计算物理作业1

谢昀城,22307110070

1.作业1

1.1 题目描述

在0到200内找方程a,b,c,d,e的整数解: $a^5+b^5+c^5+d^5=e^5$

1.2程序描述

显然，方程有平凡解:$a=b=c=d=e=0$和$a=e,b=c=d=0$ 。且a,b,c,d对称，因此不妨设
$0\leq a\leq b\leq c\leq d\leq e\leq 200$。因此，只需在0-200中满足前述关系循环$a,b,c,d,e$，且让$a,b,e$循环下界为1即可

本题源文件为findroot.f90，并且有已编译文件findroot.exe

1.3 伪代码

本题伪代码如下:

Algorithm Find integer solutions for $a^5+b^5+c^5+d^5=e^5$

```
DECLARE a, b, c, d, e as integers with upper limit of at least 10^12
for e ← 1 to 200 do
    for d ← 0 to e do
        for c ← 0 to d do
            for b ← 1 to c do
                for a ← 1 to b do
                    if a^5 + b^5 + c^5 + d^5 = e^5 then
                        PRINT "a =", a, "b =", b, "c =", c, "d =", d, "e =", e
                    end if
                end for
            end for
        end for
    end for
end for
```

1.4 程序执行结果

以下为程序实际运行截图:

2.作业

2.1作业描述

24点游戏是儿时玩的主要益智类游戏之一，玩法为:从一副扑克中抽取4张牌，对4张牌使用加减乘除中的任何方法，使计算结果为24。例如，2,3,4,6通过$((((4+6)-2)*3)=24)$，最快算出24者胜。采用Fortran90编程求解24点游戏的解。

2.2 程序描述

程序主要思路为首先从初始的1×4的数组中中抽取两个，对其应用加减乘除左乘左除共6中运算，得到的结果与剩余2个数存入一个1×3数组中，而其所有组合存入一个$6C_4^2\times 3$的二维数组中。接着不断迭代前述过程，直至最后数组列数减小为1。最后，查找其中值为24的元素即得到输入24点的解法。

在具体实现上，我们定义calculator24的类型来同时储存数字的值和表达式以记录24点的运算过程，并定义了calculate函数来对两个calculator24的类型的变量进行前述的6种运算。定义了generate_pairs函数对一个$n\times m$的calculator24数组输入计算其每一行的所有元素组合结果，得到一个$nC_m^2\times m$数组。定义reduce_pair_step函数对一个$n\times m$的calculator24数组的每一行的前两个元素应用6种运算，得到一个$6*n\times(m-1)$的数组。最后在reduce_pair函数中，反复调用generate_pairs和reduce_pair_step直到数组列数为1，再由match_value函数查找其中值为24的元素并打印其表达式可。

本程序源文件为solution24.f90,以及已编译文件solution.exe

本程序结构拥有较好的可拓展性，简单修改后便可以接受不同数量的数字输入（如使用5个数计算24点）

2.3伪代码

- 主程序伪代码:

Algorithm Main Program

procedure MAIN()
  $numElements \leftarrow 4$ *//Number of input elements*
  $calValue \leftarrow 24.0$ *//Target value for calculation*
  USERINPUT: $inputArray$
  $typeArray \leftarrow \text{InitArray}(inputArray)$ *//Initialize array to type of calculator24*
  $reducedPair \leftarrow \text{ReducePair}(typeArray)$ *//Iterative calculation*
  $answer \leftarrow \text{MatchValue}(reducedPair[:,1], calValue)$ *//Match result*
  print $answer$
end procedure

- calculator24类型的定义:

Algorithm calculator24 Module

MODULE calculator24_module
DEFINE TYPE $calculator24$
$value$ (real) *//Stores numerical value*
$expression$ (string, dynamic length) *//Stores the calculation expression history*
METHOD print()

Print $expression = value$ *//This will be achieved by defining a subroutine and binding it to the pointer 'print'.*
END MODULE

- **calculate函数的伪代码:**

Algorithm Calculate Function

function CALCULATE($x1$, $x2$, $symbol$)
INPUT: $x1$ (calculator24 object), $x2$ (calculator24 object), $symbol$ (integer)
OUTPUT: re (calculator24 object)
if $symbol = 1$ **then** *//Addition*
$re.value \leftarrow x1.value + x2.value$
$re.expression \leftarrow$ "(" $+ x1.expression +$ " + " $+ x2.expression +$ ")"
else if $symbol = 2$ **then** *//Subtraction*
$re.value \leftarrow x1.value - x2.value$
$re.expression \leftarrow$ "(" $+ x1.expression +$ " − " $+ x2.expression +$ ")"
else if $symbol = 3$ **then** *//Multiplication*
$re.value \leftarrow x1.value * x2.value$
$re.expression \leftarrow x1.expression +$ " ∗ " $+ x2.expression$
else if $symbol = 4$ **then** *//Division*
$re.value = x1.value/x2.value$
$re.expression \leftarrow x1.expression +$ "/" $+ x2.expression$
else if $symbol = 5$ **then** *//Left Subtraction*
$re.value \leftarrow x2.value - x1.value$
$re.expression \leftarrow$ "(" $+ x2.expression +$ " − " $+ x1.expression +$ ")"
else if $symbol = 6$ **then** *//Left Division*
$re.value \leftarrow x2.value/x1.value$
if / or * in $x1.expeession$ **then**
$re.expression \leftarrow x2.expression +$ "/" $+$ "(" $+ x1.expression +$ ")"
else
$re.expression \leftarrow x2.expression +$ "/" $+ x1.expression$
end if
end if
return re
end function

