

Chat history

[ChatGPT](#)

[New chat](#)

Yesterday

[HPC Homework 9 Summary](#)

Previous 7 Days

[Fatigue Tensile Testing Specimens](#)

[Ultrasonic Advancements in Diagnosis](#)

[Phased Array Ultrasonics Explained](#)

[Fatigue Testing Guidelines](#)

Previous 30 Days

[Ultrasonic Inspection Advances](#)

[Bulk Wave: Single vs. Multiple](#)

[Non-Stick Coating Effectiveness Assessment](#)

[New chat](#)

[Confidence Interval A90/95](#)

[Compile Numba function.](#)

[Wheeler-Feynman Absorber Theory](#)

[Geometric Series Summation: 250](#)

[Python Environment Check](#)

[Rework Assignment 6](#)

[Social Media's Mental Impact](#)

[Paris Law Crack Growth](#)

[Numerical Integration Methods](#)

[Easy-to-Cook Meal Ideas](#)

[Error in NumPy division.](#)

[Gaussian Quadrature Function in Python](#)

[Sample Caps for Polished Microscopy](#)

[More Time for Research](#)

[Brain Injury Assessment: Questions](#)

[Letter of Recommendation Outline](#)

Upgrade planGet GPT-4, DALL·E, and more

03

James Smith

Close sidebar

ChatGPT 3.5

03

You

HPC Homework 9

ChatGPT

ChatGPT

Sure, I can help with your HPC homework. What specific questions or topics do you need assistance with for Homework 9?

03

You

Teach me how to code a linearSearch function in a f90 fortran file titled "searchutils" . Here is a description of the linear search algorithm: Linear search is an algorithm that sequentially checks the elements of an array until the desired element is found, or until all elements have been checked. This algorithm is robust in the sense that it works on sorted and unsorted arrays.

ChatGPT

ChatGPT

Sure, I can help you with that. Below is a simple Fortran code for a linear search function in a file named "searchutils.f90":

fortran

Copy code

```
module searchutils
  implicit none
```

contains

```
! Function to perform linear search
function linearSearch(arr, n, key) result(index)
  integer, intent(in) :: arr(:) ! Input array
  integer, intent(in) :: n ! Size of the array
  integer, intent(in) :: key ! Element to search for
  integer :: index ! Index of the found element

  index = -1 ! Initialize index to -1 (indicating not found)
  do i = 1, n
    if (arr(i) == key) then
      index = i ! Element found, update index
      exit ! Exit loop
    end if
  end do
end function linearSearch
```

```
end module searchutils
```

Explanation:

- `module searchutils`: This is a Fortran module which encapsulates the linear search function.
- `function linearSearch`: This is the linear search function. It takes three arguments:
 - `arr(:)`: The input array to search.
 - `n`: The size of the array.
 - `key`: The element to search for.
- `result(index)`: This indicates that the function returns an integer `index`.
- `index = -1`: Initialize the index to -1, indicating that the key hasn't been found yet.
- `do i = 1, n`: Loop through each element of the array.
- `if (arr(i) == key) then`: Check if the current element is equal to the key.
- `index = i`: If the element is found, update the index and exit the loop.
- `exit`: Exit the loop once the key is found.

You can use this function in another Fortran program by including the module and calling the function. Here's an example program that uses this module:

fortran

Copy code

```
program test_linear_search
  use searchutils
  implicit none

  integer :: arr(10) = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
  integer :: key, index

  ! Search for key = 6 in arr
  key = 6
  index = linearSearch(arr, size(arr), key)

  if (index /= -1) then
    print *, "Key ", key, " found at index ", index
  else
    print *, "Key ", key, " not found"
  end if
end program test_linear_search
```

In this example, we search for the key 6 in the array `arr`. The output will indicate whether the key was found and at which index.

