Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

Katedra informačního inženýrství

Diplomová práce

Tvorba e-learningového kurzu pro studenty střední školy

Bc. Radek Libovický

ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA V PRAZE

Katedra informačního inženýrství

Provozně ekonomická fakulta

ZADÁNÍ DIPLOMOVÉ PRÁCE

Radek Libovický

Informatika

Název práce

Tvorba e-learningového kurzu pro studenty střední školy

Název anglicky

Creation of the e-learning course for high school students

Cíle práce

Cílem literární rešerše diplomové práce je popis teoretických principů elektronického vzdělávání a tvorby e-learningového kurzu a poskytnoutí uceleného pohledu na celou problematiku elektronického vzdělávání.
Cílem praktické části diplomové práce je v prostředí LMS Moodle vytvořit e-learningový kurz Algoritmizace a programování v jazyce C pro studenty střední školy. Pro vyhodnocení elektronického kurzu bude vytvořen závěrečný test.

Metodika

Pro dosažení cíle diplomové práce budou nejprve shromážděny informace z odborné literatury týkající se tématiky e-learningu. Na základě získaných informací bude sepsána teoretická část práce v souladu s cílem práce. V praktické části bude vytvořen v LMS Moodle výukový kurz Algoritmizace a programování v jazyce C a následně otestován na studentech střední školy.

Doporučený rozsah práce

60 – 80 stran

Klíčová slova

E-learning, elektronické vzdělávání, elektronický kurz, LMS Moodle, střední škola

---

Doporučené zdroje informací

BAREŠOVÁ, Andrea a Petr SUDICKÝ. E-Learning ve vzdělávání dospělých: učení (se) s online technologiemi. Vyd. 1. Praha: VOX, 2012, 167 s. ISBN 80-863-2427-3.

DRLÍK, Martin a Petr SUDICKÝ. Moodle: kompletní průvodce tvorbou a správou elektronických kurzů. 1. vyd. Brno: Computer Press, 2013, 344 s. ISBN 978-80-251-3759-8.

EGER, Ludvík a Petr SUDICKÝ. Vzdělávání dospělých a ICT: aktuální stav a predikce vývoje. Vyd. 1. Plzeň: Nava, 2013, 120 s. ISBN 978-807-2114-283.

ZOUNEK, Jiří a Petr SUDICKÝ. E-learning: učení (se) s online technologiemi. Vyd. 1. Praha: Wolters Kluwer Česká republika, 2012, 248 s. ISBN 978-80-7357-903-6.

Předběžný termín obhajoby

2015/06 (červen)

Vedoucí práce

Ing. Dana Vynikarová, Ph.D.

Elektronicky schváleno dne 10. 11. 2014

Ing. Martin Pelikán, Ph.D.

Vedoucí katedry

Elektronicky schváleno dne 10. 11. 2014

Ing. Martin Pelikán, Ph.D.

Děkan

V Praze dne 20. 03. 2015

Čestné prohlášení

Prohlašuji, že svou diplomovou práci " Tvorba e-learningového kurzu pro studenty střední školy" jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 31. března 2015 _____

Poděkování

Rád bych touto cestou poděkoval vedoucí mé diplomové práce paní Ing. Daně Vynikarové, Ph.D. za cenné rady, odborné vedení a pomoc při zpracování práce. Dále bych chtěl poděkovat pedagogickému pracovníkovi ze Střední průmyslové školy v Kladně panu Ing. Jaroslavu Volšickému za jeho ochotu a umožnění otestovat vytvořený e-learningový kurz na jeho studentech.

Tvorba e-learningového kurzu pro studenty střední školy

Creation of the e-learning course for high school students

Souhrn

Cílem této diplomové práce je popsat teoretické principy elektronického vzdělávání a tvorby e-learningového kurzu a poskytnout ucelený pohled na problematiku elektronického vzdělávání. Praktická část této práce je realizována na základě informací vycházející z teoretické části a jejím cílem je vytvořit v prostředí LMS Moodle e-learningový kurz Algoritmizace a programování v jazyce C pro studenty střední školy. Kurz je aplikován na vybrané studenty třetích a čtvrtých ročníků oboru Elektronické počítačové systémy ze Střední průmyslové školy v Kladně. K vyhodnocení kurzu jsou následně vytvořeny testy, které jsou povinni výše zmínění studenti absolvovat. Pro získání dalších dat ke zhodnocení kurzu je studentům i pedagogickému pracovníkovi položeno několik otázek. Po vyhodnocení výsledků jsou navržena doporučení, která zvýší kvalitu e-learningového kurzu a eliminují případné nedostatky.

Summary

The goal of the thesis is to describe theoretical principles of e-learning and creating e-learning course and to provide comprehensive view on issues of e-learning. Practical part of the paper was based on information provided in theoretical part. The main goal for practical part is to create e-learning course Algorithms and Programming in C language in LMS Moodle directed to high school students. The course is applied to selected students of third and fourth grades studying Electronic computer systems from Technical high school in Kladno. Evaluation of the course is performed with mandatory tests. Deep result analysis and data collection is also based on brief interview with students and teacher. Final part of this thesis consists of recommendations to increase quality of e-learning course and to eliminate potential shortcomings.

Klíčová slova: E-learning, elektronické vzdělávání, elektronický kurz, LMS Moodle, střední škola, student, test

Keywords: E-learning, electronic education, electronic course, LMS Moodle, high school, student, test

Obsah

1 Úvod

Součástí dnešní a rychle se vyvíjející doby je neustálé sebevzdělávání. Vzdělávání neboli, proces získávání a osvojování znalostí a vědomostí, je považované za nutnost a provází jedince celým jeho životem. Dříve vzdělání představovalo pouze jakousi průpravu na budoucí zaměstnání. Oproti klasickým způsobům vzdělávání přichází nová cesta výuky prostřednictvím moderních internetových technologií ve formě e-learningu. Vzdělávání formou e-learningu nemusí každému zcela vyhovovat, avšak přesto přináší celou řadu výhod.

Toto téma je zvoleno z toho důvodu, že v současnosti představuje velmi aktuální téma a zároveň se neustále rozvíjí. Vzdělávání prostřednictvím e-learningových kurzů nachází uplatnění nejen ve vzdělávacích institucí, ale taktéž i v komerční sféře při různých školení zaměstnanců, atp.. Po příchodu na vysokou školu, kde jsou e-learningové kurzy běžnou záležitostí, si autor uvědomil, jak moc postrádal podobnou formu výuky již během studia na střední škole. Právě proto se autor rozhodl, že vytvoří e-learningový kurz pro studenty střední školy. S ohledem na autorovo zaměření je zvolena náplň kurzu týkající se programování. Konkrétně je vytvořen kurz s názvem Algoritmizace a programování v jazyce C.

Výše zmíněný kurz je aplikován na dvě vybrané třídy ze Střední průmyslové školy v Kladně. Jedná se o střední školu, kterou autor sám absolvoval, tudíž detailně znal náplň jednotlivých předmětů programátorského charakteru a věděl, že tento kurz by pro vybrané studenty mohl být přínosný.

Tato diplomová práce nejprve charakterizuje e-learning samotný, poté se zabývá modelem pro tvorbu kurzu a dále Learning management systémy. Dle tohoto modelu je následně v praktické části práce vytvořen výše zmiňovaný e-learningový kurz, který by mohl zlepšit kvalitu výuky na zmíněné střední škole.

2 Cíl práce a metodika

Cíl práce

Cílem literární rešerše diplomové práce je popis teoretických principů elektronického vzdělávání a tvorby e-learningového kurzu a poskytnutí uceleného pohledu na celou problematiku elektronického vzdělávání.

Cílem praktické části diplomové práce je v prostředí LMS Moodle vytvořit e-learningový kurz Algoritmizace a programování v jazyce C pro studenty střední školy. Pro vyhodnocení elektronického kurzu bude vytvořen závěrečný test.

Metodika

Pro dosažení cíle diplomové práce bylo nejprve nezbytné zmapovat teoretická východiska. K vytvoření kvalitní literární rešerše autor nejdříve potřeboval vyhledat sekundární zdroje z odborné literatury a ověřených internetových zdrojů. Za pomoci klíčových slov se na zvolené téma v knihovním systému dohledala nezbytná literatura a prostřednictvím internetového vyhledávače byly nalezeny související odborné články a weby. Po shromáždění a následném prostudování všech zdrojů, byla sepsána charakteristika teoretických principů elektronického vzdělávání, tvorby e-learningového kurzu a poskytnutí uceleného pohledu na celou problematiku elektronického vzdělávání.

V praktické části byl vytvořen v LMS Moodle zmiňovaný výukový kurz Algoritmizace a programování v jazyce C dle pravidel vyplývajících z teoretické části práce. Vytvořený kurz byl aplikován na Střední průmyslové škole v Kladně a po domluvě s pedagogickým pracovníkem byl kurz představen vybraným studentům třetích a čtvrtých ročníků z oboru Elektronické počítačové systémy. Pro zhodnocení kurzu byly vytvořeny testy, které studenti absolvovali, dále zodpověděli tři otázky pro zpracování zpětné vazby autorovi kurzu. Rovněž byl se zmíněným pedagogickým pracovníkem proveden krátký rozhovor zaměřený na přínos vytvořeného kurzu.

3 Charakteristika e-learningu

Tato kapitola se nejprve zabývá definováním pojmu e-learning a stručným nástinem jeho vzniku a historie. Poté uvádí jeho základní formy a na závěr shrne výhody a nevýhody elektronického vzdělávání.

3.1 Definování pojmu

Pro porozumění této práci je nejprve nutné vědět, co vlastně e-learning znamená. Písmeno „E“ je zkratka pro anglický výraz electronic neboli elektronické. Slovo learning lze přeložit jako vzdělávání. Takže e-learning je elektronické vzdělávání. V praxi se pojem e-learning zažil tak dobře, že se již téměř vůbec nepřekládá. (Barešová, 2011)

Co se týče přesné definice, tak tady nastává problém v tom, že s vývojem informačních technologií se měnila i samotná definice e-learningu. V dnešní době lze najít velké množství definic e-learningu a některé mají rozdílné pohledy na to, co to opravdu e-learning je.

Autor volí následující tři definice, které mu připadají nejpřesnější:

„E-learning chápeme jako multimediální podporu vzdělávacího procesu s použitím moderních informačních a komunikačních technologií, které je zpravidla realizováno prostřednictvím počítačových sítí. Jeho základním úkolem je v čase i prostoru svobodný a neomezený přístup ke vzdělávání.“ (Kopecký, 2006)

„E-learning je vzdělávací proces, využívající informační a komunikační technologie.“ (Barešová, 2011)

„E-learning je vzdělávání spojené s informačními a komunikačními technologiemi.“ (Eger, 2012)

3.2 Historie

Vznik e-learningu byl umožněn příchodem nových informačních a komunikačních technologií. První náznak e-learningu lze zaznamenat na konci 19. století, kdy bylo

sestrojeno první rádio, které přenášelo informace rádiovými vlnami, a v závislosti na charakteru vysílání i vzdělávat posluchače. (Johnson, 2011)[1]

Koncem 60. let 20. století se zrodili výukové automaty. V České republice by jeden takovýto automat vyroben také a jmenoval se Unitutor. Probíraná látka byla rozvržena na stránky, které měli na svém konci kontrolní otázky s výběrem správné odpovědi. Jedinou formou zpětné vazby zde byla informace o správnosti či chybnosti řešení. Tyto vyučovací automaty se pro svou nízkou účinnost a vysokou složitost přestaly používat. (Barešová, 2011)

Teprve až s příchodem osmibitových mikropočítačů a později osobních počítačů, zaznamenal e-learning velký rozmach programů, které navázali na Unitutor. Nejprve se jednalo o programy na zkoušení, což ale samozřejmě nestačilo a vývoj pokračoval dál, až vznikly učící i zkoušející programy. Také se začíná prověřovat teorie, zda by počítač dokázal nahradit učitele. Dalšími projekty v tomto období byly pokusy o vytvoření z jednotlivých lekcí celé kurzy. Takovouto inteligentní výukovou aplikaci by bylo možné kdykoliv přerušit a později v ní zase pokračovat ze stejného místa. Dále by dlouhodobě řídila výuku studujícího až do ukončení kurzu a zaznamenávala by jeho výsledky a cestu celým kurzem. Bohužel tvorba takových kurzů byla velmi náročná, drahá a nepřinášela očekávané výsledky. Tudíž se od vývoje inteligentních výukových aplikací odstoupilo. (Barešová, 2011)

Postupem času zaznamenával e-learning mnoho dalších významných etap. Přičemž té nejzásadnější napomohl vznik internetu a rozvoj internetových technologií. Díky tomu se studenti mohou v dnešní době přihlásit do kurzu, který využívá moderní grafiku, animace, zvuky a může mít integrované zcela nezávislé programy. Na přelomu 20. a 21. století se informační zdroje v papírové podobě začaly transformovat do podoby elektronické, což zlepšilo podmínky pro e-learning samotný. V současnosti je e-learning využíván nejen ve školství, ale i v soukromém sektoru. (Barešová, 2011)

[1] S tímto tvrzením, by ne každý odborník na e-learning souhlasil.

3.3 Základní formy

Elektronické vzdělávání je závislé na technologicko-komunikačních možnostech organizace i studenta. Lze najít několik rozdílných forem e-learningu kde každá forma má své výhody a nevýhody. Tato kapitola všechny základní formy stručně charakterizuje a uvede jejich nejčastější využití.

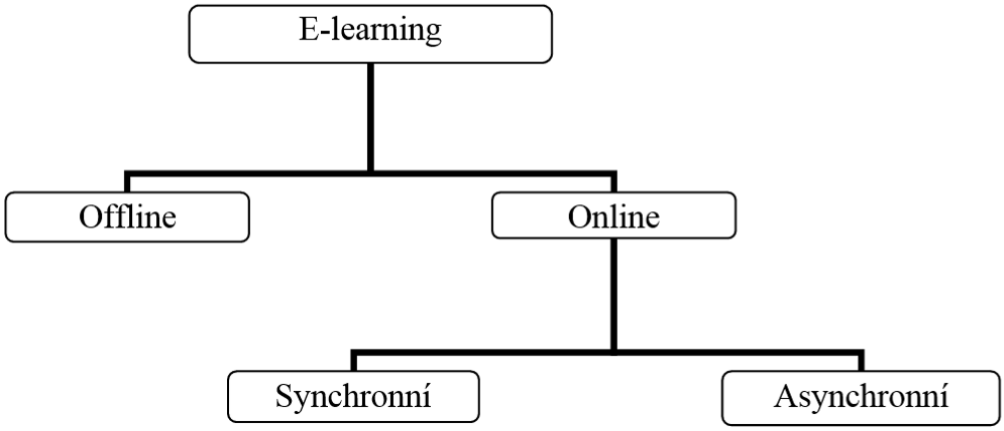


Schéma 1: Základní rozdělení e-learningu (Kopecký, 2006)

Výše zmíněné schéma (č. 1) rozděluje e-learning na dvě základní formy, tj. to offline a online. Přičemž online forma se dále dělí na synchronní a asynchronní.

