive search for the 24-game
- Supports up to 8 input numbers
- OpenMP for parallelization on larger input sizes
- Progress bar for visual feedback during the search process

1.0.2 How to Use

1. Compile the program with OpenMP support (`gfortran -fopenmp`).
2. Enter the number of numbers (between 1 and 8).
3. Provide the input numbers or card values (A=1, J=11, Q=12, K=13).
4. View the solution if found, or a message indicating no solution.

Chapter 2

Modules Index

2.1 Modules List

Here is a list of all modules with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Module Documentation

4.1 game24_module Module Reference

Functions/Subroutines

- subroutine convert_to_number (input_str, number, ios)

  Converts user input (cards or numbers) into numeric values.
- subroutine remove_decimal_zeros (str, result)

  Removes trailing zeros after the decimal point in a string.
- subroutine create_new_arrays (nums, exprs, idx1, idx2, result, new_expr, new_nums, new_exprs)

  Creates new arrays after performing an operation.
- subroutine update_progress_bar ()

  Updates and displays the horizontal percentage-based progress bar.
- recursive subroutine solve_24 (nums, exprs, found)

  Recursively solves the 24 game by trying all possible operations.

Variables

- integer, parameter max_limit = 8
- integer, parameter expr_len = 200
- integer(int64), parameter total_calls_n6 = 20000000_int64
- integer(int64), parameter total_calls_n7 = 2648275200_int64
- integer(int64), parameter total_calls_n8 = 444557593600_int64
- integer(int64) total_calls = 0
- integer(int64) completed_calls = 0
- integer last_percentage = -1
- integer, parameter progress_bar_width = 50
- character(len=1) carriage_return = char(13)
- logical show_progress = .false.

4.1.1 Function/Subroutine Documentation

4.1.1.1 convert_to_number()

```
subroutine game24_module::convert_to_number (
             character(len=*), intent(in) input_str,
             real, intent(out) number,
             integer, intent(out) ios)
```

Converts user input (cards or numbers) into numeric values.

Handles card values such as 'A', 'J', 'Q', 'K'.

Parameters

input_str	String representing the number or card value
number	The corresponding numeric value after conversion
ios	I/O status indicator (0 for success)

```
00089         implicit none
00090         character(len=*), intent(in) :: input_str
00091         real, intent(out)            :: number
00092         integer, intent(out)         :: ios
00093         character(len=1)             :: first_char
00094         real                         :: temp_number
00095
00096         ios = 0  ! Reset the I/O status to 0 (valid input by default)
00097         first_char = input_str(1:1)
00098
00099         select case (first_char)
00100         case ('A', 'a')
00101             number = 1.0
00102         case ('J', 'j')
00103             number = 11.0
00104         case ('Q', 'q')
00105             number = 12.0
00106         case ('K', 'k')
00107             number = 13.0
00108         case default
00109             read (input_str, *, iostat=ios) temp_number  ! Attempt to read a real number
00110
00111             ! If input is not a valid real number or is not an integer, set ios to 1
00112             if (ios /= 0 .or. mod(temp_number, 1.0) /= 0.0) then
00113                 ios = 1  ! Invalid input
00114             else
00115                 number = temp_number  ! Valid integer input
00116             end if
00117         end select
```

4.1.1.2 create_new_arrays()

```
subroutine game24_module::create_new_arrays (
              real, dimension(:), intent(in) nums,
              character(len=expr_len), dimension(:), intent(in) exprs,
              integer, intent(in) idx1,
              integer, intent(in) idx2,
              real, intent(in) result,
              character(len=expr_len), intent(in) new_expr,
              real, dimension(:), intent(out), allocatable new_nums,
              character(len=expr_len), dimension(:), intent(out), allocatable new_exprs)
```

Creates new arrays after performing an operation.