- **generate_pairs函数的伪代码:**

Algorithm GeneratePairs

function GENERATEPAIRS(*//Given an $n \times m$ calculator24 type 2D array, return the C_m^2 results for each row.* $array$)
INPUT: $array$ (2D array of calculator24)
OUTPUT: $pairs$ (2D array of calculator24)
$numElements \leftarrow \text{size}(array, 2)$
$rows \leftarrow \text{size}(array, 1)$
$numPairs \leftarrow rows \times (numElements \times (numElements - 1)/2)$
allocate($pairs[numPairs, numElements]$) *//Calculate the number of binary combinations*
$k \leftarrow 1$
for $r \leftarrow 1$ to $rows$ **do**
for $i \leftarrow 1$ to $numElements - 1$ **do**
for $j \leftarrow i + 1$ to $numElements$ **do**
$p1 \leftarrow 3$
$p2 \leftarrow 1$
for $m \leftarrow 1$ to $numElements$ **do**
if $m = i$ or $m = j$ **then** *//Fill the first two columns with permuted elements*

```
                pairs[k, p2] ← array[r, m]
                p2 ← p2 + 1
              else                                                    //Pad other elements
                pairs[k, p1] ← array[r, m]
                p1 ← p1 + 1
              end if
            end for
            k ← k + 1
          end for
        end for
      end for
    end function
```

- **reduce_pair_step函数的伪代码:**

```
Algorithm ReducePairStep
    function REDUCEPAIRSTEP(array)
        INPUT: array (2D array of calculator24)
        OUTPUT: reducedPair (new rows × 6, cols − 1 2D array of calculator24)
        rows ← size(array, 1)
        cols ← size(array, 2)
        if cols < 2 then                                              //If there is only one column, return as is
            allocate(reducedPair[rows, cols])
            reducedPair ← array
            return
        else
            allocate(reducedPair[rows × 6, cols − 1])
            k ← 1
            for i ← 1 to rows do
                for symbol ← 1 to 6 do                                //Traverse 6 types of calculations for each row
                    re ← Calculate(array[i, 1], array[i, 2], symbol)
                    reducedPair[k, 1] ← re
                    reducedPair[k, 2 : cols − 1] ← array[i, 3 : cols]
                    k ← k + 1
                end for
            end for
        end if
    end function
```

- **reduce_pair函数的伪代码:**

```
Algorithm ReducePair
    function REDUCEPAIR(array)
        INPUT: array (2D array of calculator24)
        OUTPUT: reducedPair (n ×1 2D array of calculator24)
        maxIter ← 100                                                 //Limit maximum number of iterations
        reducedPair ← ReducePairStep(GeneratePairs(array))
        for i ← 1 to maxIter do
            reducedPair ← ReducePairStep(GeneratePairs(reducedPair))
            if size(reducedPair, 2) = 1 then                          //If only one column remains, return
                return
            end if
        end for
    end function
```

- **match_valur的伪代码：**

Algorithm MatchValue

function MatchValue($array$, $value$)
INPUT: $array$ (1D array of calculator24), $value$ (real)
OUTPUT: re (calculator24)
$eps \leftarrow 0.001$ *//Tolerance*
$n \leftarrow \text{size}(array, 1)$
for $i \leftarrow 1$ **to** n **do**
if $\text{abs}(array[i].value - value) < eps$ **then**
$re \leftarrow array[i]$
return
end if
end for
$re.expression \leftarrow$ "$none$" *//If not found, set expression to none*
end function

2.4输入输出实例

对本程序需要用户输入4个待求解的数字，程序将返回由该组数字求解24点的方法。下表为不同输入时程序输出结果:

index	Input	Output
1	3,3,8,8	$8/(3-8/3)=24.00$
2	1,2,11,1	$(1+11)*1*2=24.00$
3	10,11,12,13	$(((10-11)+12)+13)=24.00$
4	5,6,7,8	$((5+7)-8)*6=24.00$
5	1,1,2,1	No solution

以下为程序运行截图:


C:\Users\ASUS\Deskt
Enter the elements:
1
1
2
1
Calculating...
The answer for 1 1 2 1 is:
No solution
请按任意键继续. . .