3.3.1 Offline

Nejstarší formou e-learningu s využitím počítačových technologií je forma offline. Offline formu lze někdy též najít pod zkratkou CBT (computer based training), což je volně přeloženo jako vzdělávání za podpory počítačů. Jak již vyplývá z názvu offline, používaná zařízení pro vzdělávání nejsou připojeny do internetové sítě. Offline forma e-learningu nachází své využití zejména při domácím studiu prostřednictvím určitého výukového programu. Tyto kurzy a jejich studijní materiály jsou distribuovány především ve formě disků CD (compact disc), DVD (digital versatile disc), flash disků či externích HDD (hard disc drive) a uživatel se vzdělává sám bez jakéhokoliv dohledu. V dnešní době se lze s kurzy tohoto typu setkat v podobě přílohy k zakoupené odborné knize, avšak nebývá to pravidlem. (Kopecký, 2006)

3.3.2 Online

Pravým opakem offline formy je forma online a základním předpokladem je nutnost připojení zařízení využívaného pro výuku k internetu či intranetu. Online formy bývají také označovány zkratkou WBT (web based training), neboli volně přeložené jako kurzy poskytované za pomoci internetu. (Barešová, 2011)

Do online formy lze zařadit také tzv. virtuální třídu, která však spadá i do synchronních forem, proto se jí více věnuje kapitola 3. 3. 3. 1.

3.3.3 Synchronní

Dále se do základních forem e-learningu řadí forma synchronní jejímž principem je to, že komunikace mezi studentem a učitelem probíhá v reálném čase. U synchronních forem je opět předpoklad připojení k internetu, avšak naopak od online formy je nezbytné připojení zároveň vyučujícího. Aby mohla proběhnout výuka, studenti i učitel musejí být připojeni k internetu ve stejný čas, avšak každý z nich se může vyskytovat na jiném místě. Výhodou synchronní formy je okamžitá zpětná vazba, která je zajištěna přímou interakcí mezi vyučujícím a studenty. Jako příklad je možné uvést audio/video konferenci, internetové telefonování, chat a v neposlední řadě virtuální třídu. (Kopecký, 2006)

3.3.3.1 Virtuální třída

Právě virtuální třída je nejcharakterističtější formou synchronního elektronického vzdělávání. Princip tohoto vzdělávání je stejný jako při klasické vyučovací hodině ve škole, avšak v tomto případě učebna existuje pouze ve virtuální podobě a studenti a učitel do ní vstupují prostřednictvím svých počítačů s internetovým připojením. Největší výhoda spočívá v neomezené vzdálenosti mezi všemi účastníky a během výuky probíhá okamžitá interakce, což je zásadní rozdíl od asynchronních kurzů, které zpětnou vazbu nemají okamžitou. Naopak nezbytným požadavkem pro realizaci tohoto kurzu je stabilní a dostatečně rychlé internetové připojení. Další nevýhodou je povinná přítomnost všech účastníků výuky ve virtuální třídě v předem dohodnutý čas. Tato forma e-learningu najde své využití nejen ve výuce studentů, ale též je vhodná pro realizaci meetingů, prezentací i školení zaměstnanců. (Podlahová, 2012)

3.3.4 Asynchronní

Pravým opakem synchronní výuky je výuka asynchronní. I zde je potřeba připojení k internetu. Výuka zde neprobíhá v reálném čase, tudíž dané připojení je potřebné pouze při stahování studijních pokladů, či různých zadání a odesílání svých výstupů (vyplněné testy, vypracované úkoly). Jak již bylo naznačeno výše, zpětná vazba je na značně horší úrovni, než u synchronní formy a probíhá pouze pomocí e-mailu, diskuzních fór a dalších podobných komunikačních prostředků. Asynchronní formu není vhodné používat jako hlavní vzdělávací prostředek pro studium odborné komplexní problematiky. Jelikož se jedná o samostudijní způsob sebevzdělávání, studenti mohou mít velký problém s motivací a soustředěností. (Podlahová, 2012)

Avšak naopak velkou výhodou těchto kurzů je, že student může studovat odkudkoliv a kdykoliv. Z tohoto důvodu bývají asynchronní formy kurzů označovány jako 24/7/365 (přístup 24 hodin denně, 7 dní v týdnu a 365 dní v roce). Pro studenta je taktéž pozitivním faktem to, že se může vracet k dřívější látce a určovat si vlastní tempo studia. Příkladem asynchronní formy lze označit studium webových prezentací, samostudijní kurzy, diskuzní fóra, atp. (Barešová, 2011)

3.3.5 Blended learning

Blended learning není přímo typická forma e-learningu, avšak v dnešní době se s tímto pojmem setkává čím dál tím více lidí. Jak již z názvu vyplývá, jedná se o smíšené vzdělání, čili kombinaci e-learningu s klasickým vyučováním v prezenční formě. Blended learning představuje aktivní podporu studia, kdy studenti mohou používat namísto skript interaktivní platformy, jako jsou např. LMS, diskuzní fóra, vizuální prezentace, atp. (Barešová, 2011)

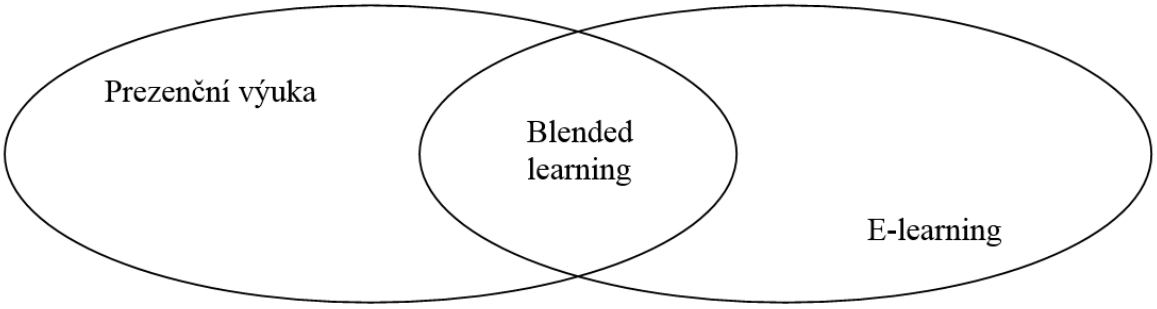


Schéma 2: Blended learning (Kopecký, 2006)

Schéma č. 2 zobrazuje průniky dvou množin, kdy jedna množina představuje klasickou výuku ve fyzických učebnách a druhá klasickou formu e-learningového kurzu. Průnik těchto dvou množin reprezentuje blended learning. Klasická výuka umí studenty lépe motivovat, zajistit zpětnou vazbu, plynulou výuku a disciplínu. Naopak e-learningový kurz disponuje mobilitou, nižší nákladovostí a umožňuje samostudium.

Vznik blended learningu šel ruku v ruce s vývojem e-learningu. Nyní nachází největší uplatnění u jazykových kurzů a školení (bezpečnost práce, řízení dopravních prostředků). Popularitu si získal především svou neomezeností a nabídkou mnoha možností přístupů, díky těmto faktům má nyní blended learning velký potencionál a široké uplatnění v budoucnu. V dnešní době jsou typickými institucemi využívající blended learning univerzity, kde studenti mají přístup k elektronickým kurzům během prezenční výuky. (Barešová, 2011)

3.4 Výhody a nevýhody e-learningu

Ačkoliv e-learning přináší spoustu komplexních řešení, které mají velké výhody, nese sebou i některá negativa. Názory na tuto problematiku se mohou lišit dle aktérovy role v e-learningu, neboť zmíněnou výhodu mohou někteří uživatelé vnímat jako nevýhodu a naopak. Je důležité rozhodnout, v jakých případech e-learning, coby výukový nástroj použít a kdy je efektivnější použít klasičtější výukovou metodu.

3.4.1 Pohled studenta

Student je nejdůležitějším aktérem v celém e-learningu, proto je jeho pohled na celý kurz velmi podstatný. V tomto systému spatřuje za nejhlavnější pozitivum především

neomezený přístup (tzn., že kurz je možné využívat 24 hodin denně, 7 dní v týdnu, 365 dní v roce) a možnost se přihlásit odkudkoliv na světě, pokud je zajištěno připojení k internetu. Dalším kladem je, že se student může sám individuálně přizpůsobit výuce a sám si určuje její tempo i časovou náročnost. Jedinou podmínkou je dodržení termínů odevzdání zadaných úkolů či testů. Naopak student velmi často ztrácí pozornost a nesoustředí se na učení vzhledem k online komunikaci v rámci sítě, nevzniká dostatečná motivace k samostudiu a chybí okamžitá zpětná vazba. Také nehlídané vyplňování testů láká studenta k podvádění. (Zounek, Sudický, 2012) Samozřejmě takovýchto výhod a nevýhod existuje mnohem více, některé z nich lze dále vidět v tabulce č. 1.

Tabulka 1: Výhody a nevýhody e-learningu z pohledu studenta (Zounek, Sudický, 2012)

VÝHODY	NEVÝHODY
• *„rychlý a snadný přístup k informacím a učebním zdrojům;* • *možnost rychlého vyhledání informací;* • *snadné uložení, zpracování, úprava, archivace materiálů;* • *možnost učit se kdykoliv a kdekoliv;* • *individualizace a flexibilita učení;* • *sdílení, vědění a spolupráce při učení;* • *zvyšování počítačové a informační gramotnosti;* • *úspora času, zdrojů a finančních prostředků;* • *snadná komunikace se všemi aktéry;*	• *cena (některých) technologií, hardwaru;* • *ceny za připojení internetu;* • *nedostatečné znalosti a dovednosti ve využívání online technologií;* • *negativní postoj k ICT obecně;* • *rozpor mezi učebním stylem studenta a použitým technologických řešením;* • *nedostatečná motivace, neschopnost samostatného učení;* • *přehlcení množstvím informací nebo učebních materiálů;* • *nesoustředěnost na učení vzhledem k neustálé online komunikace v rámci sítě;* • *plagiátorství a podvádění;* • *zdravotní problémy;“*

3.4.2 Pohled vyučujícího

Dalším aktérem e-learnignu je vyučující, jehož pohled na kurz není stejný jako u studenta. Vyučujícímu e-learning usnadňuje distribuci studijních materiálů a jejich modifikaci a výuku může lépe korigovat mezi kolegy i studenty, čímž je zajištěn jednotný sylabus předmětu. Avšak ne vždy se využívání online prostředků hodí pro výuku všech předmětů a ani sebelepší kurz nemůže nahradit přímou interakci studenta s učitelem. Dále je nutné konstatovat, že i v dnešní době se stále nacházejí pedagogičtí pracovníci, kteří neovládají moderní počítačové technologie na potřebné úrovni. (Zounek, Sudický, 2012) Další výhody a nevýhody pro vyučující lze dohledat v tabulce č. 2.

Tabulka 2: Výhody a nevýhody e-learningu z pohledu vyučujícího (Zounek, Sudický, 2012)

VÝHODY	NEVÝHODY
• *„možnosti tvorby, archivace, distribuce, inovace (multimediálních) učebních materiálů;* • *prostředek řízení výuky a sledování (diagnostiky) procesu učení studentů;* • *podpora komunikace, její sledování i archivace a využití ve výuce;* • *externí aktéři ve výuce (pomocí videokonference apod.);* • *kooperativní výuka/spolupráce s vyučujícími i studenty z jiných institucí;* • *další vzdělávání, konzultace s kolegy;* • *členství v odborných (virtuálních) komoditách;* • *podpora inovativních didaktických postupů;*	• *nedostatečné znalosti a dovednosti v práci s online technologiemi a ICT obecně;* • *příliš rychlý a proměnlivý svět technologických inovací;* • *potlačení lidské komunikace a interakce;* • *nevhodnost online prostředků pro výuku všech oborů nebo předmětů (témat);* • *nejasné představy o pedagogickém využití online technologií ve výuce a učení;* • *náročná příprava (multimediálních) učebních materiálů;* • *závislost na technickém zabezpečení/vybavení;* • *plagiátorství;“*

3.4.3 Pohled instituce

Poslední zkoumaný pohled je ze strany instituce, kterou nejvíce zajímá finanční náročnost. Nákladnější počáteční investice se instituci během několika málo let vrací vlivem snížení nákladů na provoz (menší mzdové náklady, nižší spotřeba energie, atd.). Dalším pozitivním faktem je získání konkurenční výhody, neboť propracovaný e-learning zajistí kvalitnější vzdělání či zaškolení oproti ostatním institucím. Rizikovým faktorem může být náhlá technologická závada, čímž ohrozí plynulý průběh kurzu. (Zounek, Sudický, 2012) V níže uvedené tabulce č. 3 je možné vidět další výhody a nevýhody e-learningu z pohledu instituce.

Tabulka 3: Výhody a nevýhody e-learningu z pohledu instituce (Zounek, Sudický, 2012)

VÝHODY	NEVÝHODY
• *„přístup studentům k učebním zdrojům i k pracím vyučujících;* • *administrativa v elektronické podobě;* • *motor změn v celé instituci;* • *ekonomický zisk;* • *snížení nákladů na provoz;* • *konkurenční výhoda;*	• *absence nebo špatná kvalita technologické infrastruktury;* • *omezená funkčnost či poruchovost technologií;* • *velké počáteční náklady;* • *nepřipravenost organizace a jejich členů/učitelů;* • *nepřehledná nabídka nástrojů;* • *neseriózní dodavatelé;“*

4 Tvorba kurzu dle modelu ADDIE

Tato kapitola se zabývá tvorbou e-learningového kurzu. Pro tvorbu kvalitního e-learningového kurzu neexistuje žádné přesně dané pravidlo, ale dle (Zounek, Sudický, 2012) lze najít řadu doporučovaných postupů a modelů, dle kterých lze takovýto kurz vytvořit. Jedním velmi často využívaným modelem je právě model ADDIE. Akronym ADDIE je výpisem jednotlivých fází tohoto modelu v angličtině a jsou to: Analysis, Design, Development, Implementation, Evaluation tedy česky analýza, návrh, vývoj, realizace a hodnocení. V dnešní době lze nalézt značné množství modifikací tohoto modelu, kde každá varianta má své klady a zápory. (Castagnolo, 2007). Avšak tato práce se bude zabývat jen onou základní verzí modelu ADDIE.

