Aufgabe 1: **Fakultät / Binomealkoeffizient**

- Implementiere $n!=1*2*3*...*n=\prod_{i=1}^{n} i$ analog zur Implementierung von $\sum_{i=1}^{n} i$ in mySum aus der Präsentation / aus dem Skript. Bei dieser Implementation ist auch 0! kein Problem.
- Der Parameter der Funktion fak ist ein kleiner ganzzahliger Wert.
- Der Rückgabewert wird sehr schnell sehr groß, deshalb muss er einen großen Wertebereich umfassen. Integer als Datentyp ist hier nicht sinnvoll, besser real*8 (größter Wert ca. 10^{308})
- Hauptprogramm kommt in fakultaet.f und besteht aus:
  - der Eingabe, d.h. Ausgabe des abzufragenden Wertes z.B. „Fakultät von?“ und dem Einlesen des Wertes in eine Variable (ganzzahl).
  - Dem Aufruf von fak mit der eingelesenen Variablen als Parameter und der Ausgabe des Rückgabewertes.
- Funktion fak kommt in fak.f, Implementierung s.o.

- Binomealkoeffizient
  - Einlesen der Faktoren des Binomealkoeffizienten (n und k)
  - Berechnung und Ausgabe
  - Berechnung: $\binom{n}{k}=\frac{n!}{k!(n-k)!}$ also binom=fak(n)/(fak(k)*fak(n-k))

Aufgabe 2: **Dateizugriff**

- Einlesen von Intervallgrenzen, z.B. min und max so wie der Schrittweite (schritt)
- Das Intervall und die Schrittweite sind keine ganzen Zahlen, also min, max und schritt sind real (oder real*8).
- Die Laufvariablen in DO-Schleifen sind ganze Zahlen, wir benötigen eine Umrechnung der fließkomma Werte in ganze Zahlen und umgekehrt:
- Ich verwende folgende Variablen: i=integer Laufvariable, ri=real „Laufvariable“, 0=untere Grenze (integer), imax= obere Grenze (integer)
- Umrechnung:
  - ri=min+schritt*i
  - imax=(max-min)/schritt
- Test der Grenzen: bei i=0 ist ri=min+schritt*0=min, also OK.
  Bei i=imax ist ri=min+schritt*imax=min+schritt*(max-min)/schritt=min+(max-min)=max, also auch OK
- In der DO-Schleife: Ausgabe von ri, sin(ri), cos(ri) und tan(ri) mit einem formatierten print.
- Format: (4E15.6) Warum?

Aufgabe 3: **Plot**

- Zur Nutzung von xmgr gibt es ein Video
- Zur Nutzung von gnuplot gibt es auch ein Video.

Aufgabe 4: **Trapezregel**

- Eingelesen werden: Integrationsbereich a und b, Anzahl Intervalle n
- Verwende Formel (8), wegen Δx benötigen wir auch Formel (3) und die Formel für die Stützpunkte x_k.
- Die Summation aus Formel (8) wird wieder mit einer DO-Schleife berechnet.
- Wikipedia-Artikel: https://de.wikipedia.org/wiki/Trapezregel

Aufgabe 5: **Matrix-Multiplikation (Teil 1)**

- In Fortran sind Felder statisch → die Größe (hier 50) wird im Quellcode festgelegt und ist fix.
- Verwenden Sie die Bezeichnungen aus der Aufgabe, d.h. Variablen und Verwendungszweck wie in der Aufgabe.
- Lassen Sie die Vektorlänge eingeben und lesen den Vektor Elementweise ein.
- Lassen Sie die Anzahl Zeilen und Spalten eingeben und lesen die Matrix Elementweise ein.
- Fragen Sie z.B.: m(1,2)=?
- Für die Multiplikation sind, wie in Gleichung (9) zu sehen, zwei Schleifen i und k notwendig.
- Die Ausgabe des Ergebnisvektors erfolgt am Einfachsten wieder Elementweise, z.B.: c(1)=5

- Die Matrix M führt eine 90°-Drehung um die Z-Achse aus, d.h. es dreht den Vektor a =X-Vektor in den Y-Vektor.

