# ÚVOD

V úvode autor stručne a výstižne charakterizuje stav poznania alebo praxe v oblasti, ktorá je predmetom záverečnej práce a oboznamuje čitateľa s významom, cieľmi a zámermi práce. Autor v úvode zdôrazňuje, prečo je práca dôležitá a prečo sa rozhodol spracovať danú tému

# JADRO

* 1. Jadro je hlavná časť práce a jeho členenie je určené typom práce. Vo vedeckých a odborných prácach má jadro spravidla tieto hlavné časti:
     1. súčasný stav riešenej problematiky doma a v zahraničí,  
        V časti Súčasný stav riešenej problematiky autor uvádza dostupné informácie a poznatky týkajúce sa danej témy. Zdrojom pre spracovanie sú aktuálne publikované práce domácich a zahraničných autorov. Podiel tejto časti práce má tvoriť približne 30 % práce.
     2. cieľ práce,  
        Časť Cieľ práce jasne, výstižne a presne charakterizuje predmet riešenia. Súčasťou sú aj rozpracované čiastkové ciele, ktoré podmieňujú dosiahnutie cieľa hlavného.
     3. metodika práce a metódy skúmania,   
        Časť Metodika práce a metódy skúmania spravidla obsahuje:
        1. charakteristiku objektu skúmania,
        2. pracovné postupy,
        3. spôsob získavania údajov a ich zdroje,
        4. použité metódy vyhodnotenia a interpretácie výsledkov,
        5. štatistické metódy.
     4. výsledky práce,   
        Výsledky práce a diskusia sú najvýznamnejšími časťami záverečnej práce. Výsledky (vlastné postoje alebo vlastné riešenie vecných problémov), ku ktorým autor dospel, sa musia logicky usporiadať a pri popisovaní sa musia dostatočne zhodnotiť. Zároveň sa komentujú všetky skutočnosti a poznatky v konfrontácii s výsledkami iných autorov. Ak je to vhodné, výsledky práce a diskusia môžu tvoriť aj jednu samostatnú časť a spoločne tvoria spravidla 30 až 40 % záverečnej práce.

# ZÁVER

* 1. V závere je potrebné v stručnosti zhrnúť dosiahnuté výsledky vo vzťahu k stanoveným cieľom.

1. Porovnanie jazykov C++ a Java

2. Porovnanie pôvodnej a novej platformy Windows CE / Android

3. Komplikácie, ktoré sa vyskytli pri portácií aplikácie

4. Postup testovania správania sa novej aplikácie voči pôvodnej

5. Príklady zdrojových kódov (porovnanie starého kódu s novým)

//TODO  
Našu firmu oslovil zákazník s požiadavkou, že potrebuje prerobiť svoju aplikáciu tak, aby ju bolo možné spustiť na mobilnom zariadení. Dôvodom, prečo oslovili práve nás, bol fakt, že túto aplikáciu vytvorila pre zákazníka práve naša firma. Zákazník požadoval vyšší výkon aplikácie a možnosť jej rozbehnutia na OS Android, aby sa používateľom aplikácie zjednodušilo používanie, keďže pôvodná aplikácia bola spúšťaná na fyzicky ťažkom prístroji, a keďže je používateľmi, teda zamestnancami zákazníka, používaná na dennej báze, zákazník sa rozhodol pre používateľsky priateľskejšie zariadenia, mobilné terminály Zebra TC56, na ktorých beží práve OS Android.

K danému problému sa dá pristupovať buď portáciou zdrojového kódu jedného jazyka do druhého, alebo priamo spúšťaním pôvodnej aplikácie pomocou virtuálneho stroja, ktorý prekompiluje pôvodný zdrojový kód do Java byte kódu, s ktorým následne pracuje Java Virtual Machine. Jedným z takýchto virtuálnych strojov je LLJVM (Low Level Java Virtual Machine), ktorý ale aktuálne podporuje len portáciu z jazyka C. Čo sa týka portácie zdrojového kódu aplikácie z jazyka C++ do jazyka Java, k tomuto problému možno pristupovať manuálne alebo mechanicky. Mechanicky pomocou nástrojov alebo softvérov, ktoré priamo urobia konverziu medzi jazykmi. Tento spôsob ale nie je praktický mimo triviálnych zadaní, keďže vygenerovaný kód by bolo potrebné skontrolovať a keďže človek, ktorý by kód kontroloval, do styku s týmto kódom nikdy neprišiel, v prvom rade by sa musel zorientovať v kóde. Výhodu by mali ľudia, ktorí už pracovali na pôvodnej aplikácií, a teda by im kód hovoril viac ako ľuďom, ktorí začali robiť až na portácií tejto aplikácie. Navyše by bolo potrebné byť oboznámený, ako portácia pomocou takéhoto nástroja či softvéru prebieha, keďže napríklad cyklus for v C++ môžeme v Jave prepísať aj ako klasický cyklus for, ale takisto môžeme použiť stream. Vtedy treba zvážiť, čo je vhodnejšie použiť, aj vzhľadom na čitateľnosť a možnosť budúcej opravy respektíve úpravy kódu. Pri komplexných aplikáciách zložených z viacerých modulov je teda výhodnejšie pristupovať ku portácií manuálne, pretože človek v prvom rade aspoň čiastočne rozumie prepisovanému kódu a dokáže zvážiť a rozhodnúť, aký ekvivalent v inom jazyku je najvhodnejšie zvoliť.

Informácií o postupoch alebo nejakých ustanovených a všeobecne prijatých krokoch ako postupovať pri riešení takéhoto problému veľa nenájdeme. Je to zrejme z toho dôvodu, že aplikáciu pre OS Android je možné vyvíjať aj v C++ pomocou Android NDK, teda Android Native Development Kit.

// TODO  
Pri rozhodovaní, v akom jazyku bude nová aplikácia naprogramovaná, sa zvažovalo viacero možností. Prvým jazykom, ktorý pripadal do úvahy, bola Java. Väčšina súčasných aplikácií pre OS Android je napísaná v jazyku Java a takisto je Java aj oficiálnym jazykom pre vývoj Android aplikácií. Ďalším jazykom na zozname bol C++ a to hlavne preto, že pôvodná aplikácia bola napísaná v tomto jazyku a taktiež veľa členov budúceho tímu malo skúsenosti s vývojom nedávnej Android aplikácie v jazyku C++ s využitím knižnice Qt a QML. Nakoniec, po zvážení hlavnými vývojármi a manažérmi, bol zvolený jazyk Java.