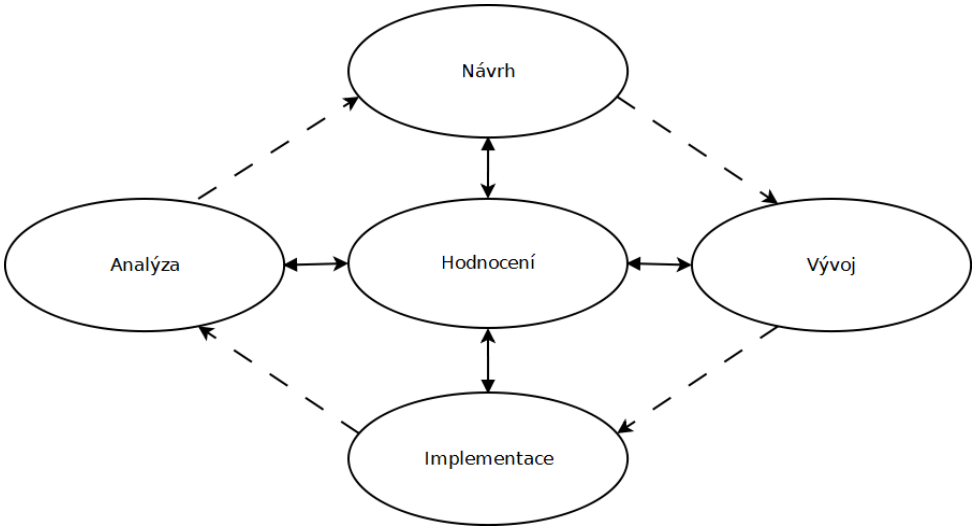


Schéma 3: Model ADDIE (Stárková, 2012)

Na schématu č. 3 je zobrazen model ADDIE, který znázorňuje jeho jednotlivé fáze. Model má čtyři fáze v kruhovém rozpoložení, neboť je to nekonečně se opakující model, který vede k neustálému zdokonalování samotného kurzu. Poslední fáze je uprostřed kruhu a vyhodnocuje nedostatky každé fáze zvlášť. Při znalosti nedostatků kurzu mohou autoři v jednotlivých fázích provézt potřebné změny a tím nedostatky odstranit.

4.1 Analýza (Analysis)

První a téměř nejdůležitější fází celého modelu ADDIE je analýza a zároveň se od této fáze odvíjí všechny ostatní. Spočívá hlavně v určení vzdělávacích cílů, volbě výukového prostředí, určení cílové skupiny studentů, nutných znalostí pro vstup do kurzu.

Na webu časopisu Inflow jsou uvedeny otázky, které by si měl každý autor ve fázi analýzy položit:

- *„Pro koho je kurz určen?*
- *Jaká jsou specifika cílové skupiny? Jaké je věkové rozložení této skupiny? Jaká je úroveň vzdělání v rámci této skupiny? O jakou profesní skupinu se jedná?*
- *Jaké jsou předpokládané vstupní znalosti účastníků?*
- *Co má být cílem kurzu? Jaké mají mít účastníci výstupní znalosti?*
- *V jakém kontextu má být kurz realizován? Proč mají účastníci kurz absolvovat? Jedná se o povinný kurz v rámci vzdělávací instituce nebo v rámci zaměstnání? Jedná se o zájmový kurz?*
- *Jaký bude mít kurz přínos pro cílovou skupinu? Co od něj účastníci očekávají? Čím jsou účastníci pro absolvování kurzu motivováni?“* (Sedláčková, 2011)

Po zodpovězení těchto otázek lze získat přesnější představu o účastnících kurzu a jejich dispozicích. Teprve poté lze navrhnout konkrétní výukový projekt plně korespondující se zadanými cíli. (Sedláčková, 2011)

4.1.1 Stanovení výukového cíle

V této části se autor konkrétněji zaměřuje na cíl kurzu, který je pro celý další postup kritický. Cíl by měl být stanoven tak, aby zajistil vysokou motivaci studentů ke vzdělávání, vedl k efektivnější organizaci celého kurzu, byl reálně dosažitelný, atp. V případě rozsáhlejšího kurzu je třeba rozdělit cíl na dílčí části. Dále je nutné brát v úvahu vstupní znalosti studentů před kurzem. (Zounek, Sudický, 2012)

4.2 Návrh (Design)

Druhou fází v tvorbě kurzu dle modelu ADDIE je návrh, který úzce navazuje na fázi analýzy. Je zde proveden systematický proces vypracovávající detailní návrh daného kurzu, čímž vzniká jakýsi prototyp kurzu, který je třeba otestovat tzv. na nečisto. Tato fáze by měla dále obsahovat precizně naplánovaný harmonogram přípravy kurzu s přesně specifikovanými úkoly a jejich časovou náročností. Z ekonomického hlediska by se nemělo zapomenout na rozvržení finančních nákladů, který bývají často kritických faktorem. (Sedláčková, 2011) Eger doplňuje, že je mimo výše zmíněné dále nutné zanalyzovat veškeré dokumenty a informace týkající se dané tématiky e-learningového kurzu a zpracovat návrh metod pro evaluaci kurzu. (Eger, 2012)

4.2.1 Časové rozvržení

Časové rozvržení se podílí na celkové úspěšnosti celého kurzu. V závislosti na pokročilosti studentů, charakteru a obtížnosti vyučovaného předmětu je nutné najít správnou intenzitu výuky, tak aby byly všechny požadavky na stanovené úkoly splněny. Kurz lze plánovat několika možnými způsoby. Jednou z variant je kontinuální výuka po celou dobu trvání kurzu. Dále pak kombinace intenzitního studia s individuálními pracemi studenta. (Zounek, Sudický, 2012)

4.2.2 Proces výuky a učení

Průběh a způsob výuky se odvíjí především od stanovených cílů, obsahu kurzu a cílové skupiny studentů. To znamená, že dva identické kurzy s různým procesem výuky mohou v reálné podobě dosáhnout naprosto rozdílných výsledků. Je doporučeno rozdělit kurz na jednotlivé moduly, které se studentům zpřístupňují vždy až po úspěšném splnění předchozího modulu. (Zounek, Sudický, 2012)

4.2.3 Hodnocení studentů

Hodnocením studentům se zjišťují jejich vědomosti, dovednosti, postoje i kompetence. Mezi typické metody hodnocení patří test či písemná zkouška ohodnocená známkou, ale též slovní hodnocení, které bývá daleko přesnější než známka, avšak obtížnější na formulaci. (Zounek, Sudický, 2012)

4.3 Vývoj (Development)

Fáze vývoje je z časového a kapacitního hlediska nejnáročnější a zároveň jsou všechny realizované kroky nejviditelnější. Dle (Kopecký, 2006) se zde jedná o vývoj kurzu dle předem stanoveného scénáře vytvořeného v předchozích dvou fázích a jeho následné uložení do zvoleného výukového systému.

Zároveň dochází k přípravě technologické infrastruktury a potřebných počítačových programů, k tvorbě konkrétních výukových materiálů, objektů a služeb a k vybudování komunikačního prostředí. Na závěr této fáze je již možné provést zkušební otestování vytvořeného kurzu, které odhalí případné nedokonalosti. Veškeré tyto zjištěné nedostatky musejí být před zahájením další fáze (realizace) neprodleně vyřešeny a opraveny. (Sedláčková, 2011)

4.4 Realizace (Implementation)

Fáze realizace je zásadním krokem, při kterém se vytvořený kurz dostává do ostrého provozu, tj. pro cílovou skupinu studentů. Eger ve své knize o této fázi uvádí, že *„zde řešíme již vlastní poskytnutí – dodání vzdělávacího objektu – kurzu uživatelům včetně instrukcí, jak jej použít.*

- *Vybíráme postup, jak integrovat objekt do kurzu či modulu atd.*
- *Vytvoříme plán pro implementaci a potom jej realizujeme v praxi.*

Zvažujeme nejenom technickou stránku věci, ale z pohledu andragogiky i to, zda objekt bude využíván při samostudiu, v kolaborativním učení se nebo v kurzu, který je řízen, tutorován.“ (Eger, 2012)

Sedláčková na fázi realizace pohlíží z poněkud jiného úhlu. Před započetím samotné výuky je nezbytné provést ještě několik nezbytných úkonů. O existenci kurzů, které nejsou povinně nastaveny vzdělávací osnovou, je nutné vzbudit povědomí u potencionálních studentů a kurz dostatečně propagovat. Poté se z možných zájemců provede selekce těch, kteří splňují všechny kladené požadavky na absolvování kurzu. Daná selekce může probíhat několika různými metodami. Obecně lze konstatovat, že mírnějším způsobem pro výběr studentů je splnění formálních požadavků, jako například absolvování

konkrétního předmětu či kurzu, dokončení stupně vzdělání nebo profesního zařazení. Při přísnějších výběrech je navíc potencionálním uchazečům položen vstupní test či dotazníkové šetření pro ověření jejich skutečných znalostí. (Sedláčková, 2011)

U zpoplatněných kurzů se bere v potaz stránka finančního vyrovnání od studujícího. Pokud jsou již veškeré předchozí kroky naplněny, zbývá pouze distribuovat vstupní pokyny (např.: přihlašovací jméno, heslo a další instrukce). V konečné fázi realizace se všem zúčastněným zpřístupní výukový systém a zahájí se výuka. Právě nyní lze vidět přínos modelu ADDIE, neboť pokud je model sestaven správně, měl by kurz obsáhnout veškerá témata potřebná pro studium a reflektovat současné schopnosti studenta. (Sedláčková, 2011)

4.5 Hodnocení (Evaluation)

Poslední fází je fáze hodnocení, která je zaměřena na ohodnocení přiměřenosti a účelnosti vzdělávacího kurzu. Na evaluaci lze pohlížet ze dvou hledisek. První aspekt hodnotí celkové funkčnost kurzu. Druhé pojetí je složitější na vyhodnocení a zkoumá, zda studující dosahují absolvováním kurzu požadovaných cílů. Především u e-learningu se doporučuje rozčlenit hodnocení kurzu ze tří pohledů, tj. pohled studenta, pohled učitele a pohled organizace. (Eger, 2012)

Jak již bylo výše zmíněno, vyhodnocení naplnění požadovaných cílů je složitější proces, proto ho lze zjišťovat několika způsoby, jako jsou například závěrečné testy, dotazníkové šetření, atp.

Evaluaci je také možné rozdělit na formativní a sumativní, přičemž formativní evaluace hodnotí každou fázi modelu ADDIE zvlášť a jejím cílem je tak uvést kurz do co nejlepšího stavu ještě před spuštěním. Naopak sumativní evaluace přináší výsledky až od absolventů kurzu, kdy se hodnotí nejen objektivní faktory (studijní výsledky), ale i subjektivní pocity studentů (naplnění očekávání, spokojenost s kurzem, srozumitelnost a celkové ohlasy). Pokud evaluace zjistila negativní výsledky o kurzu, provádí se plán úprav, kdy je nutné celou tvorbu kurzu ve všech fázích modelu ADDIE zopakovat dle nových poznatků. (Sedláčková, 2011)

5 Learning management system

V této kapitole autor nejprve vysvětlí pojem Learning management system (dále jen LMS), poté LMS rozčlení z licenčního hlediska na komerční systémy a systémy s licencí Open source. Dále budou uvedeny nejpodstatnější výhody a nevýhody LMS. Závěr kapitoly bude věnován LMS Moodle, což je systém, ve kterém bude provozován kurz vytvořený v praktické části diplomové práce.

LMS se vyvinulo z WBT a používá se pro kompletní řízení výuky, zahrnuje v sobě činnosti od plánování vzdělávacích aktivit, přes distribuci kurzu, až po koordinaci studentů celým studiem. (Barešová, 2011)

Havlíček na LMS pohlíží tak, že: *„Usnadňují tvorbu, používání a správu online kurzů tím, že poskytují:*

- *rozhraní, umožňující vytvářet prezentaci kurzu,*
- *soubor výukových nástrojů, které usnadňují studium, komunikaci a spolupráci,*
- *soubor administrativních nástrojů, které pomáhají učiteli v procesu správy, vedení a zlepšování kurzu.“* (Havlíček, 2007)

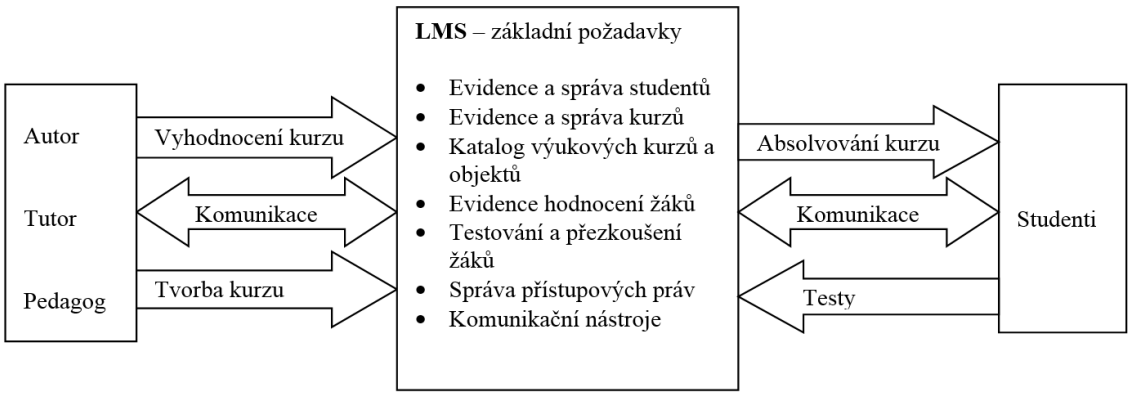


Schéma 4: Model LMS (Burian, 2014)

Schéma č. 4 představuje schéma modelu LMS, z kterého lze vidět přístup dvou skupin, tj. skupina autorů, tutorů a pedagogů, což je v mnoho případech tatáž osoba a skupina studentů. Skupina vyučujících vytváří a vyhodnocuje daný kurz a na studentech

je pak absolvování kurzů, které prokáží testy a závěrečnými pracemi. Obě tyto skupiny přes LMS vzájemně komunikují.

