SPU·FEM
Fakulta
ekonomiky
a manažmentu

Softvérový vývoj

RNDr. Tomáš Tóth, PhD.

Ústav účtovníctva a informatiky
FEM SPU v Nitre

ttoth@uniag.sk

Čo sa dnes dozvieme

- Čo je to softvér a jeho delenie
- Čím sa zaoberá softvérové inžinierstvo
- Aké sú fázy softvérového vývoja
- Kto tvorí tím softvérových vývojárov
- Čo je to programovanie a jeho význam
- Trendy a budúcnosť softvérového vývoja

Výsledkom programovania
a softvérového vývoja je

SOFTVÉR

Softvér

- Základ technologických riešení
- Súčasť smartfónov, počítačov, systémov riadiacich dopravu, zdravotníctvo, finančné služby
- Pomáha riešiť problémy, automatizovať úlohy, komunikovať, zabávať sa a učiť sa

Softvér

- **Súbor inštrukcií a údajov**, ktoré umožňujú počítačom a iným zariadeniam **vykonávať špecifické úlohy**
  - **Nehmotný** a slúži na **riadenie a ovládanie hardvéru**
    - **Nevyhnutný pre fungovanie** počítača a všetkých digitálnych zariadení

Delenie softvéru

Systémový softvér

Aplikačný softvér

Delenie softvéru

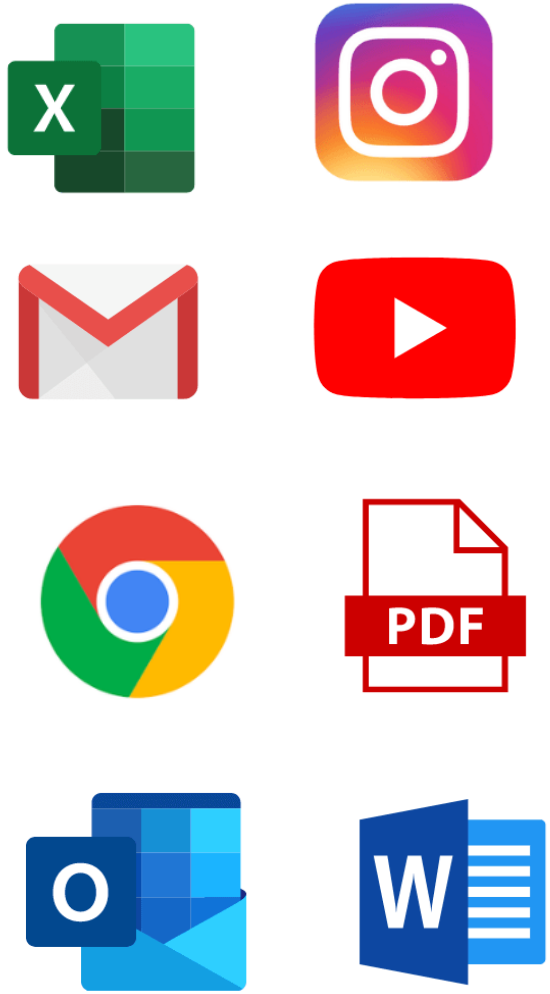
1. Systémový softvér

- Zahŕňa **operačné systémy a základné programy** umožňujúce riadenie PC
- Funkcia:
  - riadi zariadenia
  - spravuje pamäť
  - zabezpečuje interakciu medzi hardvérom a aplikačným softvérom

Delenie softvéru

2. Aplikačný softvér

- Programy poskytujúce **konkrétne nástroje a služby** pre používateľov
- Napr. webový prehliadač, textový editor, mobilné aplikácie,...
- Umožňuje používateľom vykonávať rôzne úlohy, ako **napríklad tvorbu dokumentov, správu údajov alebo prístup na internet**



Pokrok softvéru v priebehu doby

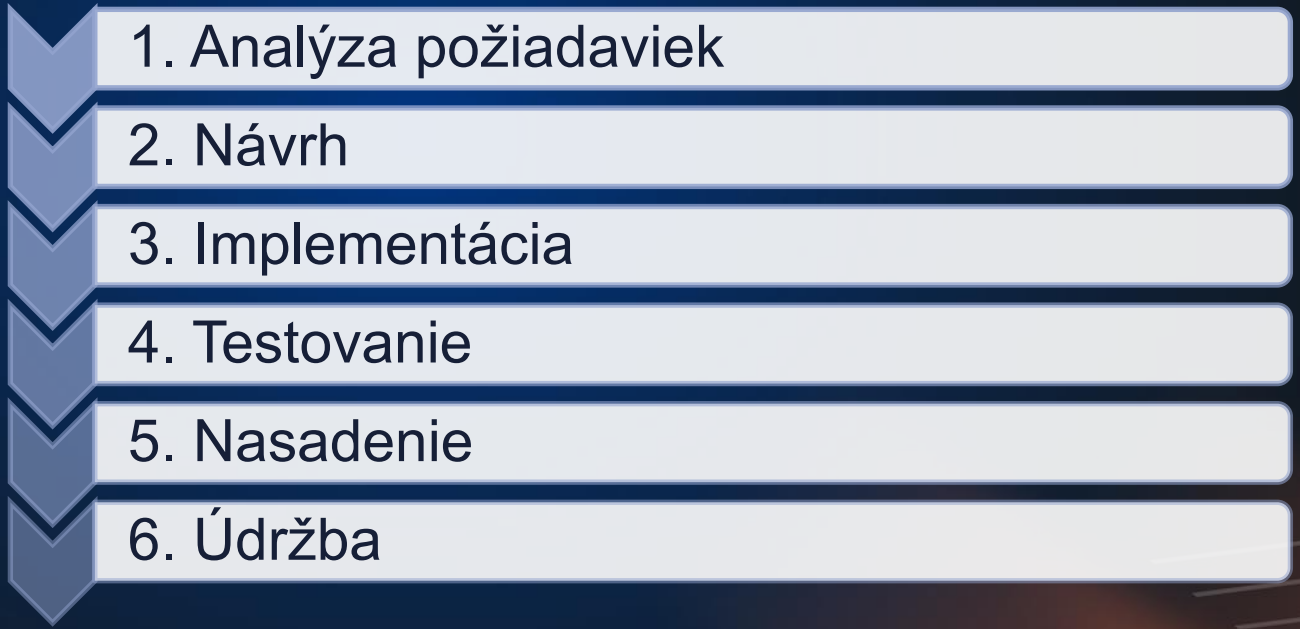
- Softvér sa neustále **vyvíja**
- Softvér sa **prispôsobuje** používateľom a novým technológiám
- Jeho neustály vývoj umožňuje **inovácie a pokrok v technológiách**

Softvérové inžinierstvo

- Disciplína, ktorá sa zameriava na **aplikáciu inžinierskych princípov a metodológií** na vývoj, nasadenie a údržbu softvéru
  - Systematický, metodický a efektívny **prístupom k vývoju softvéru**
    - Zahŕňa **celý životný cyklus softvéru**, od analýzy a špecifikácie požiadaviek až po nasadenie a dlhodobú údržbu
      - **Cieľom je vytvárať** spoľahlivý, efektívny a udržiavateľný **softvér**, ktorý spĺňa požiadavky používateľov a zákazníkov

Softvérový vývoj

- Proces vytvárania softvéru
- Hlavné fázy:



1. fáza vývoja softvéru: **analýza požiadaviek**

- **Cieľ:** Pochopiť **očakávania** klienta alebo používateľa od vyvíjaného softvéru a presne **definovať jeho funkcionalitu, vlastnosti a obmedzenia**

- Kvalitná analýza požiadaviek minimalizuje riziko nedorozumení a chýb v neskorších fázach vývoja

1. fáza vývoja softvéru: **analýza požiadaviek**

Funkčné požiadavky

- Určujú, **čo má softvér robiť**
- Napr. softvér umožní používateľovi sa zaregistrovať zadaním e-mailovej adresy a hesla, manažér si môže pozrieť vystavené faktúry, atď.)