You

implement the linear and binary search methods for arrays of real values of 8 bytes, respectively. The function must report the index in Fortran-style indexing (1-based). In the case the element is not found in the array, the function must return -1

ChatGPT

ChatGPT

Here's the implementation of both linear and binary search methods for arrays of real values in Fortran. Remember that in Fortran, arrays are typically 1-indexed, so the returned index will be 1-based. If the element is not found, the function will return -1.

fortran

Copy code

```
module searchutils
```

```
  implicit none
```

contains

```
  ! Linear search function for real arrays
```

```
  function linearSearchReal(arr, n, key) result(index)
```

```
    real(8), intent(in) :: arr(:) ! Input array
```

```
    integer, intent(in) :: n ! Size of the array
```

```
    real(8), intent(in) :: key ! Element to search for
```

```
    integer :: index ! Index of the found element
```

```
    index = -1 ! Initialize index to -1 (indicating not found)
```

```
    do i = 1, n
```

```
      if (arr(i) == key) then
```

```
        index = i ! Element found, update index
```

```
        exit ! Exit loop
```

```
      end if
```

```
    end do
```

```
  end function linearSearchReal
```

```
  ! Binary search function for real arrays
```

```
  function binarySearchReal(arr, n, key) result(index)
```

```
    real(8), intent(in) :: arr(:) ! Input array (sorted)
```

```
    integer, intent(in) :: n ! Size of the array
```

```
    real(8), intent(in) :: key ! Element to search for
```

```
    integer :: index ! Index of the found element
```

```
    integer :: low, high, mid ! Indices for binary search
```

```

index = -1 ! Initialize index to -1 (indicating not found)
low = 1 ! Initialize low index
high = n ! Initialize high index

do while (low <= high)
mid = (low + high) / 2
if (arr(mid) == key) then
index = mid ! Element found, update index
exit ! Exit loop
else if (arr(mid) < key) then
low = mid + 1 ! Search in the right half
else
high = mid - 1 ! Search in the left half
end if
end do
end function binarySearchReal

end module searchutils

```

Explanation:

- `function linearSearchReal`: This function performs a linear search on a real array. It takes three arguments: the array `arr`, its size `n`, and the `key` to search for. It returns the index of the found element (1-based) or -1 if the element is not found.
- `function binarySearchReal`: This function performs a binary search on a sorted real array. It takes the same arguments as `linearSearchReal`. It returns the index of the found element (1-based) or -1 if the element is not found.
- In both functions, the real values are of type `real(8)`, which typically represent double precision real numbers.

You can use these functions similarly as in the previous example, by including the module `searchutils` and calling the respective functions. Ensure that if you are using `binarySearchReal`, your array must be sorted beforehand.

03

You

is the indentation at the beginning necessary?

ChatGPT

ChatGPT

No, the indentation at the beginning of the function declarations in Fortran is not necessary. It's more of a style convention for readability. Here's the code without the indentation:

fortran

Copy code

```
module searchutils
implicit none
```

```
contains
```

```
! Linear search function for real arrays
function linearSearchReal(arr, n, key) result(index)
  real(8), intent(in) :: arr(:) ! Input array
  integer, intent(in) :: n ! Size of the array
  real(8), intent(in) :: key ! Element to search for
  integer :: index ! Index of the found element

  index = -1 ! Initialize index to -1 (indicating not found)
  do i = 1, n
    if (arr(i) == key) then
      index = i ! Element found, update index
      exit ! Exit loop
    end if
  end do
end function linearSearchReal
```

```
! Binary search function for real arrays
function binarySearchReal(arr, n, key) result(index)
  real(8), intent(in) :: arr(:) ! Input array (sorted)
  integer, intent(in) :: n ! Size of the array
  real(8), intent(in) :: key ! Element to search for
  integer :: index ! Index of the found element
  integer :: low, high, mid ! Indices for binary search

  index = -1 ! Initialize index to -1 (indicating not found)
  low = 1 ! Initialize low index
  high = n ! Initialize high index

  do while (low <= high)
    mid = (low + high) / 2
    if (arr(mid) == key) then
      index = mid ! Element found, update index
      exit ! Exit loop
    end if
  end do
end function binarySearchReal
```

```

    else if (arr(mid) < key) then
    low = mid + 1 ! Search in the right half
    else
    high = mid - 1 ! Search in the left half
    end if
    end do
end function binarySearchReal

end module searchutils

```

This style is perfectly valid in Fortran. It's good to keep your code consistent with a certain style, but Fortran is not as strict about indentation as some other languages are.

