**1. Vysvětlete pojem tabulkový procesor. Popište, z jakých částí se skládá sešit, co vše můžeme zapsat do buňky. Ukažte efektivní práci s tabulkovým procesorem.**

**Co je tabulkový procesor?**

Tabulkový procesor je program určený pro práci s daty v tabulkové formě. Umožňuje organizovat informace, provádět výpočty a analyzovat data. Nejznámější tabulkové procesory:

* **Microsoft Excel –** nejrozšířenější, součást balíku Microsoft Office.
* **Google Sheets –** online verze, umožňuje sdílení a spolupráci.
* **LibreOffice Calc –** open-source alternativa.

**Struktura sešitu v tabulkovém procesoru**

**Sešit obsahuje několik listů (podobně jako více stránek v dokumentu). Každý list je složen z:**

* Sloupců (A, B, C…)
* Řádků (1, 2, 3…)
* Buňek (např. A1, B3 – kombinace sloupce a řádku)

**Každá buňka může obsahovat čísla, text, vzorce, datum, čas nebo grafy.**

**Co můžeme zapsat do buňky?**

✅ Čísla – pro výpočty (např. 100, 3.14).  
✅ Text – popisky sloupců nebo poznámky.  
✅ Vzorce – například =A1+B1 (sčítání dvou buněk).  
✅ Funkce – předdefinované výpočty, například:

* =SUM(A1:A10) – součet hodnot.
* =AVERAGE(A1:A10) – průměr hodnot. **✅** Datum a čas – různé formáty (12.02.2024 nebo 12:45).

**Efektivní práce s tabulkovým procesorem**

🔹 Použití vzorců a funkcí – usnadňuje výpočty.  
🔹 Podmíněné formátování – umožňuje vizuální zvýraznění dat.  
🔹 Filtrování a řazení – pro lepší orientaci v datech.  
🔹 Tvorba grafů – například sloupcový, koláčový nebo spojnicový graf.  
🔹 Automatické vyplňování – například pro rychlé očíslování buněk.

**Praktický příklad**

**Výpočet celkových nákladů v Excelu:**

1. Do sloupce A zadáme položky (např. „Telefon“, „Internet“, „Nájem“).
2. Do sloupce B zadáme částky (např. 500, 300, 10000).
3. Do B4 napíšeme vzorec =SUM(B1:B3), čímž získáme součet.

Tabulkové procesory šetří čas a jsou klíčové pro správu financí, analýzu dat i plánování projektů.

**2. Rozdělení programovacích jazyků na kompilované a interpretované vychází z toho, jakým způsobem se zpracovává a vykonává zdrojový kód programu.**

### **Kompilované jazyky:**

Kompilované jazyky jsou ty, jejichž zdrojový kód je nejdříve přeložen pomocí **kompilátoru** do strojového kódu, tedy do formy, které rozumí konkrétní hardware, na kterém má být program spuštěn. Tento strojový kód je poté vykonán přímo procesorem bez nutnosti dalšího překladu během běhu programu. Proces kompilace probíhá ještě před samotným spuštěním programu.

**Kompilátor** je tedy program, který překládá celý zdrojový kód programu do spustitelného formátu najednou, což má určité výhody i nevýhody. Výhodou je, že výsledný program běží rychleji, protože již není potřeba žádné další překládání za běhu. Nevýhodou může být delší doba vývoje, protože každý upravený kód je nutné znovu přeložit před dalším spuštěním.

**Příkladem kompilovaného jazyka** je například **C** nebo **C++**, a příkladem kompilátoru pro tento jazyk je třeba **GCC** (GNU Compiler Collection). V tomto případě se zdrojový kód programu přeloží do binárního spustitelného souboru, který je možné spustit na cílovém systému.

### **Interpretované jazyky:**

Na druhé straně interpretované jazyky jsou ty, jejichž zdrojový kód je překládán a vykonáván postupně po jednotlivých částech za běhu programu. Tento proces zajišťuje **interpret**, což je program, který provádí příkazy přímo ze zdrojového kódu, bez nutnosti překladu celého kódu dopředu.

**Interpret** tedy zpracovává a vykonává kód přímo při spuštění programu, což znamená, že každá změna ve zdrojovém kódu je okamžitě viditelná při dalším spuštění. Tento přístup je výhodný při ladění, protože není nutné program znovu překládat. Nevýhodou je ale to, že vykonání programu může být pomalejší, protože kód je překládán a vykonáván za běhu.

**Příkladem interpretovaného jazyka** je **Python** a příkladem jeho interpretu je například **CPython**. V tomto případě interpret čte a vykonává Python kód přímo, takže při každém spuštění probíhá překlad kódu znovu.

Zatímco kompilované jazyky obvykle vedou k rychlejšímu běhu výsledného programu, interpretované jazyky poskytují větší flexibilitu během vývoje a ladění. Oba přístupy mají své výhody a nevýhody, a volba mezi nimi závisí na konkrétním použití a požadavcích na vývoj programu.

**3. Popište, jaký je rozdíl mezi rastrovým a vektorovým formátem. Popište základní rozdíl v definici a tím i výhody/nevýhody obou typů 2D grafiky. Vysvětlete formáty GIF, JPG, JPEG, BMP, PNG, PDF. Objasněte pojmy pixel, DPI.**

**Rozdíl mezi rastrovou a vektorovou grafikou:**

Obsah obrázku text, snímek obrazovky, Písmo, číslo

Popis byl vytvořen automaticky

**Praktické využití**

✅ **Rastrové obrázky** – fotky v mobilu, internetové obrázky, malby.  
✅ **Vektorové obrázky** – firemní loga, ikony, technické výkresy.

**Formáty souborů**

**Rastrové formáty:**

* **JPG/JPEG** – nejběžnější formát, využívá **ztrátovou** kompresi.
* **PNG** – bezztrátový formát, podporuje průhlednost.
* **GIF** – animace, omezená barevná paleta.
* **BMP** – starší formát, velmi velké soubory.

**Vektorové formáty:**

* **SVG** – vhodný pro web, lze upravovat bez ztráty kvality.
* **PDF** – univerzální formát, obsahuje jak rastrové, tak vektorové prvky.

**Důležité pojmy**

* **Pixel** – nejmenší jednotka rastrového obrazu.
* **DPI (dots per inch)** – hustota pixelů na palec, čím vyšší, tím lepší kvalita.
* **Bezeztrátová vs. ztrátová komprese**
* **Bezeztrátová (PNG, BMP)** – žádná ztráta kvality, větší soubory.
* **Ztrátová (JPG, GIF)** – menší velikost, ale nižší kvalita obrazu.

**Praktické příklady**

* **Fotky na sociálních sítích jsou ve formátu JPG**, protože je potřeba malá velikost.
* **Loga a ilustrace jsou v SVG/PDF**, protože je lze zvětšit bez ztráty kvality.