K vykonávání požadovaných funkcí LMS (administrace studia, prohlížení a nahrávání studijních materiálů, evidence studijních výsledků, vykonávání studijních aktivit, atp.) se využívá konkrétních nástrojů. Těmi jsou:

- *„nástroje pro prohlížení a vyhledávání na webu;*
- *nástroje synchronní a asynchronní komunikace;*
- *nástroje pro podporu personalizovaného učení;*
- *nástroje sdílení zdrojů a výukového obsahu;*
- *nástroje pro tvorbu výukových objektů;*
- *nástroje pro administraci studia;*
- *nástroje pro hodnocení a evaluaci.“* (Zounek, Sudický, 2012)

Dle Barešové (2011) existuje nemalé množství LMS, a proto je při tvorbě elektronického kurzu důležité vybrat ten nejvhodnější. Mezi nejznámější patří LMS eDoceo, LMS iTutor, LMS Oracle LMS iLearning, LMS Lotus IBM, LMS Sakai, LMS Blackboard, LMS Microsoft Class Server a v neposlední řadě LMS Moodle, kterému je podrobněji věnována kapitola 5.5.

5.1 Komerční systémy

Komerční systémy jsou takové, kdy je zdrojový kód ve vlastnictví výrobce a uživatel má přístup pouze ke zkompilované verzi, kterou nesmí nikterak modifikovat bez povolení autora. Jak již z názvu vypovídá, takovéto systémy lze používat až po zaplacení licence, což zpravidla zaručuje vyšší kvalitu oproti open source systémům. Komerční softwary lze v některých publikacích také najít pod názvem programového vybavení uzavřeného kódu. (Burian, 2014)

5.2 Systémy s licencí open source

Na open source systémy se na rozdíl od komerčních systémů nevztahují téměř žádná licenční práva a pokud ano, jsou znatelně mírnější. Jsou tudíž pro všechny uživatele zcela zdarma. Tzn., že distribuce takového systému není již ve zkompilované podobě, uživatel má zdrojový kód plně k dispozici a může si tak provádět libovolné modifikace dle vlastního uvážení. (Burian, 2014) Velkým rizikem u open source licencí je fakt, že autor programu neručí za stabilitu a bezchybovost onoho systému. S dalším používáním těchto licencí LMS se tak mohou objevit skryté finanční či časové náklady související s opravami kódu.

5.3 Výhody LMS

Jednou z výhod LMS je bezesporu snadné ovládání učitelem, který nemusí mít technické znalosti v oblasti IT, stačí mu vykonat pouze pár jednoduchých kroků dle instrukcí v systému, což mu zajistí i více času na svou pedagogickou činnost. Z hlediska bezpečnosti je příznivou skutečností možnost zajistit kurz heslem, což umožní přístup jen vyvoleným skupinám studentů. K dalším výhodám patří možnost kontrolovat a zpětně sledovat veškeré činnosti prováděné v systému daným uživatelem, a to jak na úrovni jednotlivých kurzů, tak i v celém systému. (Zounek, Sudický, 2014)

5.4 Nevýhody LMS

Ačkoliv je LMS velmi užitečný pomocník učitelů a kvalitní zdroj výuky, nese sebou i některé nevýhody. Za nevýhody lze jednoznačně označit fakt, že daný LMS nemusí být vždy plně kompatibilní s informačním systémem dané instituce. Ač se může neustálý vývoj LMS zdát jako pozitivum, opak bývá pravdou. S každým novým upgradem systému mohou přicházet problémy, kolize či nedostatky, které již byly u předchozí verze úspěšně vyřešeny. Jak již bylo řečeno výše, LMS ulehčuje práci učitele, avšak v počáteční fázi je časová náročnost poněkud vyšší. Dále zde není možnost operativně přidávat nebo odebírat dílčí nástroje. Navíc každý další vytvořený kurz v LMS zvyšuje celkovou vytíženost serveru a může docházet až k jeho přetížení. (Zounek, Sudický, 2014)

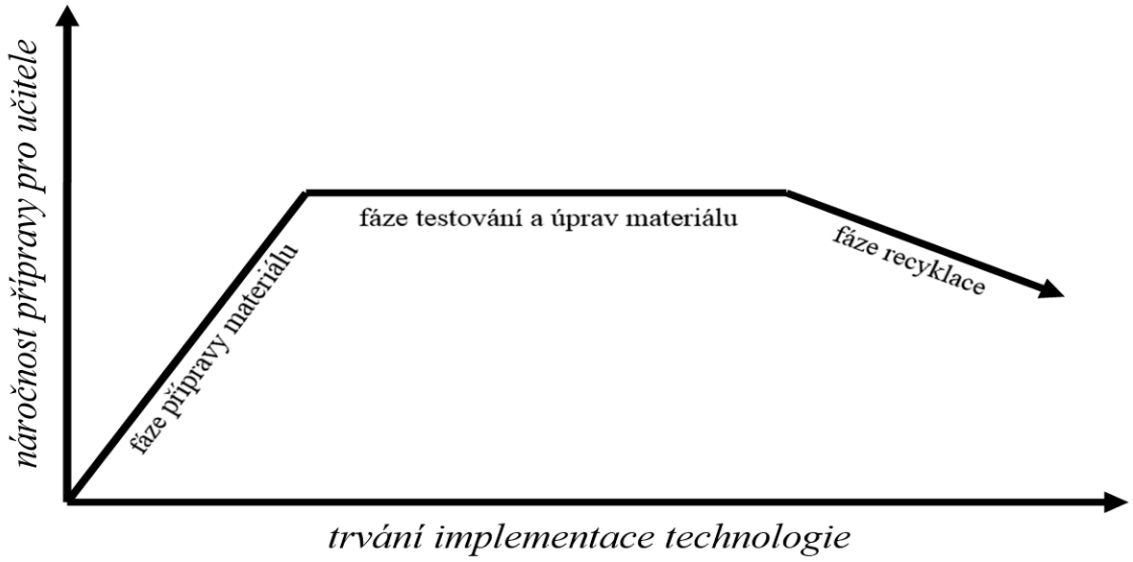


Schéma 5: Znázornění náročnosti přípravy učitele při vytváření nového studijního materiálu (Zounek, Sudický, 2012)

Schéma č. 5 popisuje náročnost přípravy učitele při vytváření nového studijního materiálu. Jak již bylo výše zmíněno, při vytváření obsahu kurzu je učitel vytížen nejvíce. Poté již při testování a upravování materiálu nedochází k vychýletní časové náročnosti. Při opakovaném použití daného kurzu již náročnost příprav klesající a učitel pouze aktualizuje studijní materiály a vylepšuje kurz.

5.5 LMS Moodle

LMS Moodle je věnována samostatná kapitola, neboť je v tomto typu systému realizována celá praktická část. Moodle je zkratka pro Modular Object-Oriented Dynamic Learning Environment, neboli volně přeloženo jako Modulární objektově orientované dynamické prostředí pro výuku. Moodle je software, který slouží k vytváření elektronických kurzů a výukových programů na internetu. Je to stále se vyvíjející projekt poskytovaný zdarma s licencí Open Source. Moodle je sice chráněn autorskými právy, avšak uživatel si ho může svobodně pořídit, modifikovat dle svých potřeb. Dále se zavazuje, že bude poskytovat tento zdroj ostatním a nebude odstraňovat či jakýmkoliv způsobem pozměňovat původní licenční údaje. Moodle je kompatibilní s jakýmkoliv počítačem s funkčním PHP a podporuje zejména databáze PostgreSQL a MySQL. (Moodle, 2006)

Obrázek 1: Logo Moodle (Moodle, 2014)

5.5.1 Komerční nadstavby a distribuce

Barešová se ve své knize také zaměřuje na komerční nadstavby a distribuce, které se orientují na vybrané skupiny zákazníků:

- *„ELIS (Enterprise Learning Intelligence Suite) – Americký partner Moodle, společnost Remote-Learner, realizovala ELIS jako sadu aplikací k řízení vzdělávání v organizaci, jejíž součástí je Moodle ve verzi 1.9.*

- *TOTARA – Projekt TOTARA je nová distribuce LMS, postavená na jádru Moodle. Projekt je orientován na firemní zákazníky. Vznikl na Novém Zélandě v konsorciu společnosti Kineo, Catalyst a Flexible Learning Network.*

- *ODALIS – Modul ODALIS vznikl v Německu pro propojení Moodle a SAP nebo jiných systémů. Systém nabízí firma sym.net.“* (Barešová, 2011)

5.5.2 Aktéři e-learningu v LMS Moodle

V e-learningu vystupuje několik aktérů, kdy každý má přiřazenou určitou roli opravňující k provádění různých činností. Je například pochopitelné, že student nemůže provádět v kurzu stejné operace jako tvůrce, manažer, či administrátor. Dále je třeba zmínit, že jeden konkrétní uživatel může i více rolí, např. v jednom kurzu být zapsán jako student a v jiném vystupovat jako učitel. V následujících podkapitolách jsou tyto jednotlivé role více specifikovány, jejich hierarchie je brána dle LMS Moodle.

5.5.2.1 Administrátor

Administrátor (nazýván též admin) je prvním a nejhlavnějším správcem ve vytvořeném výukovém systému. Má všechny práva k veškerým akcím v systému, ale sám nespravuje obsah kurzu, pokud není zároveň učitelem. (Drlík, 2013)

5.5.2.2 Manažer

Manažer může mít v e-learningu stejná práva jako admin, avšak v praxi se při nastavování kurzu doporučuje odebrat schopnost provádět zásadní změny, např. místo mazání souboru provede pouze jeho skrytí. Z důvodu vyšší opatrnosti je pro běžnou práci s kurzem vhodnější používat roli manažera, než přímo admina. (Drlík, 2013)

5.5.2.3 Tvůrce kurzu

Tvůrce kurzu je osoba vytvářející nové kurzy. Tím se stává zároveň i učitelem, následně do kurzu může přidat další učitele. Tvůrcem kurzu bývá nejčastěji vedoucí pracoviště či manažer koordinující dané kurzy. Tvůrci je možné přiřadit globálně celý systém nebo jen určitou jeho část. (Drlík, 2013)

5.5.2.4 Učitel a učitel bez práva upravovat

Učitel (nazýván též tutor, lektor, či instruktor) je ten, který vyučuje předmět a vytváří obsah kurzu. Dohlíží na odevzdané práce, známkuje testy a odpovídá na dotazy studentů. Kdežto učitel bez práva upravovat může v kurzu pouze vyučovat. Tato role se převážně přiděluje pouze odborným asistentům, kteří mají za úkol kontrolovat odevzdané práce a nesmějí modifikovat obsah kurzu. (Drlík, 2013)

5.5.2.5 Student

Jediným právem studenta je kurz používat. Nemá nárok na žádné úpravy, může se zapojovat do diskuzních fór, odevzdávat zadané úkoly, vyplňovat testy, atp. Vidí pouze tu část kurzu, kterou učitel nastavil jako viditelnou pro studenty. (Drlík, 2013)

5.5.2.6 Host

Pokud má kurz povolené prohlížení hostem, lze se do něj přihlásit pod uživatelem s rolí host. Host vstupuje do kurzů tehdy, když sám nemá vytvořený účet, ale potřebuje do určitého kurzu nahlédnout, aniž by se musel přihlašovat na zaregistrovaný účet a objevil se v seznamu uživatelů kurzu. (Drlík, 2013)

6 Tvorba e-learningového kurzu pro studenty střední školy dle modelu ADDIE

Cílem praktické části diplomové práce je v prostředí LMS Moodle vytvořit e-learningový kurz Algoritmizace a programování v jazyce C pro studenty střední školy. Tvorba zmíněného kurzu bude probíhat dle modelu ADDIE.

Programovací jazyk C byl zvolen z důvodu, že je vyučován na vybrané střední škole a dále ho autor absolvoval během vysokoškolského studia. Kurz je realizován v LMS Moodle, neboť Česká zemědělská univerzita umožňuje svým studentům vytvářet vlastní kurzy bez finančního zatížení.

6.1 Analýza

Prvním krokem analýzy je určení cílové skupiny studentů. Jak již bylo v zadání diplomové práce zmíněno, bude se jednat o studenty střední školy. Konkrétně o studenty, kteří mají v osnovách výuky předmět zabývající se programovacím jazyem C. Tento kurz by absolvovali zároveň s prezenční výukou, čímž by se mělo dosáhnout vyšší efektivity vzdělávání.

Hlavní cíl tohoto kurzu spočívá především v rozvinutí logického myšlení studentů prostřednictvím tvorby algoritmů a jejich následného přepisování do zdrojového kódu v programovacím jazyce C. Autorovým záměrem není to, aby se ze studentů po absolvování tohoto kurzu stali programátoři, ale aby zejména pochopili, jakým způsobem vlastně programy fungují, co obnáší jejich tvorba, rozuměli strukturogramům a vývojovým diagramům, ovládali základní syntaxi programovacího jazyka C, uměli se orientovat ve zdrojovém kódu a odhalovat nejen syntaktické, ale i sémantické chyby. S výše zmíněnými znalostmi by studenti měli být schopni vytvářet jednoduché programy.

Autor dále rozepisuje vstupní požadavky pro vstup do kurzu, mezi které patří především logické myšlení, poněvadž bez něho nelze tvořit kvalitní programy. Dále pak matematika, neboť při složitějším programování lze narazit na matematické problémy, které by měl být každý programátor schopen vyřešit. Jelikož většina technických materiálů a syntaxe programovacích jazyků vychází z anglického jazyka, je nutné ovládat anglický

jazyk alespoň na začátečnické úrovni, avšak pokud by se dotyčný chtěl věnovat programování více, musí pracovat i na své angličtině. V neposlední řadě se od studentů logicky předpokládá uživatelská znalost práce na PC.

Následující odstavec se věnuje vývojovému prostředí, ve kterém budou vznikat první studentské programy. Pro volbu tohoto prostředí mohla být rovněž aplikována vícekriteriální analýza, avšak nároky na volbu daného prostředí nejsou nikterak vysoké a jediným rozhodujícím kritériem by byla cena. Studentům v tomto kurzu postačí pouze textový editor a compiler, který převádí zdrojový kód na spustitelný program. Autor vybral čtyři vývojové prostředí, se kterými má zkušenosti a jsou pro danou výuku naprosto dostačující. Uvedená doporučená vývojová prostředí jsou seřazena sestupně dle subjektivních preferencí autora. Jedná se o MS Visual Studio, což je bohužel distribuováno pod placenou licencí. Ostatní vývojová prostředí jsou již zdarma a jsou to CodeBlocks, Borland C++ a Dew-C++. Studentům je rovněž umožněna samostatná volba, kdy si mohou zvolit tvorbu v jakémkoliv jiném funkčním vývojovém prostředí, se kterým mají sami zkušenosti.