Nefunkčné požiadavky

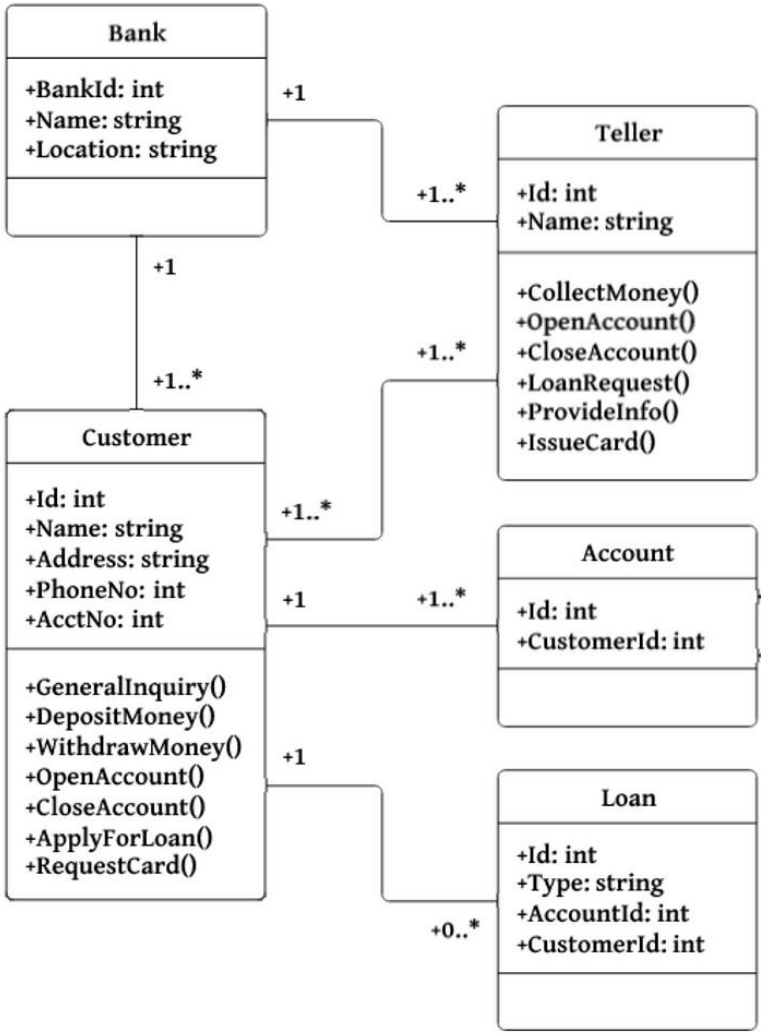
- Týkajú sa **kvalitatívnych aspektov softvéru**, ako sú výkon, bezpečnosť a použiteľnosť
- Napr. overenie karty sa zrealizuje do dvoch sekúnd, softvér má byť škálovateľný, kompatibilný s inými systémami, atď.

2. fáza vývoja softvéru: **návrh**

- **Cieľ: Transformovať požiadavky** do popisu technického riešenia a špecifikácií pre vyvíjaný softvér
- Vytvára sa **plán a architektúra softvérového riešenia** tak, aby bol systém efektívny, udržateľný a ľahko rozšíriteľný
- Rozhoduje sa o **technológiách, programovacom jazyku** a celkovej **architektúre systému**

2. fáza vývoja softvéru: **návrh**

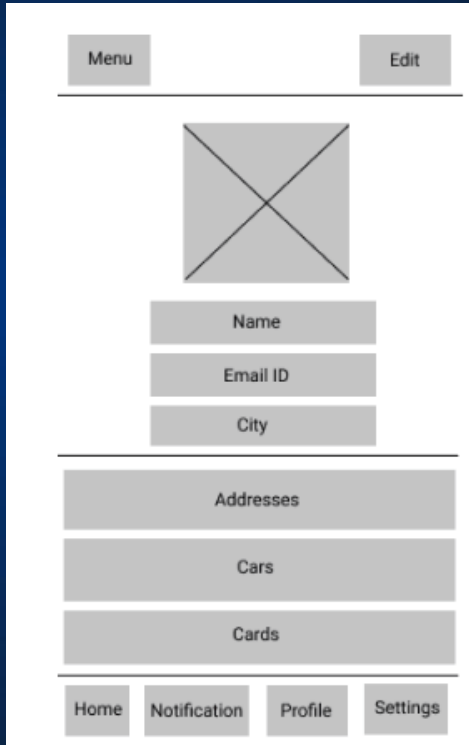
- **UML diagramy**
  (z angl. Unified Modeling Language)
  - Pre lepšiu **vizualizáciu, špecifikáciu a pochopenie** návrhu softvéru
  - Zobrazujú **štruktúru a funkcie** softvéru



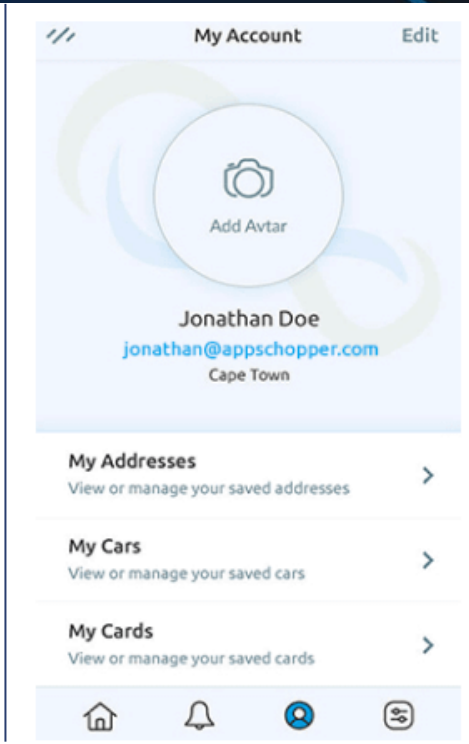
2. fáza vývoja softvéru: **návrh**

- Návrh **používateľského rozhrania**
  (z angl. user interface, UI)

- Vytvárajú sa **náhľady softvéru – wireframe alebo mockup**
  (návrhy rozloženia prvkov na obrazovke)



