Opgave 4 til uge 4: Funktioner

Opgave 4a

Skriv en funktion, der finder ud af hvor mange bits typen kan indeholde for en variabel af typen **int**.

Du starter med at sætte en variabel **int n** til værdien 1. Programmet kan bruge en løkke til at fordoble (2*n) værdien af **n** indtil du får overflow. Du kan se at der er overflow ved at den nye værdi ikke er større end den gamle. Antallet af bits i **n** kan findes ud fra det antal gange du kan fordoble værdien før du får overflow. Skriv antal bits ud på konsollen med en printf.

Hvis du kører den samme funktion på en PC og en Arduino vil du se at der ikke er det samme antal bits i en **int**. Fortæl hvor mange bits du finder på PC og på Arduino.

Du kan også prøve med andre heltals-typer: **char**, **short int**, og **long int**. (Hvis du har adgang til en computer med MacOS eller Linux vil du kunne se at **long int** ikke har det samme antal bits i 32-bit og 64-bit mode i disse systemer).

Opgave 4b

Skriv en funktion med et retvisende navn, der beregner kvadratroden af et tal y ved brug af den såkaldte Newton-Raphson metode, der er beskrevet her: https://www.quora.com/Using-Newton%E2%80%93Raphson- method-establish-the-formula-X_-n+1-frac-1-2-X_n-+-frac-N-X_n-to-calculate-sqrt-N-Hence-find- sqrt-5-correct-to-four-places-of-decimals

Den benyttede algoritme for kvadratrod er:

$$(1)\ x_{ny} = 0.5 \cdot \left(x_{gammel} + \frac{N}{x_{gammel}} \right)$$

$$(2)\ x_{gammel} = x_{ny}$$

start med et vilkårligt gæt for x_{gammel}, for eksempel 1. Gentag (1) og (2) for eksempel 5 eller 10 gange. Nøjagtigheden bliver forbedret for hver iteration (genemløb). Brug typen float eller double x variablerne.

Lav et program der indlæser N, beregner x = √N ved at kalde din funktion, og udskriver N, x (resultat), og x*x. Hvor godt stemmer N med x*x? - hvis N er negativ skal det håndteres!

Alternativ til opgave 4b: Taylors formel
Denne opgave kan vælges som et alternativ til overstående 4b.

Sinusfunktionen kan beregnes ved hjælp af Taylors formel:
Her for n=0,1,2,3

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots$$

generelt n=0 til G

$$\sin(x) \approx \sum_{n=0}^{G} \frac{(-1)^n x^{2n+1}}{(2n+1)!} = \sum_{n=0}^{G} \frac{A_n}{B_n}$$

Dette kan beregnes effektivt ved at beregne hvert led ud fra det forrige i rækken. For tælleren A_n har vi:

$$A_n = -x^2 A_{n-1}, \quad A_0 = x$$

For nævneren B_n har vi

$$B_n = B_{n-1}(2n)(2n+1), \quad B_0 = 1$$

1. **Lav en funktion** der beregner sin(x) som en Taylorrække ved hjælp af en løkke der summerer alle led op til n = G. **Der lægges vægt på at koden skal være effektiv**. Hint: Så **ikke noget med n! forfra for hvert led**

Rækkeudviklingen konvergerer hurtigst hvis x er tæt på 0. Hvis x er langt fra 0 kan vi bruge forskellige omskrivninger for at få en hurtigere konvergens:

- Hvis x < 0 bruges omskrivningen sin(x) = -sin(-x)
- Hvis x ≥ 2π bruges omskrivningen sin(x) = sin(x -2π)
- Hvis x ≥ π bruges omskrivningen sin(x) = -sin(x - π)
- Hvis $x \geq \frac{\pi}{2}$ bruges omskrivningen sin(x) = sin(π - x)

Brug disse omskrivninger (i den nævnte rækkefølge) til at sikre at $|x| < \frac{\pi}{2}$ før du benytter rækkeudviklingen.

Find en værdi for graden G der giver en passende

nøjagtighed.

Option udvidelse til 4b vi ønsker nu at beregne

integralet

$$\int_0^{\pi} \sin(x)\, dx$$

Vi approksimerer integralet med en sum

$$\int_0^{\pi} \sin(x)\, dx \approx \sum_{i=0}^{\frac{\pi}{\Delta x}} \sin(i\Delta x)\, \Delta x$$

Beregn dette integral ved hjælp af en løkke der kalder den sinusfunktion, du har lavet. Find en værdi af skridtlængden Δx der giver en passende nøjagtighed.

Udskriv den beregnede værdi af integralet og sammenlign resultatet med den teoretiske værdi.

Programmeringsprocessen

1. læs opgaven – er det til at forstå? – stil spørgsmål til hinanden, til underviseren til hjælpelæreren.

2. Diskuter med din gruppe-makker hvad det går ud på – Hvilke delproblemer kan opgaven brydes ned i?
Prøv hypoteser af for hvad der skal ske under afviklingen – skriv et flow chart over afvikling for delproblem(er), der vise sekvensen af ting der skal ske og betingelser– hvis det er noget er kan regnes på så sæt tal på regn ud hvad der sker delvist og samlet – skitser på papir – tavle –
Hvor mange variable, hvilke type(r).

3. Forestil dig et gennemløb af programmet eller del af det og lad som om der kommer noget ind og ud – regn på det som du mener det skal være – dvs. hypotese test. Begynd så at programmere delproblemet

4. Hvilke C konstruktioner kan løse del-problemet, er der behov for input fra brugeren, Hvad skal brugeren give af input, tal bogstav sætning af tekst. Skal der skrives noget ud til brugeren, hvad skal skrives ud, hvordan skal det skrives ud – formatering, er der betingede statements, if, if else, Swirch case, with select, while/for loop etc. Hvilke typer for variable er nødvendige - skriv alt det ned – diskuter med din makker.

5. Er der tidligere opgaver eller problemløsninger der kan benyttes – bygges videre på? Er der eksempler i bogen der kan bygges på eller direkte benyttes? Afgør hvad vil kunne benyttes til at løs del-problemet/problemet.

6. Åben Visual studio og vælg consol c++, giv et sigende navn til programmet vælg en projektfolder. Begynd at programmere løsningen – skriv en line afslut med; og benyt build dvs. compiler - så ved du hvor fejlen er hvis det ikke compilerer. Skriv kun så få linjér kode før du begynder at debugge – dvs. lav hele tiden små del-test! Sammenhold del test med dine hypoteser i 2. Passer det eller skal noget justeres i hypotesen eller i programmet læs igen opgave teksten/spørgsmålet – diskuter det med din makker! Programmet bygges typisk ved flere gennemløb af 4 til 6 og besøg af 1 og 2 ind og imellem – er du på rette vej?

7. Når alt compilerer – så lav en test – lad din makker test – ved at han/hun giver input.