E-learningový kurz by měl studenty nějakým způsobem motivovat. V kurzu Algoritmizace a programovací jazyk C je hlavním stimulem pro studenty zejména skutečnost, že po splnění kurzu budou schopni samostatně vytvářet jednodušší programy. Dále tento e-learning pomůže výrazným způsobem s přípravou ke zkoušce z odborného předmětu vyučovaného na dané střední škole, který se zabývá programováním. Aby studentům tento kurz nebyl lhostejný, musí plnit dílčí autotesty a závěrečný test, za který budou známkováni.

6.2 Návrh

Druhou fází v tvorbě kurzu Algoritmizace a programování jazyka C je návrh tohoto kurzu. V této fázi je naplánován harmonogram přípravy kurzu, který je detailněji rozepsán v tabulce č. 4, přičemž lze vidět, že v každém měsíci byla provedena určitá činnost spočívající od prvotní analýzy, až po samotné testování studentů.

Tabulka 4: Časový harmonogram přípravy kurzu

Časový harmonogram	Realizovaná činnost
listopad 2014	detailní analýza kurzu (určení cílové skupiny, cíle, vstupních požadavků, vývojového prostředí, motivace)
prosinec 2014	seznámení s LMS Moodle (nastavení kurzu, vytvoření struktury kurzu)
leden 2015	tvorba studijních materiálů a databáze otázek
únor 2015	tvorba testů, zkušební provoz, eliminace nedostatků
březen 2015	představení kurzu studentům střední školy a jejich následné otestování

V této fázi je důležité vyčíslit finanční náročnost celé realizace kurzu. Vzhledem k tomu, že Česká zemědělská univerzita v Praze umožňuje svým studentům tvorbu e-learningových kurzů v LMS Moodle zdarma, které lze dohledat na adrese https://projekty.czu.cz/, nepředstavoval vytvořený kurzu Algoritmizace a programovací jazyk C pro autora žádné finanční náklady.

Každý e-learningový kurz by měl mít taktéž správně zvolenou intenzitu výuky. Časové rozvržení tohoto kurzu je nastavené dle souběžné prezenční výuky na střední škole. Tzn., v té době, kdy vyučující probírá se studenty dané téma, je studentům zpřístupněna lekce týkající se dané látky.

Kurz Algoritmizace a programovací jazyk C je navržen do dvanácti lekcí, přičemž první lekce je úvodní. Algoritmizaci se věnuje lekce č. 2 a 3., od 4. lekce se připojuje tématika programování. Od páté lekce by měl již být student schopen vytvářet jednoduché programy, přičemž náročnost programování se stupňuje s každou další lekcí. Od této lekce jsou do jednotlivých kapitol přidávány taktéž zdrojové kódy, na kterých je demonstrováno využití probírané látky. Ve 4., 8. a 11. výukové lekci je navíc pro studenty připraven autotest. Desetiminutový autotest se skládá vždy z deseti otázek a látka je zaměřena na probraná témata z předešlých lekcí. Autotesty je třeba splnit na 60%, přičemž počet pokusů není pro studenty omezený. Neomezený počet pokusů je zvolen z důvodu usnadnění přípravy studentů na závěrečný test. Dvanáctá kapitola již není výuková, avšak obsahuje velký závěrečný test z veškeré vysvětlované problematiky. Tento test

se již skládá z třiceti otázek a studenti na jeho vypracování budou mít třicet minut. Závěrečný test má otázky stejného charakteru jako autotest, avšak žádná otázka z autotestu není použita v závěrečném testu a často se zde objevují tzv. chytáky. Na závěrečný test mají studenti oproti autotestům pouze jeden řádný pokus a jeden opravný a hranice úspěšnosti je stanovena na 70 %.

6.3 Vývoj

Tato fáze se projevila jako velmi časově náročná fáze. Byly vytvořeny veškeré studijní materiály, otázky, testy a domácí úkoly. Tato kapitola popisuje všech 12 částí kurzu, přičemž první a poslední nejsou výukového charakteru. První kapitola je pouze úvodní a seznamuje studenty se základními informacemi o kurzu. Poslední obsahuje závěrečný test a zpětnou vazbu autorovi pro zkvalitnění kurzu samotného. Každá výuková prezentace má na svém začátku logickou hádanku a na konci je odhalena správná odpověď. Neboť je kurz vytvořen jako doplňkový kurz k prezenční výuce, jsou v každé lekci vysvětleny pouze ty nejpodstatnější pojmy či teoretické principy, které studenti budou skutečně potřebovat při tvorbě zadaných programů. Autor nemá v úmyslu zahlcovat studenty rozsáhlými a přebytečnými znalostmi, ale naopak se snaží maximálně rozvíjet jejich logické myšlení tím, že budou muset aplikovat své znalosti praktickým způsobem při vytváření zadaných programů.

6.3.1 Lekce č. 1: Úvod

V první lekci tohoto kurzu je pouze jedna prezentace, která vítá studenty v kurzu Algoritmizace a programování v jazyce C a seznamuje čtenáře se základními informacemi o kurzu. Druhý slajd se týká autora kurzu a jsou zde uvedeny kontaktní údaje (email a telefon) pro případné dotazy studentů. Následující části seznamují studenty s cílem, podmínkami pro vstup do kurzu a doporučenými vývojovými prostředími. Všechny zmíněné části jsou podrobně charakterizovány v kapitole 6. 1.

Další část úvodní prezentace se zabývá podmínkami pro absolvování kurzu. Student bude povinen zvládat průběžné autotesty (minimální úspěšnost je stanovena na 60% s neomezeným počtem pokusů. Na konci kurzu je třeba splnit závěrečný test,

u kterého je již minimální hranice úspěšnosti stanovena na 70% a student má možnost test opakovat pouze jednou.

V posledním slajdu první lekce se student seznamuje s celkovou osnovou kurzu, čímž by se mu měla ucelit představa o náplni kurzu a znalostech, které by měl absolvováním kurzu nabýt. Výuka kurzu bude probíhat ve sledu následujících lekcí:

1. Úvod
2. Základní pojmy
3. Způsoby zaznamenávání algoritmů
4. Programovací jazyk C
5. Hello world
6. Proměnné, jejich typy a práce s nimi
7. Podmínky (if, switch)
8. Cykly (while, do while, for)
9. Pole a práce s nimi
10. Tvorba vlastních funkcí
11. Práce se soubory
12. Závěrečný test

Obrázek 2: Vybrané slajdy úvodní lekce

Na obrázku č. 2 lze vidět vybrané dva slajdy z úvodní prezentace, přičemž celou prezentaci si lze prohlédnout v příloze č. 1.

6.3.2 Lekce č. 2: Základní pojmy

Druhá lekce je čistě teoretického charakteru a má za úkol seznámit studenty se základními pojmy. Tyto pojmy je nutné znát, neboť bez jejich znalosti nelze pochopit výklad v dalších lekcí.

Nejprve je studentům vysvětlen pojem algoritmus. Autor po jeho definování doplňuje, že algoritmus nemusí vždy souviset s programováním, ale že i recept v obyčejné kuchařce lze považovat svým způsobem za algoritmus. Následně se studenti dozvídají, že právě program je přespaný algoritmus do konkrétního programovacího jazyka.

Na dalším slajdu je popsán životní cyklus programu a všech jeho pět částí skládající se z analýzy problému, stanovení podmínek, za kterých má program fungovat, sestavení algoritmu, následně z něj sestavení programu a v konečné fázi samotné otestování a ladění programu.

Každý algoritmus musí splňovat osm kritérií, proto autor na následujících dvou slajdech svými slovy vysvětluje jednotlivé vlastnosti algoritmů:

- **Konečnost** – každý algoritmus musí mít konečný počet kroků, jinými slovy, nesmí být nekonečný.
- **Správnost** – výsledek vydaný algoritmem musí být správný.

- **Obecnost** – algoritmus neřeší jeden konkrétní problém (např.: 2*9), ale obecnou třídu obdobných problémů (např.: součin dvou čísel).
- **Rezultativnost** – po zadání vstupních dat vždy vrátí výsledek (může to být i chybové hlášení).
- **Jednoznačnost** – v každé situaci musí být naprosto zřejmé, co a jak se má provést a jaké kroky budou následovat.
- **Opakovatelnost** – při stejných vstupních hodnotách musí vyjít vždy shodný výsledek.
- **Srozumitelnost** – musí být srozumitelný i pro uživatele, který jej nevytvořili.

Prezentace pokračuje ve vymezení tří základních řídících struktur algoritmu, jimiž jsou sekvence, která představuje posloupnost prováděných operací, selekce značící větvení algoritmu/programu a iterace, která provádí opakování nějaké části algoritmu (u programátorů je na místo iterace více používaný pojem cyklus).

Studenti se dále v druhé lekci seznamují s problematikou syntaxe a sémantiky. Po vysvětlení těchto pojmů je navíc nastíněno, jak se projevuje skutečnost, kdy programátor udělá syntaktickou či sémantickou chybu a jak lze pomocí kompilátoru syntaktickou chybu vyhledat a opravit. Naopak v případě sémantické chyby musí programátor projít celý kód a chybu nalézt, což u rozsáhlých projektů může být skutečně velký problém. Na konci prezentace je uveden zdroj, z kterého autor čerpal.

Obrázek 3: Vybrané slajdy lekce č. 2

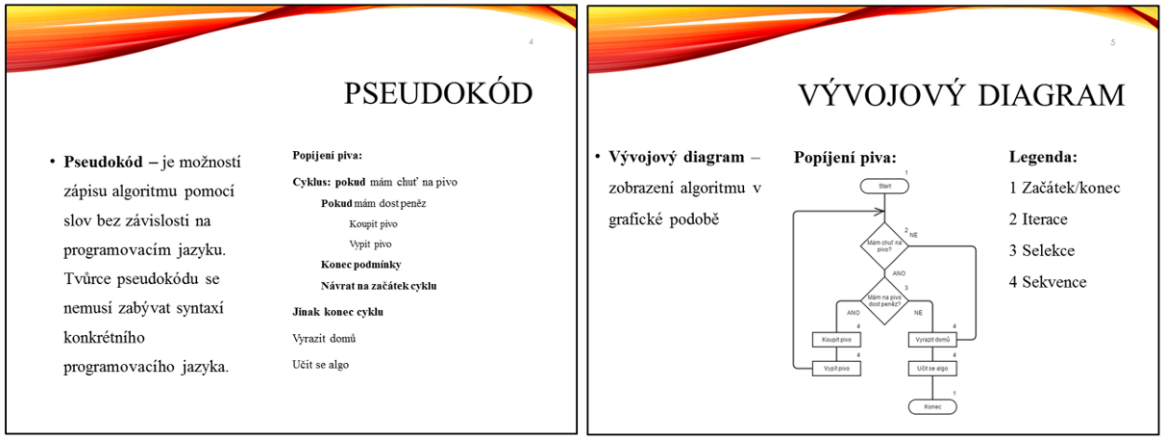
Na obrázku č. 3 lze vidět vybrané dva slajdy z úvodní prezentace, přičemž celou prezentaci si lze prohlédnout v příloze č. 1.

6.3.3 Lekce č. 3: Způsoby zaznamenávání algoritmů

Lekce č. 3 se zabývá způsoby, jak zaznamenat navržený algoritmus. Lekce je poměrně krátká. Jakmile ji student jednou pochopí, neměla by mu činit v budoucnu problémy.

Student se nejprve dozvídá, jakými způsoby lze algoritmy prezentovat, tj. popisem, graficky a programovacím jazykem. Každý z těchto způsobů je na následujících slajdech dále popsán a vysvětlen na stejném jednoduchém algoritmu. Pro zaujetí studentů použil autor vzorový příklad, který řeší během návštěvy v hospodě, zda má host chuť na pivo. Pokud ano, následuje položení otázky, zda má dost peněz, pokud je odpověď znovu kladná, nastává fáze koupení a zkonzumování piva. Poté se vrací na začátek a znovu si pokládá otázku, zda má chuť na pivo. Tímto je demonstrován cyklus algoritmu, v jehož těle je navíc vložena další podmínka. V případě, že je na nějakou z otázek odpovězeno záporně, cyklus se zastaví a host vyráží domů učit se algoritmizaci.

Výše zmíněný příklad je demonstrován pseudokódem, který zapisuje algoritmus slovně a přitom je nezávislý na programovacím jazyku a grafických značkách. Dále vývojovým diagramem, který zobrazuje algoritmus v grafické podobě pomocí základních řídících struktur (sekvence, selekce, iterace) a speciální značky pro vyjádření začátku a konce algoritmu. Následuje ukázka přikladu prezentována prostřednictvím strukturogramu, což je zobrazení algoritmu v tabulkové podobě, rovněž za použití základních řídících struktur, avšak již bez dalších speciálních značek.



Obrázek 4: Vybrané slajdy lekce č. 3

Na obrázku č. 4 lze vidět vybrané dva slajdy z úvodní prezentace, přičemž celou prezentaci si lze prohlédnout v příloze č. 1.

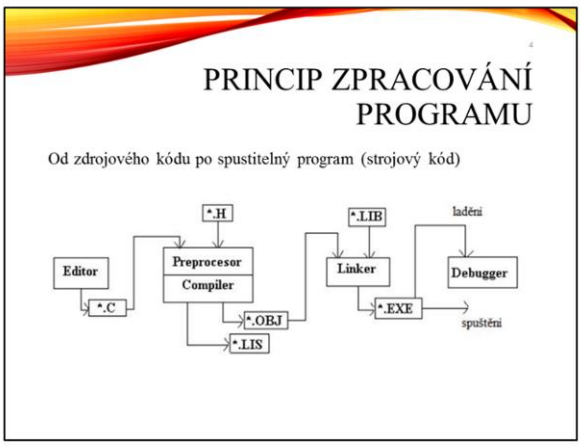
Třetí lekce neobsahuje pouze prezentaci, ale také položku pro odevzdávání domácího úkolu. Studenti mají za úkol vytvořit vývojový diagram či strukturogram na volitelnou činnost z běžného života (startování auta, čištění zubů, vaření čaje, apod.). Úkol nemusí být rozsáhlý, ale musí obsahovat sekvenci, selekci i iteraci. Po domluvě s vyučujícím na střední průmyslové škole, na které byl kurz prezentován, byla položka pro odevzdání domácího úkolu skryta, neboť měli studenti v dané době několik vlastních úkolů a nebylo třeba je více zatěžovat.