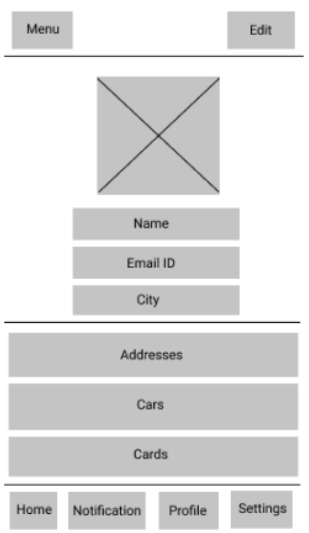
wireframe



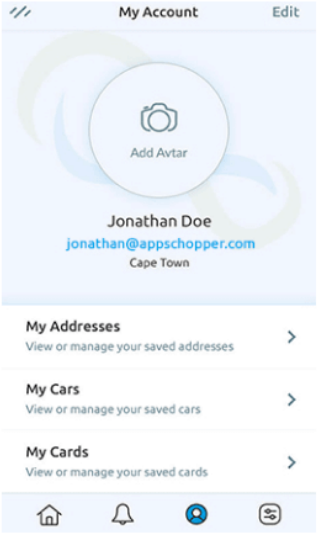
mockup

2. fáza vývoja softvéru: **návrh**

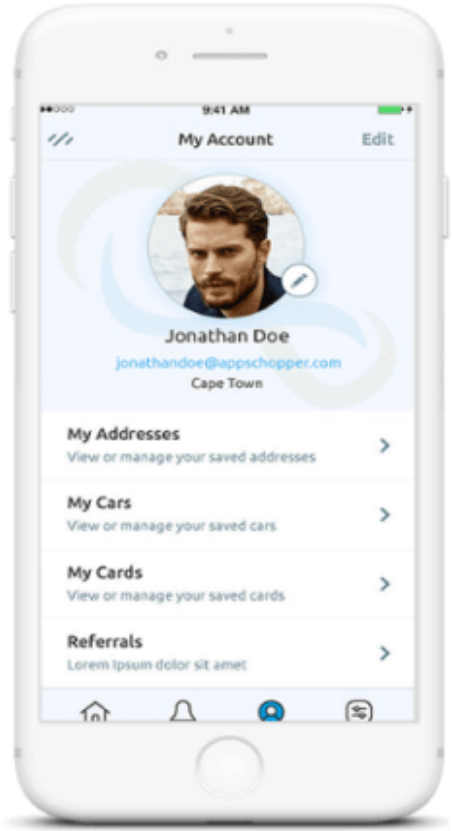
- **Výsledok:** dokumentácia alebo aj prototyp (slúži ako interaktívna ukážka budúceho produktu)



wireframe



mockup



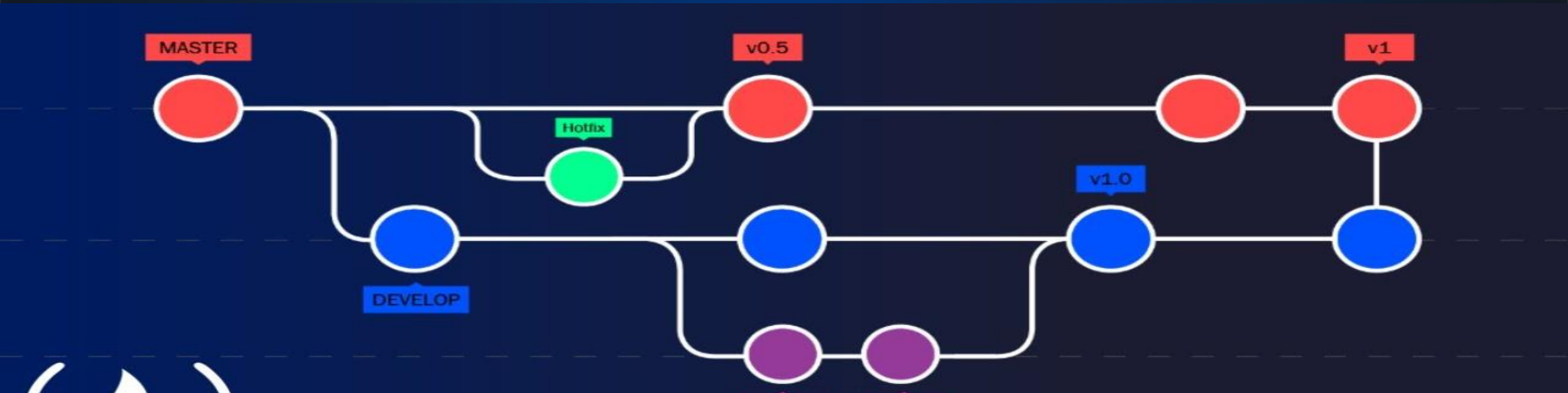
prototyp

3. fáza vývoja softvéru: **implementácia**

- Proces prekladu návrhu softvéru do funkčného a strojom (počítačom, smartfónom,...) vykonateľného kódu
- Využívajú sa programovacie jazyky, vývojové prostredia, frameworky a knižnice na realizáciu definovaných funkcionalít systému
- Kvalitná implementácia si vyžaduje dodržiavanie kódovacích štandardov

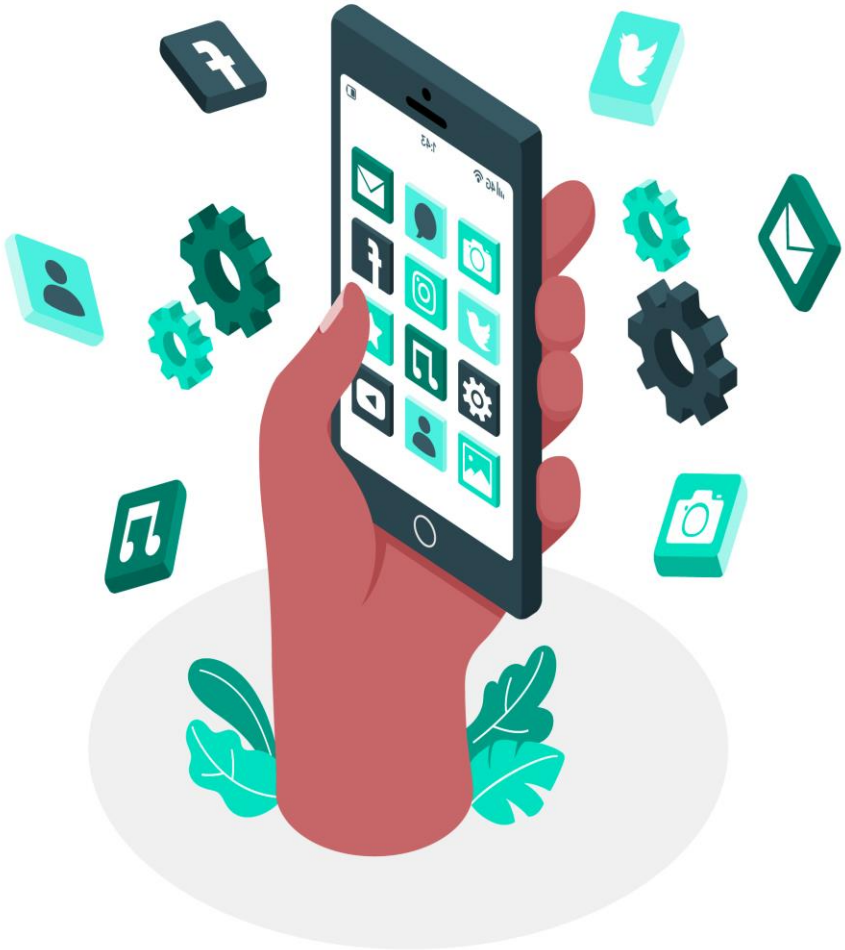
3. fáza vývoja softvéru: **implementácia**