03

You

How can I remove the indentations in vim?

ChatGPT

ChatGPT

In Vim, you can remove leading whitespace (including indentation) by using the `:%left` command. Here's how you can do it:

1. Open your Fortran file in Vim.
2. Enter command mode by pressing `Esc`.
3. Type `:%left` and press `Enter`.

This command will remove leading whitespace (indentation) from all lines in the file.

If you want to remove leading whitespace only from a specific range of lines, you can specify the range before `:%left`. For example, to remove indentation only from lines 1 to 10, you can type `:1,10left`.

After removing the indentations, save the changes by typing `:w` and press `Enter`, or `:x` to save and exit, or `:q!` to exit without saving if you made a mistake.

03

You

What if I only want to remove 5 spaces from each line?

ChatGPT

ChatGPT

To remove only 5 spaces from the beginning of each line in Vim, you can use the substitute command with a regular expression. Here's how you can do it:

1. Open your Fortran file in Vim.
2. Enter command mode by pressing `Esc`.
3. Type the following command and press `Enter`:

vim

Copy code

```
:%s/^  //
```

Explanation:

- `%s/` : Starts a substitution command that will apply to the entire file.
- `^` : Matches the beginning of each line.
- : Represents five spaces. Ensure you have an actual space character here.
- `//` : Replaces the matched pattern with nothing, effectively removing it.

So `:%s/^ //` will remove five spaces from the beginning of each line in the file.

If you're typing it directly in Vim and not copying and pasting, ensure you press the space bar five times to represent five spaces.

After running this command, if you find there are more than five spaces on some lines or if there are tabs mixed with spaces, you can use the following more general command:

vim

Copy code

```
:%s/^\s\{1,5}//
```

Explanation:

- `\s\{1,5}` : Matches between 1 and 5 whitespace characters (spaces or tabs) at the beginning of each line.

- `%s/^\s\{1,5}\s//` : This substitutes any sequence of 1 to 5 leading whitespace characters with nothing.

After removing the spaces, save the changes by typing `:w` and press `Enter`, or `:x` to save and exit, or `:q!` to exit without saving if you made a mistake.

03

You

is fortran case sensitive

ChatGPT

ChatGPT

Fortran is traditionally case insensitive for variable names, but it is case sensitive for keywords and intrinsic functions.

This means that you can write your variable names using any combination of uppercase and lowercase letters, and they will be treated as the same variable. For example, `MyVariable`, `myvariable`, and `MYVARIABLE` are all considered the same variable.

However, Fortran keywords and intrinsic functions are case sensitive. For example, `INTEGER`, `integer`, and `Integer` are not equivalent.

Here's an example to illustrate this:

fortran

Copy code

```
program case_sensitive_example
  integer :: INTEGER, MyVariable
  INTEGER = 5
  MyVariable = 10
  print *, INTEGER
  print *, myvariable
end program case_sensitive_example
```

In this program, `INTEGER` and `MyVariable` are declared as variables. `INTEGER` is also used as a variable name, but it's not confused with the Fortran keyword `INTEGER`. The output will be:

Copy code

5
10

So, while Fortran is generally not case sensitive for variables, it is case sensitive for keywords and intrinsic functions.

03

You

is it case sensitive for the module or implicit lines

ChatGPT

ChatGPT

Fortran is not case-sensitive for the `MODULE` or `IMPLICIT` lines.

- **MODULE:** You can write `MODULE`, `module`, or `Module` - all are equivalent.
- **IMPLICIT:** Similarly, you can use `IMPLICIT NONE`, `implicit none`, or `Implicit None`.