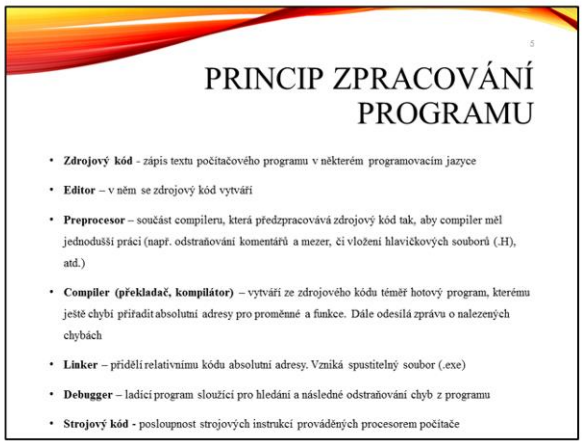
6.3.4 Lekce č. 4: Programovací jazyk C

Ve čtvrté lekci již autor opouští část algoritmickou a začíná studenty seznamovat se základy programovacího jazyka C. V této lekci studenti prozatím nebudou sami vytvářet žádné programy, ale budou pouze seznámeni s teoretickými východisky o dané problematice.

Na začátku prezentace jsou studentům vysvětleny základní specifikace programovacího jazyka C. Student se dozvídá, že tento konkrétní programovací jazyk C vznikl v 70. letech minulého století, že se jedná o:

- **nízkoúrovňový jazyk** – představuje velmi nízkou úroveň abstrakce (tzn., že kód programovacího jazyka je velmi podobný reálným strojovým instrukcím),
- **kompilovaný jazyk** – před spuštěním musí být zdrojový kód nejprve zkompilován do strojového kódu, který je již spustitelný,
- **strukturovaný a procedurální jazyk** – při tvorbě algoritmu či programu se postupuje shora dolů, používají se jenom tři základní řídicí struktury (sekvence, selekce, iterace),
- **nespecializovaný jazyk** – není specializovaný pouze na jednu oblast používání,
- **efektivní jazyk** – velmi vysoká efektivita přeloženého kódu (téměř srovnatelný s Assemblerem),
- **case sensitive jazyk –** rozlišuje velká a malá písmena (promenna, Promenna, PROMENNA – pokaždé se jedná o jinou proměnnou).



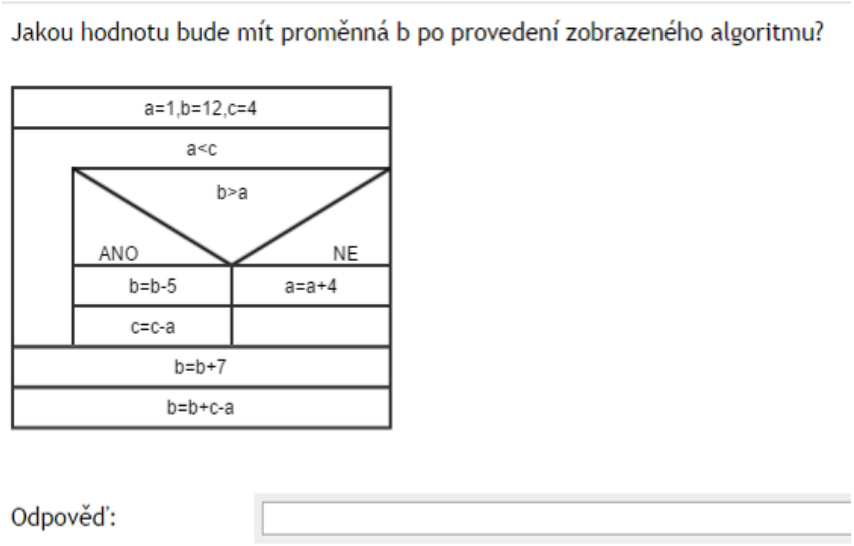


Obrázek 5: Vybrané slajdy lekce č. 4

Následně je studentům vysvětlen princip zpracování programu, tj. od zdrojového kódu po spustitelný program. Pro lepší pochopení je tento princip vysvětlen pomocí schématu převzatého z odborné literatury, který lze vidět na obrázku č. 5. Na následujícím slajdu jsou charakterizovány jednotlivé funkce všech částí zobrazených na schématu, jimiž

jsou editor, preprocesor, compiler, linker, debugger. Tento slajd je možné vidět taktéž na obrázku č. 5. Celá prezentace je k nahlédnutí v příloze č. 1.

V této lekci je pro studenty navíc připraven první autotest, který se skládá z deseti otázek. Studenti mají na vypracování deset minut. Autotest obsahuje otázky z výukových lekcí 2 – 4. Test byl koncipován tak, že obsahoval z každé lekce tři otázky a jednu matematicko-logickou hádanku. Celou ukázku autotestu č. 1 lze vidět na přiloženém CD/Testy/autotest1.



Obrázek 6: Vybraná otázka z prvního autotestu

Obrázek č. 6 představuje jednu z deseti otázek, které se v testu vyskytují. Tato konrktění otázka je z lekce č. 3 a jedná se o průchod strukturogramem, přičemž studenti musejí určit jaká hodnota bude uložena v proměnné b.

6.3.5 Lekce č. 5: Hello world

Od páté lekce kurz začíná být praktického charakteru a studenti začínají aplikovat své teoretické znalosti do praxe prostřednictvím programování. Lekce je pojmenována, Hello world, což je nejtypičtější název prvního programu, kteří programátoři vytvářejí. Znamená to, že již mají nainstalované a správně nastavené vývojové prostředí a zvládnou vytvořit program, který má jediný účel. Tím je výpis textového řetězce, který obsahuje textový řetězec: Hello world.

Pro programování je klíčové mít správně nastavené vývojové prostředí. Proto autor ihned zpočátku prezentace uvádí návod, jak se ve vývojovém prostředí CodeBlocks zorientovat. Pokud se student držel všech pokynů pro nastavení, může začít se samotným programováním.

První krok při vytváření programu spočívá v připojení hlavičkových souborů, které zpřístupňují konkrétní funkce. Autor v prezentaci uvádí, že pro většinu studentských programů bude stačit pouze jediná knihovna s názvem stdio.h (neboli standardní input/output). Poté studentům ukazuje způsob, jak tuto knihovnu připojit, tj. pomocí #include <stdio.h>. Po připojení všech potřebných knihoven se může začít psát hlavní funkce programu.

Po definování hlavičkových souborů se začíná tvořit hlavní funkce programu. Studenti se na tomto následujícím slajdu dozvídají, jak vypadá nejjednodušší hlavní funkce, a že má vždy nastaven datový typ (v konkrétním případě typ int – zkratka pro integer neboli celé číslo). Poté je zmíněno, že hlavní funkce má vždy název main a po spuštění programu je volána jako první. Každá funkce může mít vstupní parametry, které jsou obsaženy v závorkách za názvem funkce, v tomto případě funkce žádné tyto vstupní parametry nemá. Dále je zde uvedeno, že za každou funkcí se ve složených závorkách nachází tělo funkce. V ukázkovém příkladu je tělo funkce prázdné, kromě příkazu return, což je příkaz, který vrací výslednou hodnotu funkce. Jelikož je funkce datového typu int, musí funkce vracet celé číslo (v uvedeném případě nulu).

Aby program něco vykonával, musí se do těla hlavní funkce přidat další příklady. Jelikož je zadáno pouze vypisovat text, postačí jednoduchá funkce sloužící pro výpis na konzoly. Tato funkce se nazývá printf a printf (“Libovolný text“); je její nejjednodušší zápis. Jako každá funkce, i tato funkce vrací hodnotu, která je rovna počtu vypsaných znaků do konzole.

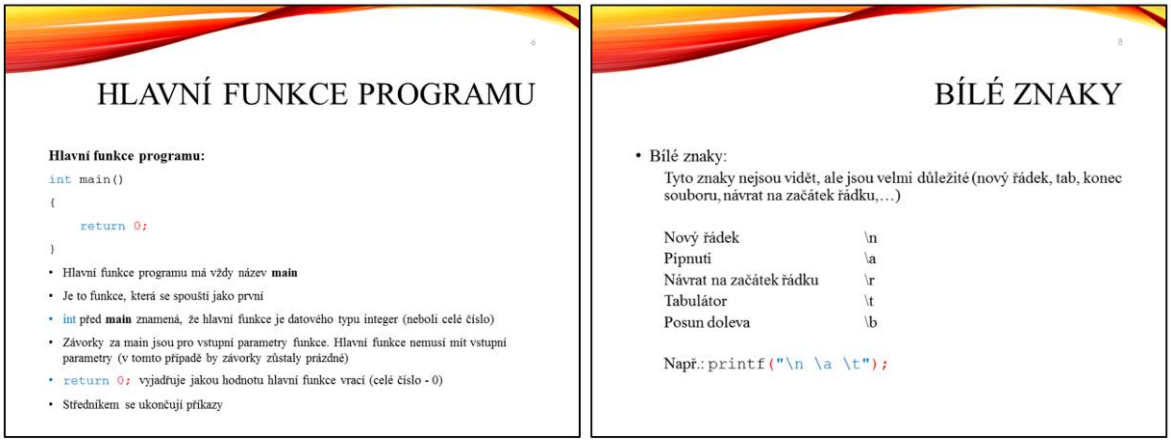
Na následujícím slajdu jsou studentům vysvětleny bílé znaky, což jsou znaky, které nejsou vidět, ale jsou velmi důležité. Patří mezi ně např. nový řádek (\n), pípnutí (\a), návrat na začátek řádku (\r), tabulátor (\t), či posun doleva (\b) a umisťují se přímo do textového řetězce, který se má vypsat.

Pro lehčí orientaci v programu a pro zapsání myšleny, či důvodu umístění nějakého příkladu lze využívat komentářů. Tyto komentáře nemají žádný vliv na program samotný, slouží pouze pro orientaci v kódu. Komentáře jsou dvojího typu, a to jednořádkové, které se zapisují dvěma lomítky, nebo víceřádkové, které začínají lomítkem a hvězdičkou a končí hvězdičkou a lomítkem.

Ke konci prezentace je studentům ukázán, jak má vypadat celý zdrojový kód programu, který vypisuje na konzoli Hello world:

```
#include <stdio.h>
int main()
{
   printf("Hello world!\n");
   return 0;
}
```

V komentářích jsou detailně vysvětleny všechny části kódu. Tento zdrojový kód lze taktéž nalézt v této páté lekci přímo pod prezentací.



Obrázek 7: Vybrané slajdy lekce č. 5

Na obrázku č. 7 lze vidět vybrané dva slajdy z úvodní prezentace, přičemž celou prezentaci si lze prohlédnout v příloze č. 1.

6.3.6 Lekce č. 6: Proměnné, jejich typy a práce s nimi

Aby program mohl dělat něco více, než jen vypisovat text na konzoly, musí při své práci používat proměnné. V lekci č. 6 je tedy studentům vysvětleno, jakým způsobem

proměnné vytvářet, jak proměnné načítat, jaké operace lze s nimi provádět a jak je za pomoci již známé funkce printf vypisovat.

Autor uvádí ve své prezentaci pouze tři datové typy, které považuje za nezbytné pro nadcházející programy a ostatním existujícím datovým typům se nevěnuje. Jsou to celá čísla (značící se int), reálná čísla (značící se float) a znaky (značí se char). Na daném slajdu je autor nejen vyjmenuje, ale ukazuje i zápis, jakým způsobem se vytvářejí (např. vytvoření celé číselné proměnné vypadá takto: int počet_lidi;)

Na následujícím slajdu jsou definovány operátory, které umožňují matematické operace s číselnými proměnnými (+, -, *, /, %), přičemž první čtyři operátory jsou tak zažité, že je netřeba dále vysvětlovat a poslední % znamená tzv. dělení modulo, neboli zbytek po celočíselném dělení).

Autor se dále v prezentaci věnuje všem možným přístupným operací s proměnnými, jako je např. naplnění proměnné, přiřazení hodnoty do proměnné, matematické operace, speciální zkrácené zápisy, inkrementace (tj. zvýšení proměnné o jedna) a dekrementace (tj. snížení proměnné o jedna). S proměnnými typu znak (char) lze provádět také matematické operace, neboť je tento znak v paměti počítače reprezentován hodnotou znaku uvedené v ASCII tabulce (viz příloha č. 2).

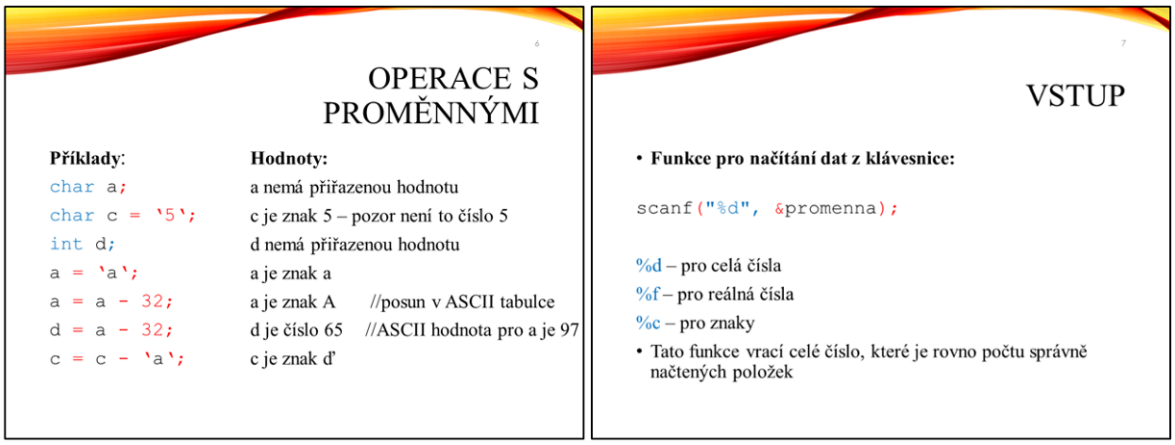
Studentům je dále vysvětleno, že programy nemohou mít pouze pevně definované proměnné, ale některé jejich hodnoty se musí načítat z klávesnice. Za tímto účelem uvádí funkci scanf, která vrací takové celé číslo, které je rovno počtu správně načtených položek. Správný zápis funkce scanf vypadá takto: scanf(“%d“, &promenna);, přičemž %d značí načítání celého čísla. Pokud by chtěl programátor načítat znaky, použije %c, pokud reálná čísla, použije %f.

Operace s proměnnými jsou studentům demonstrovány na sestavení programu, který načte 3 reálná čísla a vypíše jejich aritmetický průměr. Zdrojový kód zadaného programu vypadá takto:

```
#include <stdio.h>

int main()
{
```

```
    float a;
    float b;
    float c;
    float prumer;

    printf("Zadejte 3 realna cisla\n");

    scanf("%f",&a);
    scanf("%f",&b);
    scanf("%f",&c);

    prumer = (a+b+c)/3;
    printf("Aritmeticky prumer zadanych 3 cisel je :%f",prumer);
    return 0;
}
```

Tento zdrojový kód lze taktéž nalézt pod uvedenou výukovou prezentací.