Aufgabe 6: **Matrix-Multiplikation (Teil 2)**

- Aufgabe 6 ist eine Erweiterung von Aufgabe 5.
- Matrix Eingeben und Ausgeben wie in Aufgabe 5.
- Matrix-mal-Matrix besteht, wie in Gleichung (14) zu sehen, aus drei Schleifen i, j und k.

Aufgabe 7: **lineare Interpolation**

- Einlesen der beiden Stützpunkte (x_0,y_0=f(x_0)) und (x_1,y_1=f(x_1)) und der gesuchten Stelle x.
- Umstellen der Formeln x_1=x_0+h nach h und y_1=y_0+Δ nach Δ
- Einsetzen von h und Δ in Formel (17) und Berechnung und Ausgabe des Ergebnisses.
- Wikipedia-Artikel: https://de.wikipedia.org/w/index.php?title=Lineare_Interpolation

Aufgabe 8: **Nullstellensuche (Regula falsi)**

- Bestimmung der Quadratwurzel.
- Einlesen des Wertes unter der Wurzel. Welche Quadratwurzel soll bestimmt werden?
- Anfangsbedingung $f(x_0)f(x_1) < 0$, d.h. bei $x_0=0$ und $f(x)=x^2-a$ ist $-a*(x_1^2 -a)<0$ also $x_1^2>a$,
  ist a >= 1 gilt dies z.B. für $x_1=a$
  ist a < 1 reicht $x_1=1$
- Durch iteratives anwenden von Formel (19) und (18) wird das Ergebnis immer weiter angenähert. Die Iteration, d.h. die Schleife muss sooft angewendet werden, bis eine ausreichende Genauigkeit erreicht wird.
  Die Schleife kann entweder durch eine DO-Schleife aufgebaut werden, dabei stellt sich aber folgendes Problem: Wie viel Durchläufe benötige ich für eine ausreichende Genauigkeit. Besser ist es eine Schleife mit Hilfe einer Sprungmarke und einer IF-Bedingung zu bilden. Hierbei bestimmt die IF-Bedingung wann eine ausreichende Genauigkeit $x_m-x_{m-1}< \varepsilon$ erreicht ist und die Schleife beendet wird.
- Die Variablen x_m oder s_m werden/müssen nicht durch Felder implementiert werden. Von s_m wird nur der aktuelle Wert benötigt, d.h. $s=s_m$. Von x_m wird in Iteration m nur der aktuelle Wert also x_m und der Vorgängerwert x_{m-1} benötigt, d.h. ich benötige hierfür zwei Variablen $x=x_m$ und $x\_alt=x_{m-1}$. Wobei ich bei jeder Iteration vor Berechnung von x den alten Wert sichern muss x_alt=x
- Wikipedia-Artikel: https://de.wikipedia.org/wiki/Regula_falsi

Aufgabe 9: **Monte Carlo Integration**

- Die Fortran Funktion rand() liefert pseudo-Zufallszahlen im Bereich [0, 1[.
- Zur Funktion f(x,y): Die Punkte (0,0),(x,0) und (x,y) bilden ein Dreieck, dessen Hypotenuse die Distanz d des Punktes (x,y) zum Nullpunkt (0,0) beschreibt. Ist die Distanz d kleiner oder gleich 1 liegt der Punkt (x,y) im Teilkreis. Es reicht also zu prüfen, ob $d^2=x^2+y^2<=1$ ist, dann ist f(x,y)=1, sonst 0.
- Die Anzahl der Iterationen bestimmt die Genauigkeit des Ergebnisses.
- Die Teilfläche ist das Verhältnis der Anzahl der Treffer im Teilkreises mit f(x,y)=1 zur Gesamtanzahl der Versuche, d.h. Iterationen.
- Wikipedia-Artikel: https://de.wikipedia.org/wiki/Monte-Carlo-Simulation