- **Verzionovanie kódu**
  - umožňuje efektívnu správu zmien v kóde prostredníctvom systémov na riadenie verzií
  - užitočné pre tímovú spoluprácu



3. fáza vývoja softvéru: **implementácia**

- **Výsledok:**
  funkčný softvérový produkt



4. fáza vývoja softvéru: **testovanie**

- Kľúčovým krokom v procese **zabezpečenia kvality a spoľahlivosti** vyvíjaného softvéru
- **Identifikácia chýb a nedostatkov** v kóde, funkčnosti alebo správaní systému
- Sledovanie, či aplikácia **spĺňa definované požiadavky a štandardy**
- Podporované nástrojmi na automatizáciu
- **Rôzne typy:** funkčnosti, bezpečnosti, výkonnosti
- Nie je jednorazový proces, ale **prebieha počas celého životného cyklu** softvéru

5. fáza vývoja softvéru: **nasadenie**

- **Premiestnenie** vyvíjaného softvéru z vývojového alebo testovacieho prostredia **do produkcie** – **sprístupnenie koncovým používateľom**
- **Cieľ:** Zabezpečiť, aby softvér fungoval spoľahlivo a efektívne v reálnom prostredí a spĺňal požiadavky používateľov
- **Zahŕňa:**
  - spustenie softvéru
  - konfigurácia systému
  - nastavenie databáz
  - implementácia bezpečnostných opatrení
  - integrácia s inými systémami
  - školenie používateľov

6. fáza vývoja softvéru: **údržba**

- Nevyhnutné **monitorovať** jeho výkon a stabilitu, **vykonávať** pravidelnú údržbu
- Dlhodobý proces
- **Aktivity** potrebné na zachovanie, optimalizáciu a vylepšenie systému počas jeho životného cyklu
  - Opravy chýb
  - Aktualizácie softvéru – zohľadnenie zmien v požiadavkách používateľov alebo technologických podmienkach, implementáciu nových funkcií
  - Zlepšovanie výkonu
  - Zlepšenie bezpečnosti systému
- Spravujú aj otázky kompatibility so staršími verziami systému, keďže nové aktualizácie môžu ovplyvniť existujúce funkcionality

Tím softvérových vývojárov a ich role

- **Vedúci tímu** (z angl. team leader) – Vedenie a koordinácia práce tímu, plánovať a organizovať prácu, motivovať členov tímu
- **Analytik** (z angl. analyst) – Úlohou je komunikovať s používateľmi, zhromažďovať a analyzovať požiadavky a navrhnúť riešenie
- **Softvérový architekt** (z angl. software architect) – Úlohou je navrhnúť a definovať technickú štruktúru a architektúru softvéru
- **Vývojár** (z angl. developer) – Je zodpovedný za písanie kódu a implementáciu funkcionality softvéru. V tíme môžu byť rôzni vývojári so špecializáciou na rôzne oblasti (frontend vývojári, backend vývojári alebo databázoví špecialisti)
- **Tester** (z angl. tester) – Testovanie softvéru a identifikáciu chýb. Jeho úlohou je realizovať rôzne typy testov a zdokumentovať výsledky

Tím softvérových vývojárov a ich role

- V závislosti od veľkosti projektu sa v tíme môžu vyskytovať aj rôzne **ďalšie role:**
  - projektový manažér
  - produktový manažér
  - UX/UI špecialista
  - Grafik
  - scrum master
- Pri menších projektoch sa môžu niektoré **role zlučovať** a viacero rolí môže vykonávať jedna osoba

Algoritmizácia

vs

Programovanie

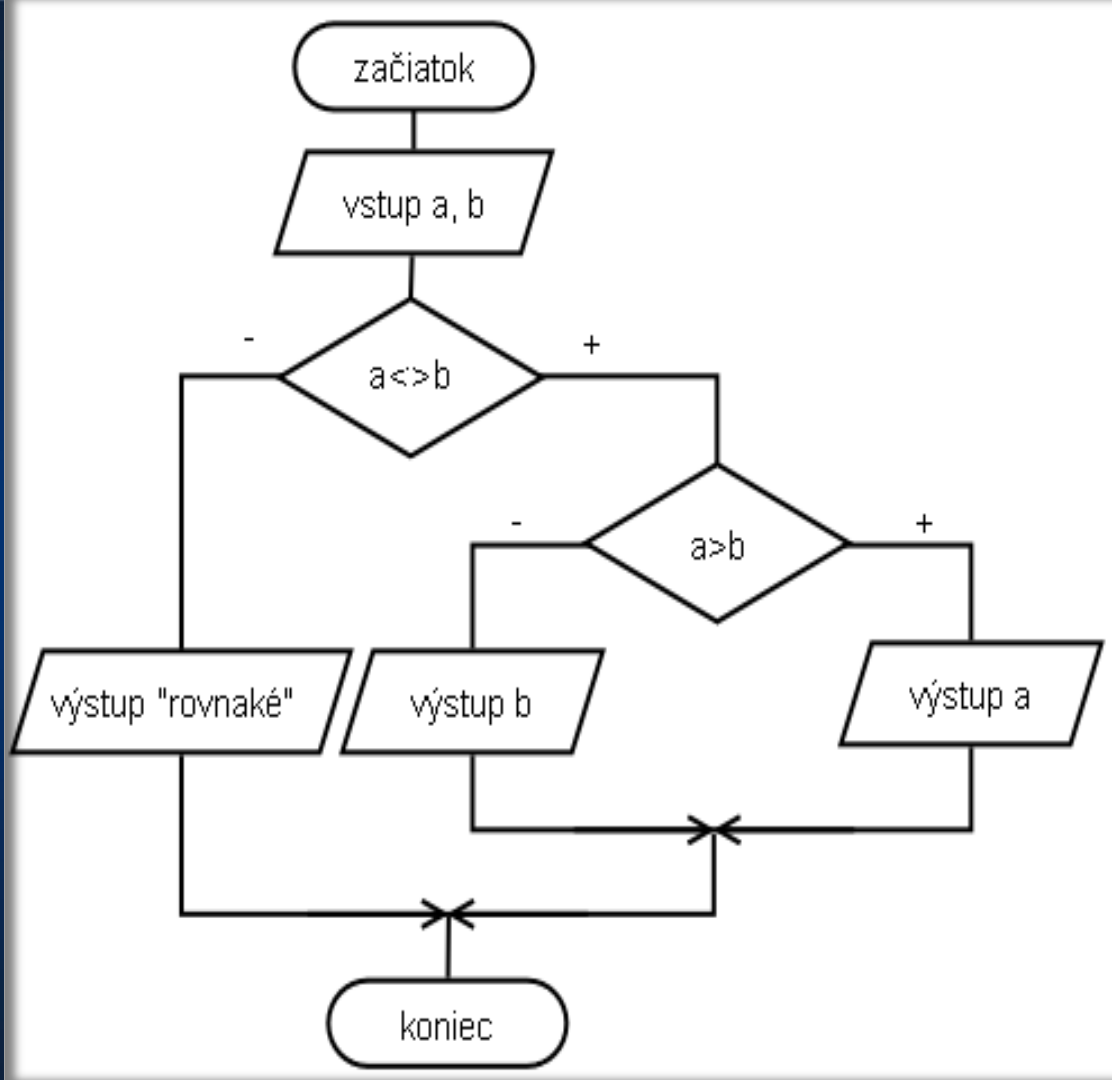
Definícia algoritmu

Algoritmus je návod
(presná postupnosť krokov a inštrukcií)
na vykonanie činnosti, ktorý nás
od (meniteľných) vstupných údajov
privedie v konečnom čase
k výsledku