Obrázek 8: Vybrané slajdy lekce č. 6

Na obrázku č. 8 lze vidět vybrané dva slajdy z úvodní prezentace, přičemž celou prezentaci si lze prohlédnout v příloze č. 1.

6.3.7 Lekce č. 7: Podmínky (if, switch)

Sedmá lekce je zaměřena na druhou základní řídící strukturu, tj. selekci, neboli větvení programu. Programátoři, nežli selekce, používají spíše pojem podmínka. Tato prezentace se věnuje tvorbě logického výrazu a dvěma základním podmínkám, což jsou if a switch.

Nejprve je studentům ukázáno, co jsou logické výrazy a jak mohou být testovány. Proměnné mohou být testovány na rovnost (==), nerovnost (! =), menší (<), větší (>), větší nebo rovno (>=), menší nebo rovno (<=), jednotlivé výrazy lze negovat značkou !. Logické výrazy lze spojovat do jednoho prostřednictvím značek &&, značící si logický součin a značek ||, značící logický součet.

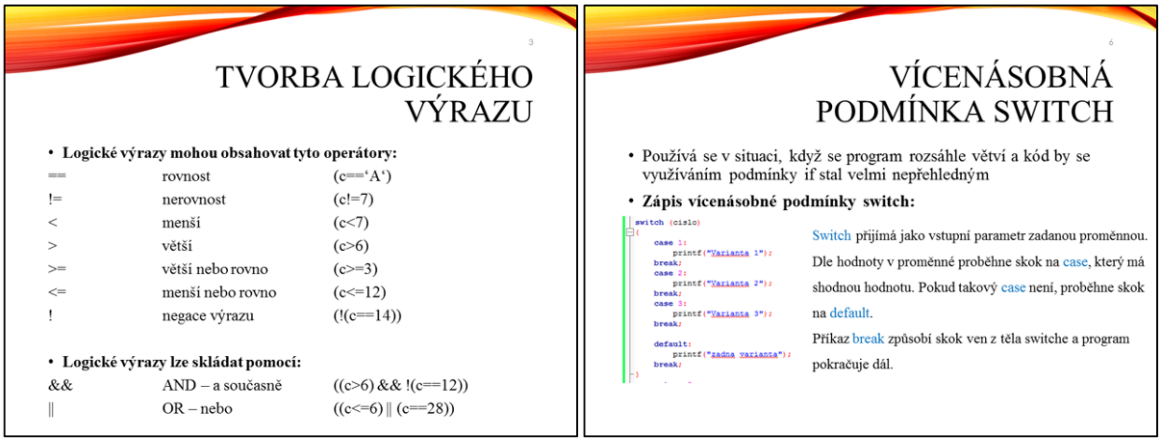
Na následujícím slajdu již autor vysvětluje v jakých situacích je potřeba využít větvení programu, tj. když je nutné, aby se program choval rozdílně v závislosti na vyhodnocené podmínce. Následně představuje podmínku if a demonstruje ji na části programu, který porovná dvě čísla a to větší následně vypíše. Pro lepší pochopení uvádí další příklad programu, který má za úkol rozeznat, zda je číslo sudé či liché. Algoritmus je zde zaznamenán graficky pomocí vývojového diagramu a zároveň ve formě zdrojového kódu s využitím podmínky if.

Prezentace pokračuje definováním vícenásobné podmínky switch, která se používá v situaci, kdy se program rozsáhle větví a kód by se využíváním podmínky if stal velmi nepřehledným. Na slajdu je proveden zápis vícenásobné podmínky s rozsáhlejším komentářem, který se zaměřuje zejména na klíčová slova switch, case, default a break. Switch přijímá jako parametr jednu zadanou proměnnou. Dle této hodnoty probíhá skok na case se shodnou hodnotou. Pokud takovýto case není, proběhne skok na default. Příkaz break způsobí skok ven z těla switche a program pokračuje dál.

V sedmé lekci se pod výukovou prezentací rovněž nachází dva programy. První demonstruje využití podmínky if, který je uveden níže a druhý vícenásobné podmínky switch, který je uveden na přiloženém CD/Zdrojové kódy/L7_switch.

```
#include <stdio.h>
int main()
{
   int cislo;

   printf("Zadejte cislo\n");
   scanf("%d",&cislo);

   if(cislo%2==0)
   {
      printf("\nZadane cislo je sude\n");
   }
```

```
    else
    {
        printf("\nZadane cislo je liche\n");
    }
    return 0;
}
```



Obrázek 9: Vybrané slajdy lekce č. 7

Na obrázku č. 9 lze vidět vybrané dva slajdy z úvodní prezentace, přičemž celou prezentaci si lze prohlédnout v příloze č. 1.

6.3.8 Lekce č. 8: Cykly (while, do while, for)

Osmá lekce je zaměřena na poslední řídící strukturu algoritmu, tj. iterace, někdy též nazývanou jako cykly. V prezentaci jsou uvedeny tři druhy cyklů, jež budou studenti ve svých programech potřebovat. Jedná se o cykly while, do while a for.

Prvním vysvětlovaným cyklem je cyklus while. V prezentaci autor uvádí, že pro vykonání cyklu se nejprve testuje logický výraz v podmínce. Pokud je tato podmínka splněna, vykoná se celé tělo cyklu a poté se program vrátí zpět na vyhodnocování podmínky. Takto se cyklus neustále opakuje, až do okamžiku, kdy je logický výraz v podmínce nesplněn. Na tomto slajdu je taktéž vytvořen zápis cyklu while, který je aplikován na jednoduchý příklad, kdy se tento cyklus může použít. Zápis je doplněn o postačující komentář.

Následující slajd je věnován cyklu do while. Studentům je vysvětleno, že jediný rozdíl od cyklu while je ten, že se nejdříve provede tělo cyklu a teprve poté se vyhodnocuje

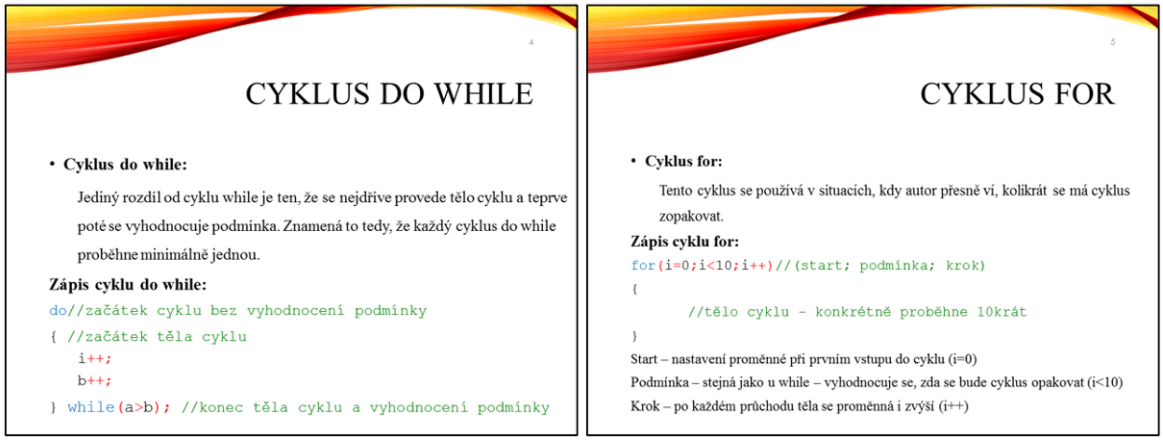
logický výraz v podmínce. To znamená, že každý cyklus do while proběhne minimálně jednou. Zde je rovněž pro snazší pochopení uveden zápis cyklu do while na jednoduchém příkladu s příslušnými komentáři.

Poslední z vysvětlovaných cyklů je cyklus for. Autor uvádí, že cyklus for se od předchozích cyklů liší v tom, že tento cyklus se používá v situacích, kdy autor přesně ví, kolikrát se má cyklus zopakovat. V závorkách za klíčovým slovem for není pouze vložen podmíněný výraz, jakož je tomu u cyklů whlie a do while, ale také je zde výraz pro nastavení výchozí proměnné a následně se za středníkem nachází logický výraz pro vyhodnocení podmínky a za posledním středníkem je tzv. krok, tj. příkaz, který se vykoná při každém průchodu cyklem. Opět je zde pro studenty uveden názorný příklad zápisu cyklu for.

Rovněž v osmé lekci, jsou pod prezentací uvedeny zvlášť tři zdrojové kódy programů, kde každý demonstruje využití jednoho zmíněného cyklu. Takto vypadá zdrojový kód, který disponuje cyklem while:

```
#include <stdio.h>

int main ()
{
  int i;
  int vysledek=0;
  printf("Zadavejte cisla ke scitani\nZadavani ukoncite znakem 0\n");

  scanf("%d",&i);
  while(i!=0)
  {
     scanf("%d",&i);
     vysledek=vysledek+i;
  }

  printf("Vysledek je: %d",vysledek);
  return 0;
}
```

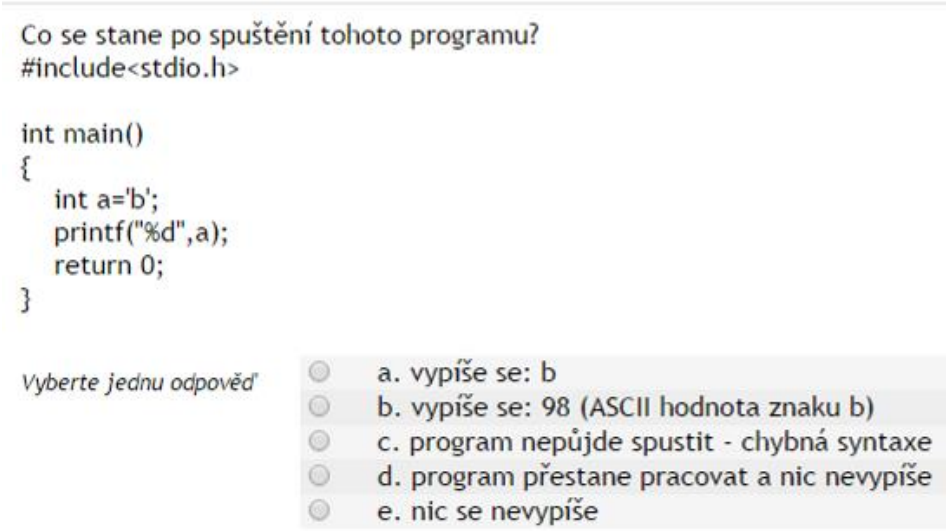
Zbylé dva zdrojové kódy týkající se využití cyklu do while a cyklu for, lze nalézt na přiloženém CD/Zdrojové kódy/L8_do_while a */L8_for.



Obrázek 10: Vybrané slajdy lekce č. 8

Na obrázku č. 10 lze vidět vybrané dva slajdy z úvodní prezentace, přičemž celou prezentaci si lze prohlédnout v příloze č. 1.

V osmé lekci je pro studenty připraven druhý autotest, který má stejné podmínky pro splnění, jako první autotest. Tento autotest obsahuje látku z 5 – 8 lekce. Celý autotest lze vidět v přiloženém CD/Testy/autotest2.



Obrázek 11: Vybraná otázka z druhé autotestu

Na obrázku č. 11 je vybraná otázka z druhého autotestu a studenti by v ní měli ze zdrojového kódu odvodit, jak se bude po spuštění daný program chovat. Správnou odpovědí je zde možnost b), tj. vypíše se 98 (ASCII hodnota znaku b).

6.3.9 Lekce č. 9: Pole a práce s nimi

Devátá lekce je zaměřena na proměnné typu pole. Autor v této prezentaci nejprve pojem pole vysvětluje, poté uvádí vybrané operace s nimi a zadává studentům program k vyřešení na toto téma.

Autor nejprve studentům objasňuje, v jakých situacích lze pole používat, tj. v případech, kdy je potřeba hromadně definovat nějaký počet proměnných určitého typu (nejčastější využití polí lze zaznamenat při tvorbě textových řetězců). Dále je na tomto slajdu definován syntakticky správný způsob zápisu pro vytvoření pole. Autor doplňuje, že definice pole je shodná jako u klasických proměnných, avšak parametr uvedený v hranatých závorkách udává, kolik znaků či čísel bude pole obsahovat. Pro určení položky v poli se používají indexy. Studenti by také měli mít na paměti, že pole začíná vždy indexem 0.

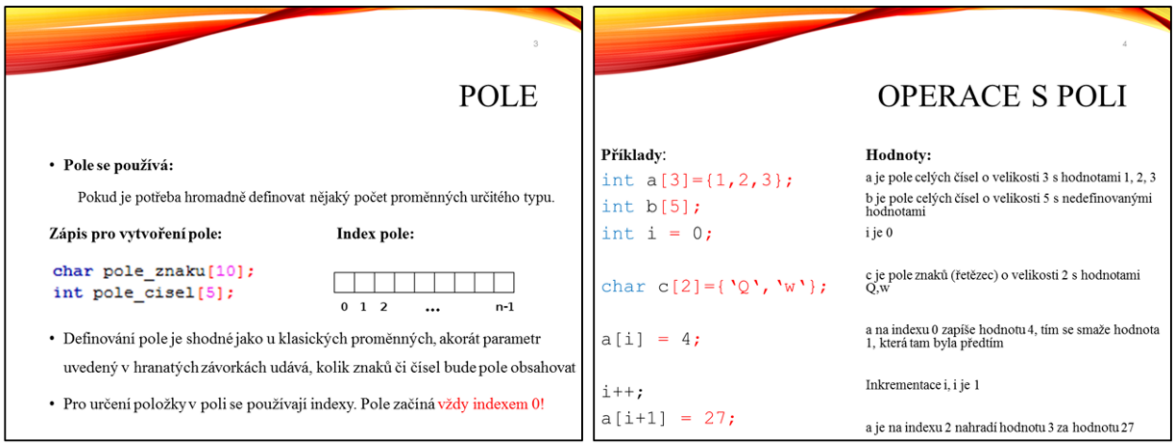
Na následujícím slajdu jsou uvedeny možné operace s poli. Studenti se mohou například dozvědět, jakým způsobem vytvořit předdefinované pole, jak vytvořit prázdné pole o velikosti pěti proměnných, či jakým způsobem lze přiřazovat do určitých proměnných hodnotu pole například na x-tým indexu.