For example:

fortran

Copy code

```
MODULE MyModule
  IMPLICIT NONE
  INTEGER :: MyVariable
END MODULE MyModule
```

This is perfectly valid Fortran syntax.

However, note that the actual names of entities within Fortran, such as variables and procedures, are case-insensitive. So, while the keywords `MODULE` and `IMPLICIT` themselves can be written in any case, the names you define inside a module (like `MyModule`) are not case-sensitive once declared.

03

You

how many spaces is indentation in fortran

ChatGPT

ChatGPT

In Fortran, indentation is not specified by a specific number of spaces like in some other languages such as Python, where indentation is typically four spaces. Instead, indentation in Fortran is used for readability and to denote the structure of the program, but the number of spaces used for each indentation level is not standardized.

Traditionally, indentation in Fortran code is done using multiple spaces (typically 2 to 4 spaces) to visually distinguish blocks of code. Some programmers might prefer a larger number of spaces for indentation, while others might prefer fewer.

For example, indentation might look like this:

fortran

Copy code

```
program main
  implicit none

  integer :: i

  do i = 1, 10
    if (i > 5) then
      print *, "i is greater than 5"
    else
      print *, "i is less than or equal to 5"
    end if
  end do

end program main
```

In this example, the `implicit none` and `do`, `if`, and `end do` statements are indented to visually denote their respective blocks.

The key is to maintain consistency throughout your codebase, regardless of the specific number of spaces used for each indentation level. This consistency helps in readability and understanding the code structure.

You

does the implicit none line need to be indented

ChatGPT

ChatGPT

No, the `IMPLICIT NONE` line doesn't need to be indented. It's a statement that applies globally to the entire program, and it's typically placed at the beginning of the program, before any other statements.

Here's an example:

fortran

Copy code

```
program main
  implicit none

  ! Variable declarations
  integer :: i, j
  real :: x, y

  ! Executable statements
  i = 5
  j = 10
  x = 3.14
  y = 2.718

  ! Output
  print *, "The value of i is: ", i
  print *, "The value of j is: ", j
  print *, "The value of x is: ", x
  print *, "The value of y is: ", y
end program main
```

In this example, `IMPLICIT NONE` is placed at the beginning of the program without any indentation. This is a common style to indicate that it's a global directive affecting the entire program.

You

how do I copy paste several lines in vim

ChatGPT

ChatGPT

In Vim, you can copy and paste several lines using visual mode. Here's a step-by-step guide:

1. Enter Visual Mode: Move your cursor to the first character of the first line you want to copy. Press `v` to enter visual mode. This will start visual mode in character-wise selection. Alternatively, you can press `V` (capital v) to enter line-wise selection. This selects entire lines at a time.
2. Select Text: Move your cursor to the last character of the last line you want to copy. If you're in character-wise visual mode (`v`), the selection will expand character by character as you move the cursor. If you're in line-wise visual mode (`V`), the entire lines will be selected as you move the cursor.
3. Copy: Once you've selected the desired lines, press `y` to copy (yank) them. If you want to cut (delete) the selected lines instead of copying, press `d` instead of `y`.
4. Move Cursor: Move your cursor to where you want to paste the copied lines.
5. Paste: Press `p` to paste the copied lines below the current line. If you want to paste above the current line, press `Shift + p` (uppercase P) instead.

That's it! You've copied and pasted several lines in Vim. Here's a quick summary:

- `v`: Enter visual mode for character-wise selection.
- `V`: Enter visual mode for line-wise selection.
- `y`: Copy (yank) selected text.
- `d`: Cut (delete) selected text.
- `p`: Paste below the current line.
- `Shift + p` (uppercase P): Paste above the current line.

03

You

Check this for errors: MODULE searchutils

implicit none

contains

! Description: Function that finds the location (idx) of a value x
! in an array using the linear search algorithm.

!