Parameters

nums	Input array of numbers
exprs	Input array of expressions
idx1	Index of the first element to remove
idx2	Index of the second element to remove
result	Result of the operation
new_expr	New expression string
new_nums	Output array of numbers with elements removed and result added
new_exprs	Output array of expressions with elements removed and new_expr added

```
00155         implicit none
00156         real, intent(in)                        :: nums(:)       ! Input: Array of numbers
00157         character(len=expr_len), intent(in)     :: exprs(:)      ! Input: Array of expressions
00158         integer, intent(in)                     :: idx1, idx2    ! Input: Indices of elements to
      remove
```

```
00159         real, intent(in)                          :: result        ! Input: Result of the operation
00160         character(len=expr_len), intent(in)       :: new_expr      ! Input: New expression
00161         real, allocatable, intent(out)            :: new_nums(:)   ! Output: New array of numbers
00162         character(len=expr_len), allocatable, intent(out) :: new_exprs(:) ! Output: New array of
      expressions
00163         integer                                   :: i, j, n       ! Loop counters and size of input
      arrays
00164
00165         n = size(nums)
00166         allocate (new_nums(n - 1))
00167         allocate (new_exprs(n - 1))
00168
00169         j = 0
00170         do i = 1, n
00171             if (i /= idx1 .and. i /= idx2) then
00172                 j = j + 1
00173                 new_nums(j) = nums(i)
00174                 new_exprs(j) = exprs(i)
00175             end if
00176         end do
00177
00178         ! Add the result of the operation to the new arrays
00179         new_nums(n - 1) = result
00180         new_exprs(n - 1) = new_expr
```

4.1.1.3 remove_decimal_zeros()

```
subroutine game24_module::remove_decimal_zeros (
             character(len=*), intent(in) str,
             character(len=*), intent(out) result)
```

Removes trailing zeros after the decimal point in a string.

Parameters

str	Input string that may contain trailing zeros
result	Output string with trailing zeros removed

```
00124         implicit none
00125         character(len=*), intent(in)  :: str       ! Input: String to remove zeros from
00126         character(len=*), intent(out) :: result    ! Output: String without trailing zeros
00127         integer                       :: i, len_str  ! Loop counter and string length
00128
00129         len_str = len_trim(str)
00130         result = adjustl(str(1:len_str))
00131
00132         ! Find the position of the decimal point
00133         i = index(result, '.')
00134
00135         ! If there's a decimal point, remove trailing zeros
00136         if (i > 0) then
00137             do while (len_str > i .and. result(len_str:len_str) == '0')
00138                 len_str = len_str - 1
00139             end do
00140             if (result(len_str:len_str) == '.') len_str = len_str - 1
00141             result = result(1:len_str)
00142         end if
```

4.1.1.4 solve_24()

```
recursive subroutine game24_module::solve_24 (
             real, dimension(:), intent(in) nums,
             character(len=expr_len), dimension(:), intent(in) exprs,
             logical, intent(inout) found)
```

Recursively solves the 24 game by trying all possible operations.

Utilizes OpenMP tasks for parallelization.