Proces, ktorý vykonávame pri zápise algoritmu sa nazýva ***algoritmizácia***

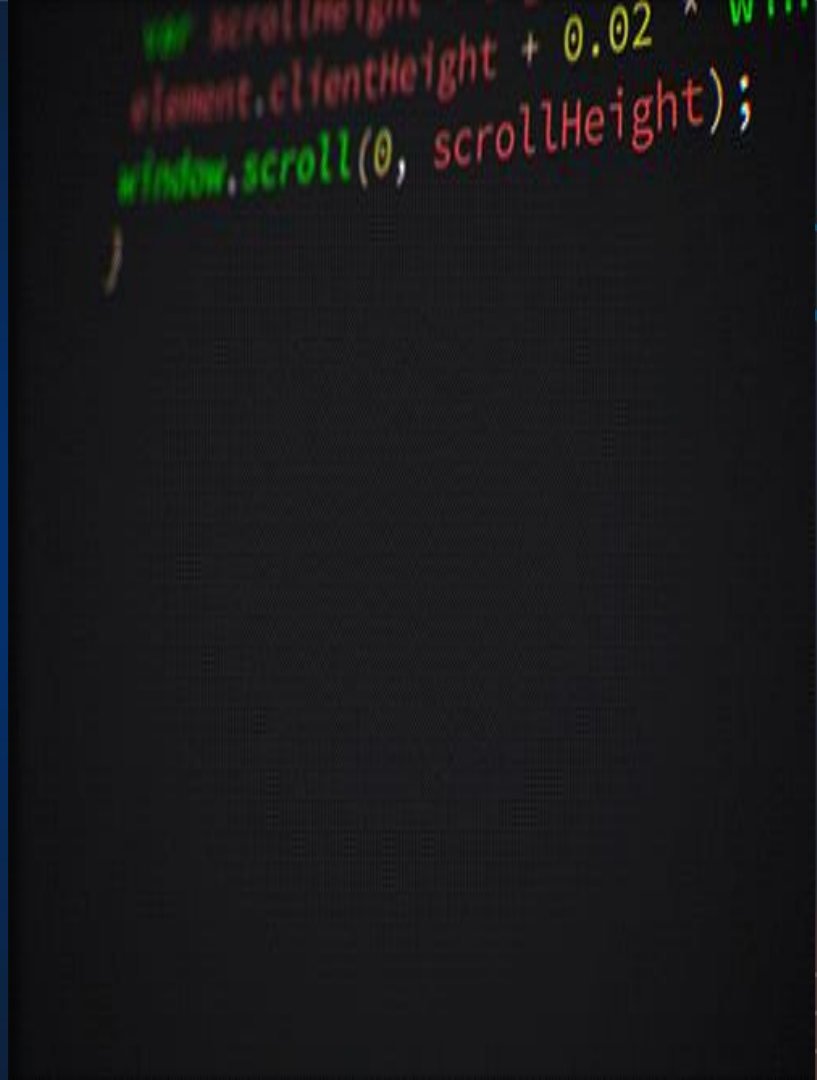
Algoritmický jazyk

- Rôzne typy
- Napr. vývojé diagramy a ďalšie



Programovanie

- Proces **tvorby a implementácie algoritmov** (algoritmizácie) prostredníctvom **programovacích jazykov**
- Proces **vytvárania inštrukcií** pre počítač **s cieľom riadiť** jeho **činnosť** (a aj iných zariadení)



Programovanie

- V dnešnom svete mimoriadne **dôležité**
- Je **základom pre fungovanie** mnohých **technológií**, ktoré používame každý deň
- Používa sa na **vytváranie rôznych typov softvérových riešení**
  - od jednoduchých mobilných hier
  - cez aplikácie, ktoré zjednodušujú a zefektívňujú každodenné činnosti
  - až po komplexné systémy, ktoré riadia chod spoločností
- Prostredníctvom programovania je možné vytvárať **aplikácie, ktoré slúžia na**:
  - spracovanie údajov, komunikáciu, zábavu, vzdelávanie či riadenie podnikových operácií
- Umožňuje **automatizovať rôzne procesy** – to zvyšuje efektivitu a produktivitu práce

Ada Lovelace

- Nar. 10.12.1815
- Grófka z Lovelace
- Britská matematička
- Napísala prvý program pre analytický stroj Charlesa Babbagea

Programovací jazyk

- Patrí medzi algoritmické jazyky
- **Formalizuje algoritmický jazyk** do zápisu, **ktorý dokáže spracovať a zrealizovať počítač** (alebo iné zariadenie)
- Rôzne typy pre rôzne úlohy
- Často založené na redukcii slov anglického jazyka
- Umožňuje programátorom komunikovať s počítačom spôsobom, ktorému počítač rozumie

Syntax a sémantika

- Každý programovací jazyk má svoju syntax a sémantiku
- **Syntax**
  - definuje **pravidlá pre zápis** kódu
- **Sémantika**
  - určuje **význam** jednotlivých konštrukcií v jazyku
  - definuje, aký účinok majú jednotlivé inštrukcie na stav programu

- Príklad:

Budem vynikajúci/a v programovaní.

Syntax: OK
Sémantika: OK

Programovanie bude vynikajúco vo mne.