! Find idx such that `arr(idx) == x`

!

```

FUNCTION linearSearch(arr, n, x) RESULT(idx)
  REAL(8) :: arr(n) ! Array to search
  INTEGER :: n      ! Number of elements in array.
  REAL(8) :: x      ! Value to search for in array.
  INTEGER :: idx     ! Result of the search. [arr(idx) == x]

  idx = -1 ! Initialize index to -1 (indicating not found)
  DO i = 1, n
    IF (arr(i) == x) THEN
      idx = i
      EXIT
    END IF
  END DO
END FUNCTION linearSearch

```

! Description: Function that finds the location (idx) of a value x
! in a sorted array using the binary search algorithm.
!
! Find idx such that arr(idx) == x
!

```

FUNCTION binarySearch(arr, n, x) RESULT(idx)
  REAL(8) :: arr(n) ! Array to search
  INTEGER :: n      ! Number of elements in array.
  REAL(8) :: x      ! Value to search for in array.
  INTEGER :: idx     ! Result of the search. [arr(idx) == x]
  INTEGER :: low, high, mid ! Indices for binary search

  idx = -1
  low = 1
  high = n

  DO WHILE (low <= high)
    mid = (low + high) / 2
    IF (arr(mid) == x) THEN
      idx = mid
      EXIT
    ELSE IF (arr(mid) < x) THEN
      low = mid + 1
    ELSE
      high = mid - 1
    END IF
  END DO
END FUNCTION binarySearch

```

END MODULE searchutils

ChatGPT

ChatGPT

Your code looks mostly good, but there are a few issues and improvements to be made:

1. **Parameter Passing:** In Fortran, array parameters should not be declared with their size in the subroutine's argument list. Instead, you can specify the size using the `DIMENSION` attribute.
2. **Correcting DO loop index:** In Fortran, it's good practice to declare the loop index (`i`, `j`, etc.) locally within the subroutine or function.
3. **Improvement in Binary Search:** The binary search algorithm typically uses integer indices rather than real numbers. So, we should use `INTEGER` instead of `REAL(8)` for the `arr` array.
4. **Indentation:** Ensure consistent indentation for better readability.

Here's the corrected version:

fortran

Copy code

```
MODULE searchutils
```

```
  IMPLICIT NONE
```

```
CONTAINS
```

```
  ! Linear search function
```

```
  FUNCTION linearSearch(arr, n, x) RESULT(idx)
```

```
    REAL(8), INTENT(IN) :: arr(:) ! Array to search
```

```
    INTEGER, INTENT(IN) :: n ! Number of elements in array.
```

```
    REAL(8), INTENT(IN) :: x ! Value to search for in array.
```

```
    INTEGER :: idx ! Result of the search. [arr(idx) == x]
```

```
    INTEGER :: i ! Loop index
```

```
    idx = -1 ! Initialize index to -1 (indicating not found)
```

```
    DO i = 1, n
```

```
      IF (arr(i) == x) THEN
```

```
        idx = i
```

```
      EXIT
```

```
    END IF
```

```
  END DO
```

```

END FUNCTION linearSearch

! Binary search function
FUNCTION binarySearch(arr, n, x) RESULT(idx)
REAL(8), INTENT(IN) :: arr(:) ! Array to search
INTEGER, INTENT(IN) :: n ! Number of elements in array.
REAL(8), INTENT(IN) :: x ! Value to search for in array.
INTEGER :: idx ! Result of the search. [arr(idx) == x]

INTEGER :: low, high, mid ! Indices for binary search
INTEGER :: i ! Loop index

idx = -1
low = 1
high = n

DO WHILE (low <= high)
mid = (low + high) / 2
IF (arr(mid) == x) THEN
idx = mid
EXIT
ELSE IF (arr(mid) < x) THEN
low = mid + 1
ELSE
high = mid - 1
END IF
END DO
END FUNCTION binarySearch

END MODULE searchutils

```

These changes should address the issues in your code and ensure correct behavior.

ChatGPT can make mistakes. Consider checking important information.

?

ChatGPT

Get citation