Parameters

nums	Array of numbers to use in the game
exprs	Array of string expressions representing the numbers
found	Logical flag indicating if a solution has been found

```
00226         use omp_lib
00227         implicit none
00228         real, intent(in)                            :: nums(:)       ! Input: Array of numbers
00229         character(len=expr_len), intent(in)         :: exprs(:)      ! Input: Array of expressions
00230         logical, intent(inout)                      :: found         ! Input/Output: Flag indicating if a
      solution is found
00231         integer                                     :: n             ! Size of the input arrays
00232         integer                                     :: i, j, op      ! Loop counters
00233         real                                        :: a, b, result  ! Temporary variables for
      calculations
00234         real, allocatable                           :: new_nums(:)   ! Temp array to store numbers after
      an operation
00235         character(len=expr_len), allocatable        :: new_exprs(:)  ! Temp array to store expressions
      after an operation
00236         character(len=expr_len)                     :: expr_a, expr_b, new_expr ! Temp variables for
      expressions
00237
00238         n = size(nums)
00239
00240         ! Increment the completed_calls counter and update progress bar
00241         if (show_progress) then
00242             !$omp atomic
00243             completed_calls = completed_calls + 1
00244             call update_progress_bar()
00245         end if
00246
00247         ! If a solution is found, return
00248         if (found) return
00249
00250         ! Base case: If only one number is left, check if it is 24
00251         if (n == 1) then
00252             if (abs(nums(1) - 24.0) < 1e-4) then
00253                 if (show_progress) then
00254                     write (*, '(A, F5.1, A)', advance='no') carriage_return//'['//repeat('=',
      progress_bar_width)//'] ', 100.0, '%'
00255                     write (*, '(A)') ''  ! Insert a blank line
00256                 end if
00257                 !$omp critical
00258                 write (*, '(A, A, A, F10.7, A)') 'Solution found:', trim(exprs(1)), '= 24 (', nums(1),
      ')'
00259                 found = .true.
00260                 !$omp end critical
00261             end if
00262             return
00263         end if
00264
00265         ! Iterate over all pairs of numbers
00266         do i = 1, n - 1
00267             do j = i + 1, n
00268                 a = nums(i)
00269                 b = nums(j)
00270                 expr_a = exprs(i)
00271                 expr_b = exprs(j)
00272
00273                 ! Iterate over all operators
00274                 do op = 1, 4
00275                     ! Avoid division by zero
00276                     if ((op == 4 .and. abs(b) < 1e-6)) cycle
00277
00278                     ! Perform the operation and create the new expression
00279                     select case (op)
00280                     case (1)
00281                         result = a + b
00282                         new_expr = '('//trim(expr_a)//'+'//trim(expr_b)//')'
00283                     case (2)
00284                         result = a - b
00285                         new_expr = '('//trim(expr_a)//'-'//trim(expr_b)//')'
00286                     case (3)
00287                         result = a * b
00288                         new_expr = '('//trim(expr_a)//'*'//trim(expr_b)//')'
00289                     case (4)
00290                         result = a / b
00291                         new_expr = '('//trim(expr_a)//'/'//trim(expr_b)//')'
00292                     end select
00293
00294                     ! Create new arrays with the selected numbers removed
00295                     call create_new_arrays(nums, exprs, i, j, result, new_expr, new_nums, new_exprs)
00296
```

```
00297                     ! For the first few recursion levels, create parallel tasks
00298                     if (n >= 6 .and. omp_get_level() < 2) then
00299                         !$omp task shared(found) firstprivate(new_nums, new_exprs)
00300                         call solve_24(new_nums, new_exprs, found)
00301                         !$omp end task
00302                     else
00303                         call solve_24(new_nums, new_exprs, found)
00304                     end if
00305
00306                     ! If a solution is found, deallocate memory and return
00307                     if (found) then
00308                         deallocate (new_nums)
00309                         deallocate (new_exprs)
00310                         return
00311                     end if
00312
00313                     ! Handle commutative operations only once
00314                     if (op == 1 .or. op == 3) cycle
00315
00316                     ! Swap operands for subtraction and division
00317                     if (op == 2 .or. op == 4) then
00318                         if (op == 4 .and. abs(a) < 1e-6) cycle  ! Avoid division by zero
00319
00320                         select case (op)
00321                         case (2)
00322                             result = b - a
00323                             new_expr = '('//trim(expr_b)//'-'//trim(expr_a)//')'
00324                         case (4)
00325                             result = b / a
00326                             new_expr = '('//trim(expr_b)//'/'//trim(expr_a)//')'
00327                         end select
00328
00329                         ! Create new arrays with the selected numbers removed
00330                         call create_new_arrays(nums, exprs, i, j, result, new_expr, new_nums,
      new_exprs)
00331
00332                         ! For the first few recursion levels, create parallel tasks
00333                         if (n >= 6 .and. omp_get_level() < 2) then
00334                             !$omp task shared(found) firstprivate(new_nums, new_exprs)
00335                             call solve_24(new_nums, new_exprs, found)
00336                             !$omp end task
00337                         else
00338                             ! Recursively call the solve_24 function with the new arrays
00339                             call solve_24(new_nums, new_exprs, found)
00340                         end if
00341
00342                         ! If a solution is found, deallocate memory and return
00343                         if (found) then
00344                             deallocate (new_nums)
00345                             deallocate (new_exprs)
00346                             return
00347                         end if
00348                     end if
00349
00350                 end do  ! End of operator loop
00351             end do  ! End of j loop
00352         end do  ! End of i loop
```

4.1.1.5 update_progress_bar()

`subroutine game24_module::update_progress_bar`

Updates and displays the horizontal percentage-based progress bar.

```
00185         implicit none
00186         real :: percentage
00187         integer :: filled_length
00188         character(len=progress_bar_width) :: bar
00189         integer :: int_percentage
00190
00191         if (total_calls == 0 .or. .not. show_progress) return  ! Avoid division by zero and check the
      flag
00192
00193         percentage = real(completed_calls) / real(total_calls) * 100.0
00194
00195         ! Ensure percentage does not exceed 100%
00196         if (percentage > 100.0) percentage = 100.0
00197
00198         ! Calculate integer percentage
00199         int_percentage = int(percentage)
00200
00201         ! Update progress bar only when percentage increases by at least 1%
00202         if (int_percentage > last_percentage) then
```

```
00203             last_percentage = int_percentage
00204
00205             ! Calculate the filled length of the progress bar
00206             filled_length = min(int(percentage / 100.0 * progress_bar_width), progress_bar_width)
00207
00208             ! Construct the progress bar string
00209             bar = repeat('=', filled_length)
00210             if (filled_length < progress_bar_width) then
00211                 bar = bar//'>'//repeat(' ', progress_bar_width - filled_length - 1)
00212             end if
00213
00214             ! Print the progress bar and integer percentage
00215             write (*, '(A, F4.1, A)', advance='no') carriage_return//'['//bar//'] ', percentage, '%'
00216             call flush (0)  ! Ensure output is displayed immediately
00217         end if
```