Syntax: OK
Sémantika: Nesprávne

Programovacie jazyky - delenie

Nízkoúrovňové

- Blízke hardvéru počítača a strojovému kódu
- Príkladom je Assembly

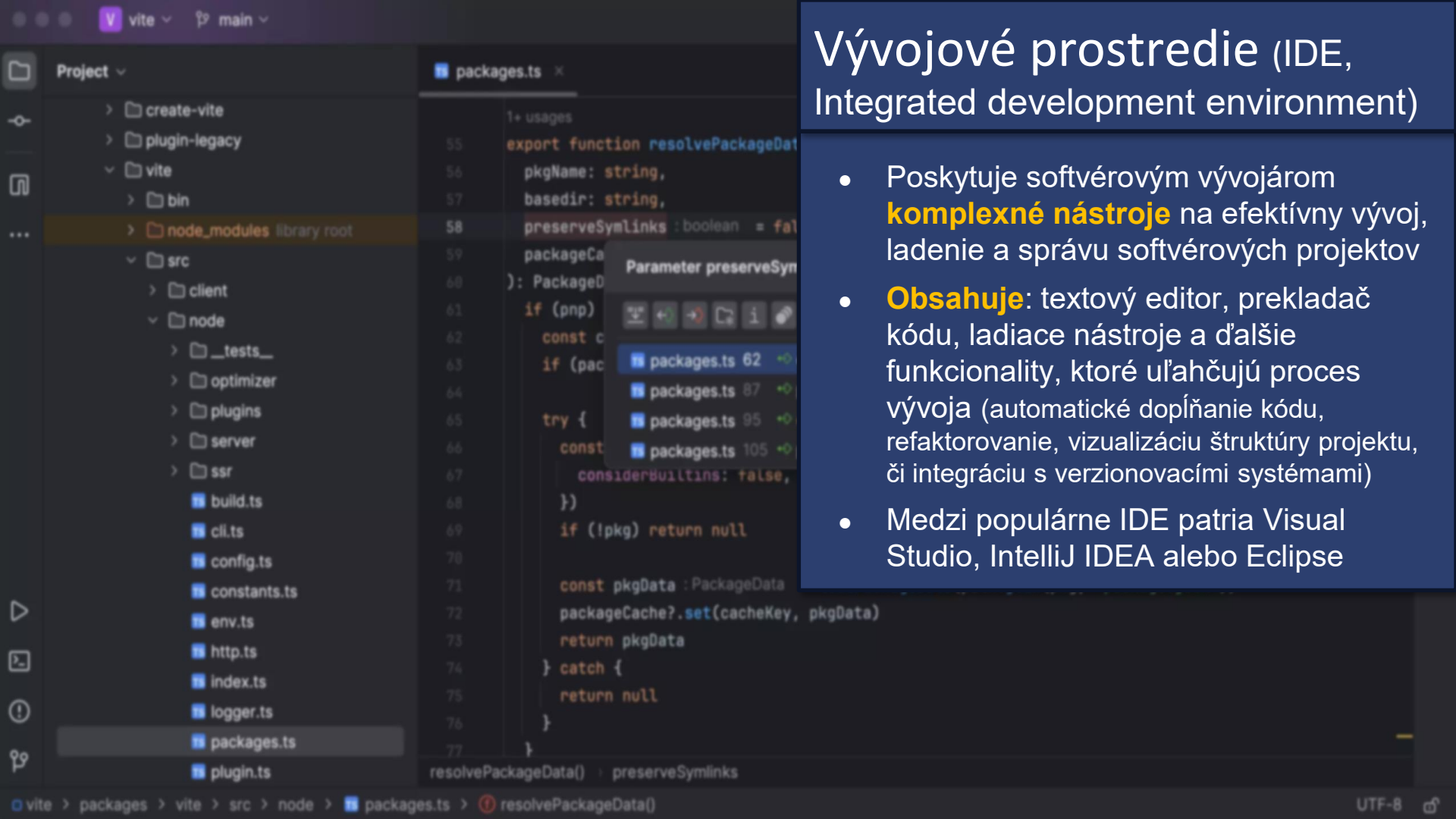
Vysokoúrovňové

- Viac abstraktné a umožňujú programátorom pracovať s dátami a operáciami na vyššej úrovni
- Príkladmi sú C, C++, Java, Python


Z histórie programovania - dierne štítky
Programovanie vytváraním rôznych kombinácií dier na štítku

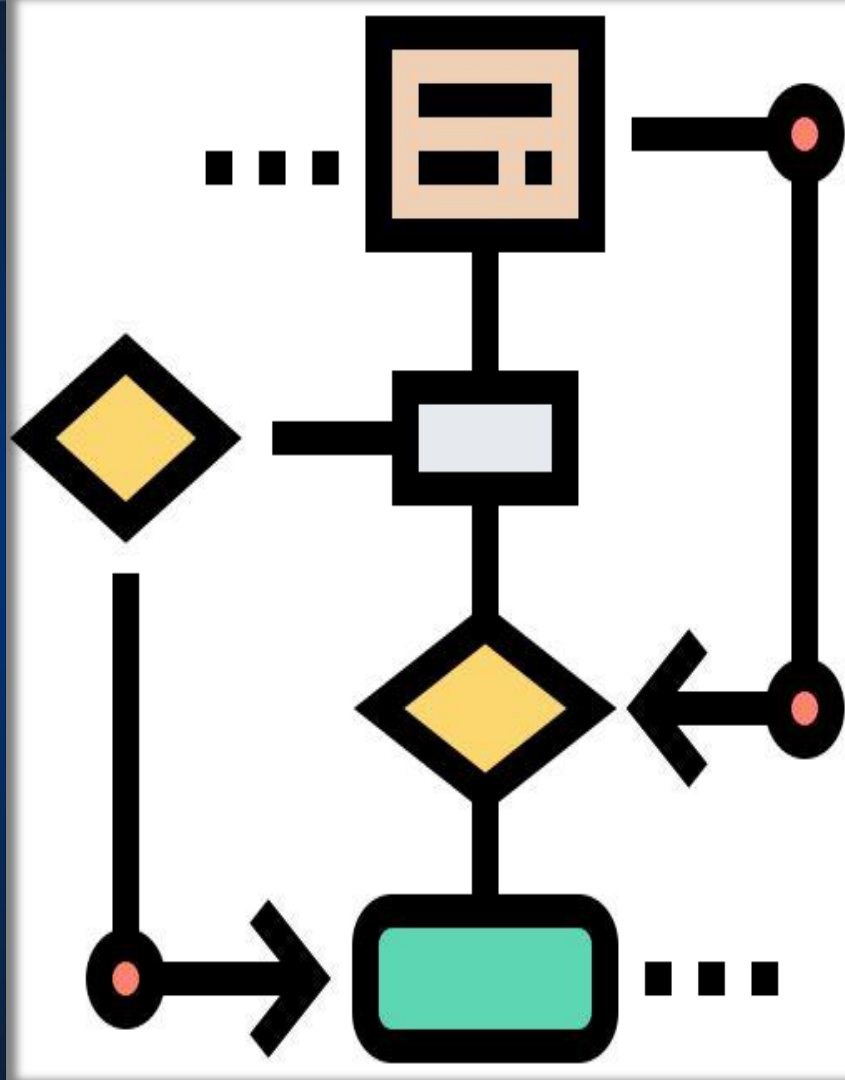
Vývojové prostredie (IDE, Integrated development environment)

- Poskytuje softvérovým vývojárom **komplexné nástroje** na efektívny vývoj, ladenie a správu softvérových projektov
- **Obsahuje**: textový editor, prekladač kódu, ladiace nástroje a ďalšie funkcionality, ktoré uľahčujú proces vývoja (automatické dopĺňanie kódu, refaktorovanie, vizualizáciu štruktúry projektu, či integráciu s verzionovacími systémami)
- Medzi populárne IDE patria Visual Studio, IntelliJ IDEA alebo Eclipse



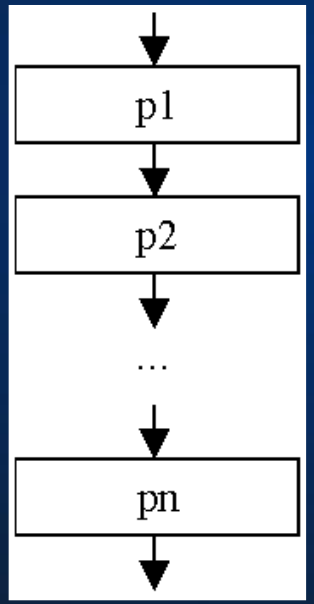
Riadiace štruktúry

- **Riadia sled vykonávania programu**
- Patrí sem:
  - Sekvencia
  - Vetvenie
  - Cyklus



Sekvencia

- Postupnosť príkazov vykonávaná **v** takom **poradí, v akom** sú jednotlivé časti **zapísané**

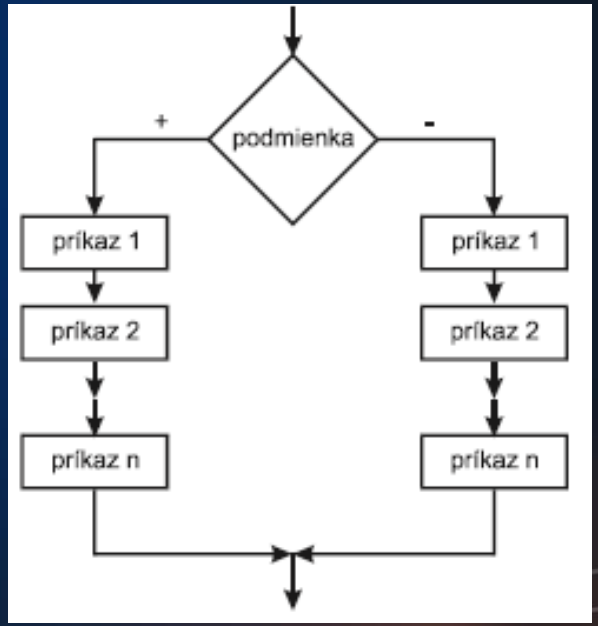


Zápis sekvencie vo vývojovom diagrame

p1, ..., *pn* sú príkazy
vykonajú sa v poradí, v akom sú zapísané

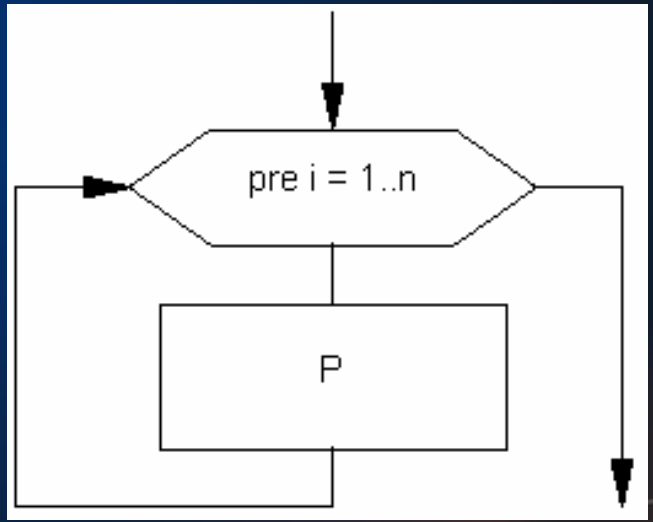
Vetvenie

- Umožňuje vykonanie rôznych častí kódu na základe splnenia určitej logickej podmienky
- Podmienka je tvorená slovíčkom *„ak“*
- Táto podmienka rozdeľuje tok programu štandardne na dve časti - **vetvy**
- Ak je podmienka splnená –> vetva „+“
- Ak nie je podmienka splnená –> vetva „-“



Cyklus

- Zabezpečuje **opakované vykonávanie príkazov**, až kým nie je splnená podmienka
- Opakovaním určitej časti kódu cykly znižujú redundantnosť a zvyšujú efektivitu programov
- Existuje viacero typov cyklov, pričom ich hlavné delenie je podľa toho, či pred začatím vykonávania cyklu poznáme, resp. nepoznáme presný počet opakovaní cyklu.



Prečo sa učiť programovanie?

Prečo sa učiť programovanie

Aby sa človek stal programátorom a dokázal **tvoriť softvér** (desktopové aplikácie, mobilné aplikácie, webové aplikácie) a **automatizovať** opakujúce sa úlohy a procesy – **zvýšenie efektivity a produktivity** v rôznych odvetviach

Prečo sa učiť programovanie

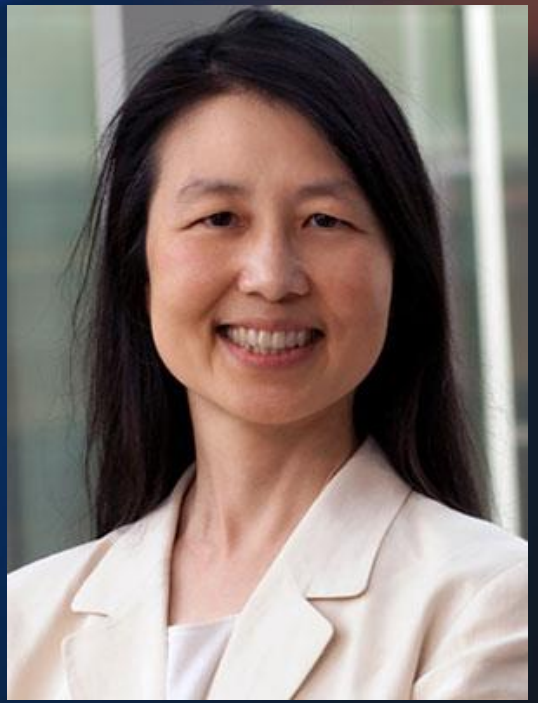
Umožňuje **zbierať, spracovávať a analyzovať veľké množstvá dát** – dôležité pre vedecký výskum, podnikanie a rozhodovanie v rôznych oblastiach

Je pre mňa potrebné programovanie aj keď nechcem byť programátorom?

Prečo sa učiť programovanie – nie len pre programátorov

- Zlepšenie digitálnych zručností
- Lepšie porozumenie svetu technológií a fungovaniu technických zariadení
- Rozvoj zručností dôležitých v 21. storočí:
  - Zručnosť riešenia problémov
  - Informatické myslenie
  - Kritické myslenie
  - Algoritmické myslenie
  - Kreatívne myslenie
  - Analytické myslenie
  - Logické myslenie
  - Abstraktné myslenie
  - Deduktívne myslenie
  - Schopnosť spolupracovať

Schopnosť “myslieť ako informatik” by mala patriť medzi základné zručnosti, akými sú čítanie, písanie a počítanie.