4.1.2 Variable Documentation

4.1.2.1 carriage_return

```
character(len=1) game24_module::carriage_return = char(13)
00055     character(len=1) :: carriage_return = char(13) ! Carriage return character
```

4.1.2.2 completed_calls

```
integer(int64) game24_module::completed_calls = 0
00052     integer(int64) :: completed_calls = 0       ! Number of completed recursive calls
```

4.1.2.3 expr_len

```
integer, parameter game24_module::expr_len = 200
00043     integer, parameter :: expr_len = 200   ! Maximum length for expressions
```

4.1.2.4 last_percentage

```
integer game24_module::last_percentage = -1
00053     integer :: last_percentage = -1             ! Last percentage reported
```

4.1.2.5 max_limit

```
integer, parameter game24_module::max_limit = 8
00042     integer, parameter :: max_limit = 8     ! Maximum allowed value for the number of inputs
```

4.1.2.6 progress_bar_width

```
integer, parameter game24_module::progress_bar_width = 50
00054     integer, parameter :: progress_bar_width = 50  ! Width of the progress bar
```

4.1.2.7 show_progress

```
logical game24_module::show_progress = .false.
00056     logical :: show_progress = .false.          ! Flag to show progress bar
```

4.1.2.8 total_calls

```
integer(int64) game24_module::total_calls = 0
00051     integer(int64) :: total_calls = 0          ! Total number of recursive calls
```

4.1.2.9 total_calls_n6

```
integer(int64), parameter game24_module::total_calls_n6 = 20000000_int64
00046     integer(int64), parameter :: total_calls_n6 = 20000000_int64
```

4.1.2.10 total_calls_n7

```
integer(int64), parameter game24_module::total_calls_n7 = 2648275200_int64
00047     integer(int64), parameter :: total_calls_n7 = 2648275200_int64
```

4.1.2.11 total_calls_n8

```
integer(int64), parameter game24_module::total_calls_n8 = 444557593600_int64
00048     integer(int64), parameter :: total_calls_n8 = 444557593600_int64
```

Chapter 5

File Documentation

5.1 game24_ultra.f90 File Reference

The main entry point for the 24-game solver, utilizing recursive search, progress bar, and OpenMP parallelization.

Modules

- module game24_module

Functions/Subroutines

- subroutine game24_module::convert_to_number (input_str, number, ios)

  Converts user input (cards or numbers) into numeric values.
- subroutine game24_module::remove_decimal_zeros (str, result)

  Removes trailing zeros after the decimal point in a string.
- subroutine game24_module::create_new_arrays (nums, exprs, idx1, idx2, result, new_expr, new_nums, new_exprs)

  Creates new arrays after performing an operation.
- subroutine game24_module::update_progress_bar ()

  Updates and displays the horizontal percentage-based progress bar.
- recursive subroutine game24_module::solve_24 (nums, exprs, found)

  Recursively solves the 24 game by trying all possible operations.
- program game24_ultra

Variables

- integer, parameter game24_module::max_limit = 8
- integer, parameter game24_module::expr_len = 200
- integer(int64), parameter game24_module::total_calls_n6 = 20000000_int64
- integer(int64), parameter game24_module::total_calls_n7 = 2648275200_int64
- integer(int64), parameter game24_module::total_calls_n8 = 444557593600_int64
- integer(int64) game24_module::total_calls = 0
- integer(int64) game24_module::completed_calls = 0
- integer game24_module::last_percentage = -1
- integer, parameter game24_module::progress_bar_width = 50
- character(len=1) game24_module::carriage_return = char(13)
- logical game24_module::show_progress = .false.

5.1.1 Detailed Description

The main entry point for the 24-game solver, utilizing recursive search, progress bar, and OpenMP parallelization.

Author

Gilbert Young

Date

2024/09/15

5.1.2 Function/Subroutine Documentation

5.1.2.1 game24_ultra()

```
program game24_ultra
```

Index