Jeannette M. Wing

- Profesorka informatiky na Columbia University
- Zastávala a aj stále zastáva rôzne vedúce pozície v akademickej oblasti a aj v komerčnej sfére

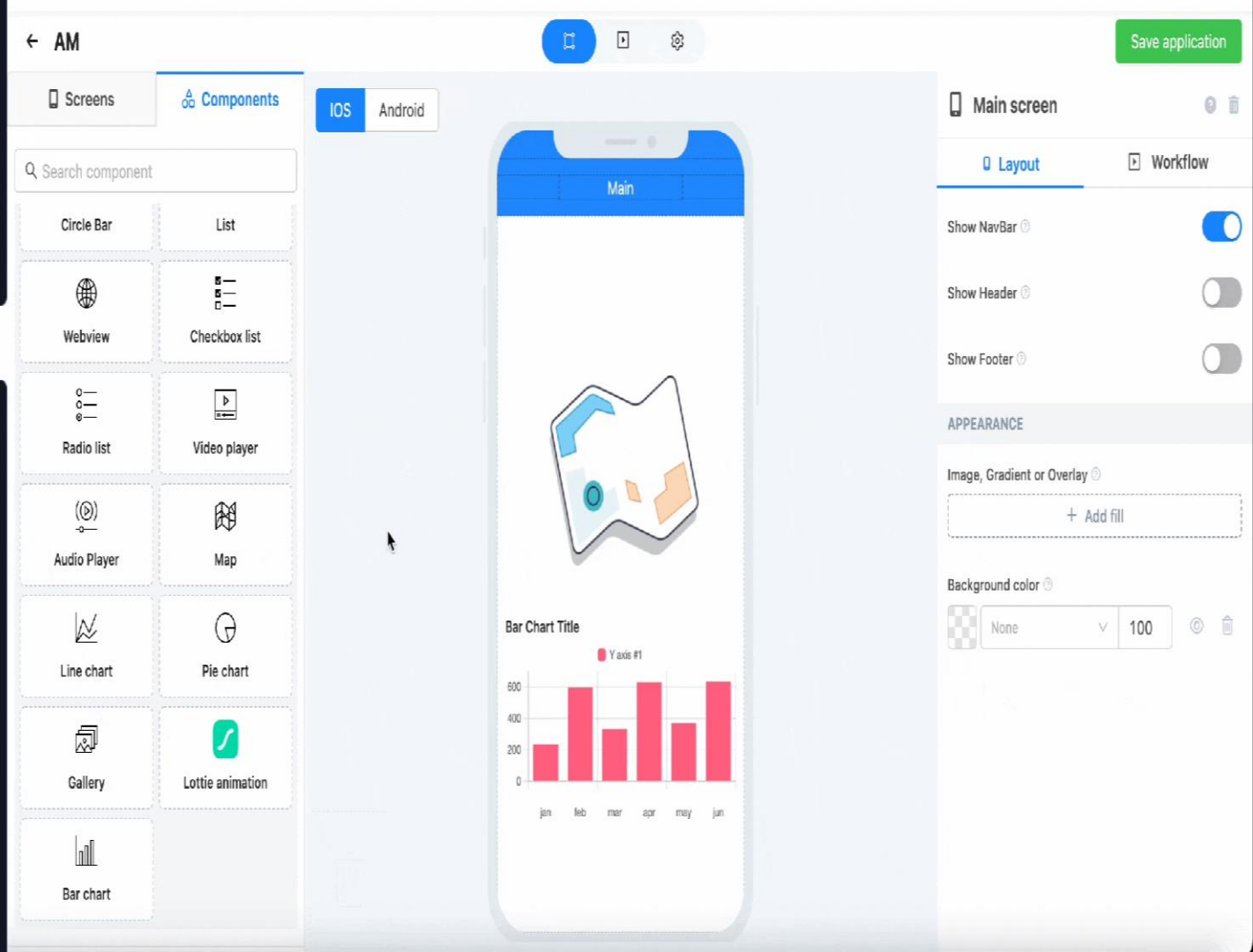
Trendy a budúcnosť softvérového vývoja

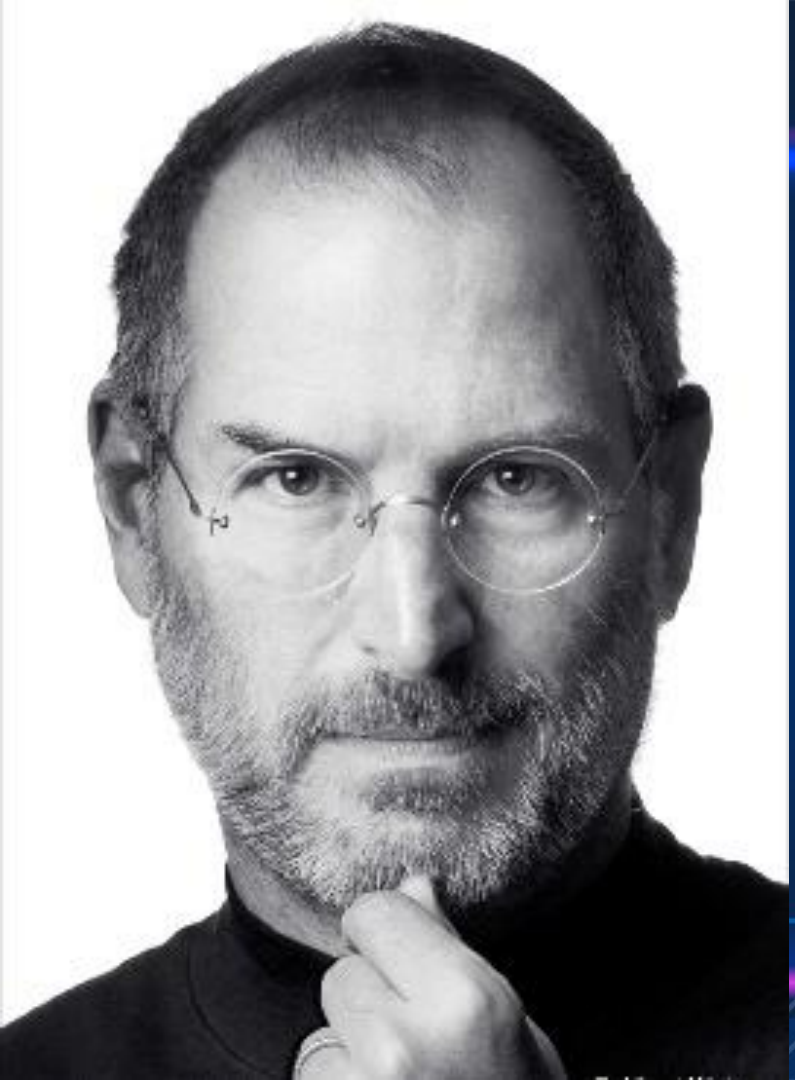
Umelá inteligencia (AI) v softvérovom vývoji

- AI vplýva na rôzne odvetvia vrátane programovania a softvérového vývoja
- AI **umožňuje**: automatizáciu rutinných úloh (generovanie kódu, detekcia chýb), asistovať pri písaní kódu, navrhovať riešenia programátorských problémov a odporúčať optimálne postupy pre dané úlohy
- AI umožňuje vývojárom **sústrediť sa na vyššiu úroveň** návrhu a architektúry softvéru
- AI **zrýchľuje proces** programovania a **zvyšuje produktivitu** vývojových tímov
- AI vytvára **možnosti pre nové typy aplikácií** (inteligentné chatboty a systémy rozpoznávania obrazu a reči)

Platformy so žiadnym alebo minimálnym kódom

- Low-Code, No-Code
- Zjednodušenie procesu vytvárania softvéru – minimalizujú alebo úplne odstránia potrebu písania kódu
- Aplikácie sa vytvárajú prostredníctvom vizuálneho programovania a metódou „drag & drop“





„Everybody in this country should learn to program a computer...
because it teaches you how to think.“

Steve Jobs, co-founder of Apple Inc.

ĎAKUJEM ZA POZORNOSŤ

ttoth@uniag.sk