Ke konci prezentace je studentům zadán program, který by se měli samostatně pokusit vyřešit. Program tkví v tom, že bude načítat čísla tak dlouho, dokud se nezadá nula. Poté vypíše pět nejvyšších zadaných čísel v sestupném pořadí. Na následujícím slajdu je pro kontrolu, či inspiraci studentů, kteří vytvořit program nezvládli, uvedena jedna z variant vyřešeného programu:

```
#include <stdio.h>

int main()
{
   int PetNejvetsich[5]= {0,0,0,0,0};
   int nactene;
   int pomocna;
   int i;

   printf("Zadevejte cisla k razeni\nzadavani cisel se ukonci znakem \"0\"\n\n");
   while(1)
   {
      scanf("%d",&nactene);
```

```
        if(nactene==0)
        {
           break;
        }
        for(i=0; i<5; i++)
        {
           if(nactene>PetNejvetsich[i])
           {
              pomocna=PetNejvetsich[i];
              PetNejvetsich[i]=nactene;
              nactene=pomocna;
           }
        }
     }
     printf("\nRada je:\n");
     for(i=0; i<5; i++)
     {
        printf("\n %d",PetNejvetsich[i]);
     }
     return 0;
}
```

Tento zdrojový kód lze nalézt v lekci č. 9 i mimo výukovou prezentaci.



Obrázek 12: Vybrané slajdy lekce č. 9

Na obrázku č. 12 lze vidět vybrané dva slajdy z úvodní prezentace, přičemž celou prezentaci si lze prohlédnout v příloze č. 1.

6.3.10 Lekce č. 10: Tvorba vlastních funkcí

Desátá lekce se zabývá tvorbou funkcí. Pokud se budou vyskytovat v kódu programu nějaké úseky, které se budou opakovat na různých místech kódu, je vhodné si pro tyto

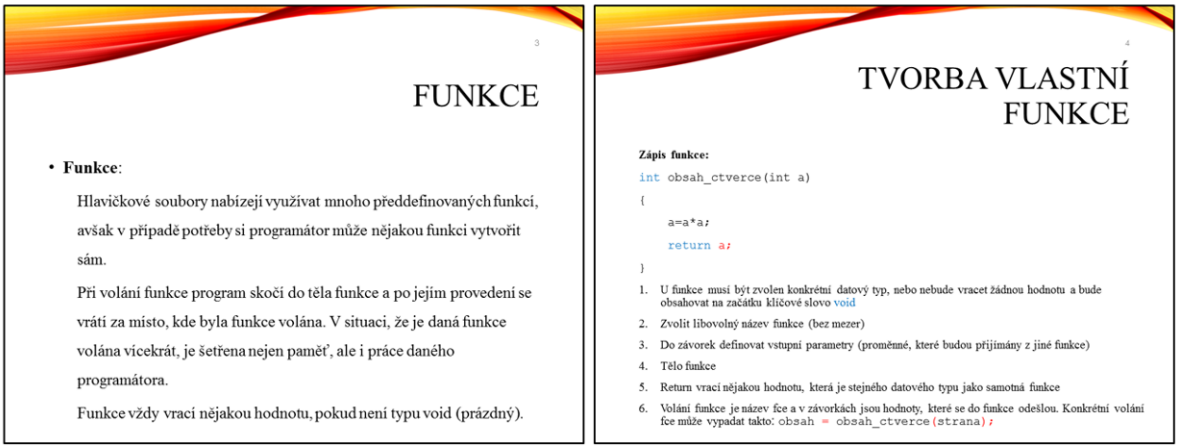
úseky kódu vytvořit funkci, která bude volána. Tímto se stane kód přehlednější a úspornější.

Autor zpočátku své prezentace vysvětluje další způsoby, kde je vhodné vlastní funkce vytvářet, jakým způsobem program do funkcí vstupuje a jak se vrací zpět. Dále uvádí, že každý funkce vždy vrací nějakou hodnotu a pouze funkce typu void (prázdný) žádnou hodnotu nevrací.

Na dalším slajdu je již studentům představena samotná tvorba vlastní funkce, přičemž je zápis funkce demonstrován na funkci, která má za úkol vypočítat obsah čtverce. Autor v jednotlivých bodech popisuje, že u funkce musí být zvolen konkrétní datový typ, nebo nebude vracet žádnou hodnotu a bude obsahovat na začátku klíčové slovo void. Poté je nutné zvolit libovolný název funkce (bez mezer) a do závorek definovat vstupní parametry (proměnné, které budou přijímány z jiné funkce). Dále je třeba vytvořit tělo funkce, neboli sekvenci příkazů a funkcí, které budou vykonány po zavolání dané funkce. Dále upozorňuje studenty, že klíčovým slovem return se vrací hodnota shodného datového typu, jako je samotná funkce. Posledním bodem je ukázka, jakým způsobem lze funkci volat. Volání funkce je název fce a v závorkách jsou hodnoty, které se do funkce odešlou. Konkrétní volání fce může vypadat takto: obsah = obsah_ctverce(strana);.

V této prezentaci je dále definován význam rekurzivní funkce, tj. funkce, která volá samu sebe. Studentům je pro lepší představu objasněno, že tato funkce nachází své využití například při výpočtu faktoriálu.

Pod prezentací jsou navíc vloženy dva zdrojové kódy programů. První je již pro zmíněný výpočet obsahu čtverce doplněn o výpočet objemu čtverce a pro veškeré činnosti jsou vytvořeny speciální funkce, které jsou volány v hlavní funkci programu. Druhý zdrojový kód je zaměřen na výpočet faktoriálu pomocí rekurzivní funkce. Oba tyto kódy jsou pro svou velikost uvedeny až na přiloženém CD/Zdrojové kódy/L10_obsah_obovod a */L10_faktorial.



Obrázek 13: Vybrané slajdy lekce č. 10

Na obrázku č. 13 lze vidět vybrané dva slajdy z úvodní prezentace, přičemž celou prezentaci si lze prohlédnout v příloze č. 1.

6.3.11 Lekce č. 11: Práce se soubory

Lekce číslo 11 je poslední výukovou lekcí a zabývá se prací se soubory. Studenti se jistě v budoucnu setkají s programy, které nebudou načítat vstupní hodnoty z klávesnice, ale ze souboru a naopak výsledky se nebudou vždy vypisovat na konzoly, ale bude je nutné vypsat do souboru.

V této prezentaci je definován datový typ file, který slouží pro práci se souborem. Poté je na tomto slajdu znázorněn zápis definování datové proměnné, který může být v takovéto podobě:

```
FILE *fr;       // pro čtení ze souboru (anj. read)

FILE *fw;       // pro zápis do souboru (anj. write)
```

Následně jsou studenti seznámeni s funkcí sloužící k otevírání souborů. Autor v prezentaci uvádí, že v té chvíli, kdy jsou již nadefinované proměnné pro práci se souborem, je nutné k nim přiřadit adresu souboru a soubor otevřít. Otvírání se provádí funkcí fopen, což je funkce která má dva parametry. První je celý název souboru včetně adresy a v případě, že se soubor nachází ve shodném adresáři jako program, pak stačí jen název souboru. Druhý parametr označuje, co se se souborem bude dělat (např. r = číst, w = zapisovat). Na slajdu je znázorněn rovněž zápis kódu sloužící pro otevření souboru.

```
fr=fopen(“adresa_souboru“,“r“);        // otevření souboru pro čtení

fw=fopen(“adresa_souboru“,“w“);     // otevření souboru pro zápis
```

Pokud se při otevírání zjistí, že cílový soubor pro zápis neexistuje, bude touto funkcí vytvořen.

Stejně jako existují funkce pro zápis na konzoly, existují taktéž funkce, které mohou číst i zapisovat přímo do souboru. Autor tedy v prezentaci dále uvádí, že pro zapisování a vypisování ze souboru dostatečně postačí výpis a zápis:

```
Výpis: znak=getc(fr);              // načte znak ze souboru pro čtení

Zápis: putc(znak,fw);              // zapíše znak do souboru pro zápis
```

Tyto funkce se používají zejména ve spojení s cyklem while – např. načítej znaky tak dlouho, dokud znak nebude EOF (end of file).

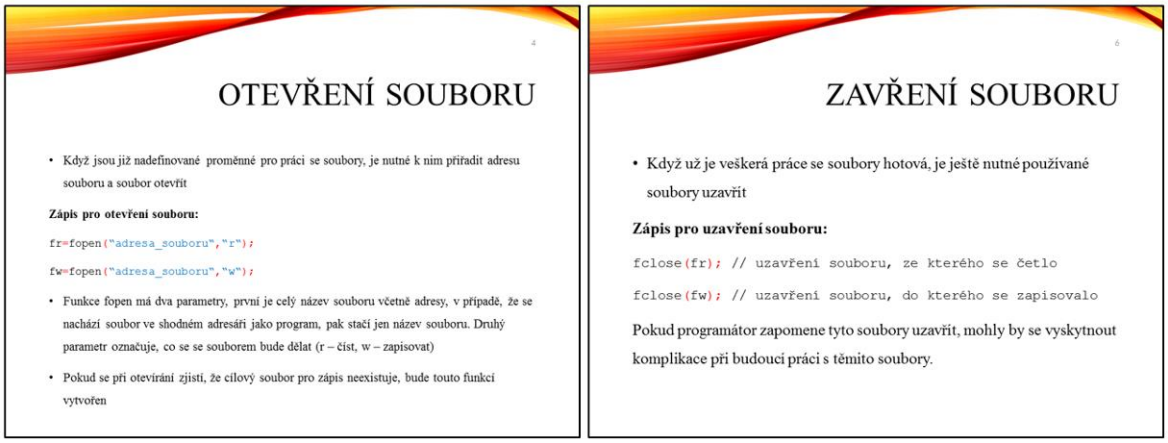
Pokud již je veškerá práce se soubory dokonána, je ještě potřeba všechny otevřené soubory uzavřít. Pokud by programátor tyto soubory zapomněl uzavřít, mohly by se vyskytnout komplikace při budoucí práci s nimi. Studentům je opět představen zápis, kterým se provede uzavření souboru:

```
fclose(fr);        // uzavření souboru, ze kterého se četlo

fclose(fw);       // uzavření souboru, do kterého se zapisovalo
```

V poslední výukové lekci jsou rovněž pod prezentací uvedeny dva zdrojové kódy programů zabývající se práci se soubory. Jeden zdrojový kód slouží k počítání znaků a řádků v souboru a lze ho nalézt na přiloženém CD/Zdrojové kódy/L11_znaky_radky. Druhý zdrojový kód kopíruje obsah souboru do jiného a má takovouto podobu:

```
#include <stdio.h>

int main()
{
   FILE *fr, *fw;
   char znak;
```

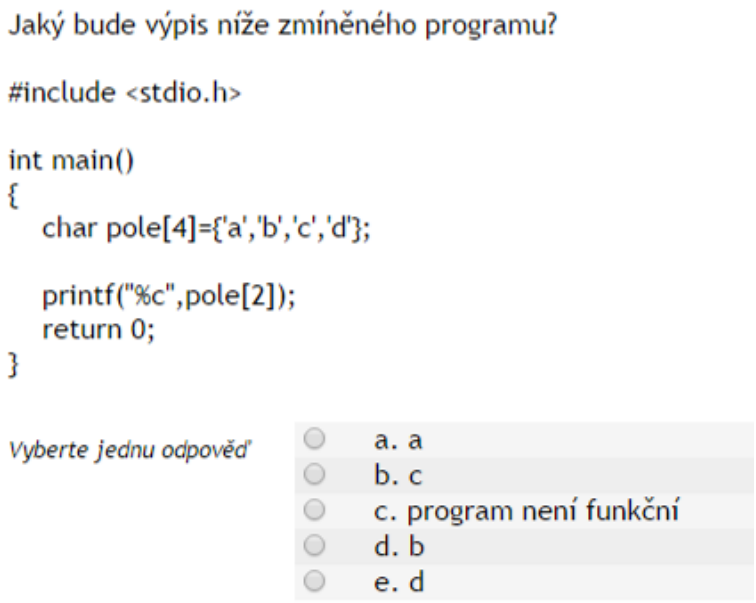
```
    fr=fopen("text.txt","r");
    fw=fopen("kopie_text.txt","w");
    printf("Byla vytvorena kopie souboru:text.txt\n");

    while((znak=getc(fr))!=EOF)
    {
       putc(znak,fw);
    }
    fclose(fr);
    fclose(fw);
    return 0;
}
```



Obrázek 14: Vybrané slajdy lekce č. 11

Na obrázku č. 14 lze vidět vybrané dva slajdy z úvodní prezentace, přičemž celou prezentaci si lze prohlédnout v příloze č. 1.

V jedenácté lekci se nachází poslední autotest, který mají studenti za úkol absolvovat. Tento autotest je na stejné bázi, jako předchozí dva autotesty a je v něm obsažena látka z 9. až 11. lekce. Celý test lze vidět na přiloženém CD/Testy/autotest3.



Obrázek 15: Vybraná otázka ze třetího autotestu

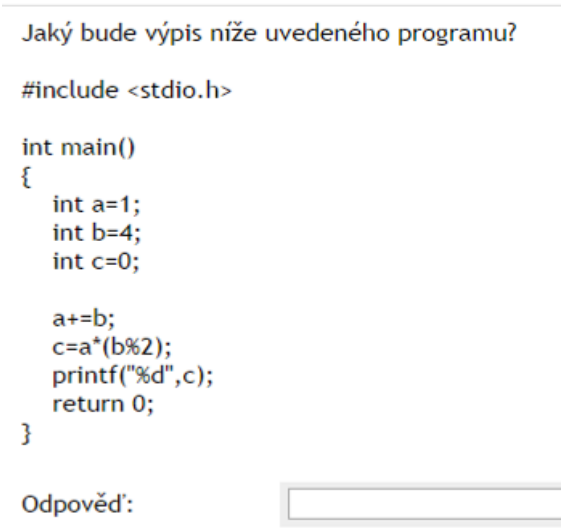
Na obrázku č. 6 je vybraná otázka z autotestu č. 3, která se zaměřuje na práci poli. Konkrétně se týká problematiky indexování položek v poli. Studenti tedy mají vyplnit, jaký bude výpis níže zmíněného programu, přičemž správný výpis je znak ‘c’.

Jedenáctá lekce je pro studenty pravděpodobně nejnáročnější, neboť mimo vyplňování autotestu obsahuje i zadání domácí úkolu. Tento domácí úkol č. 2 spočívá ve vytvoření programu, který ze souboru načte cca deset rodných čísel a do druhého souboru zapíše data narození a určí, zda se jedná o muže či ženu. Po domluvě s vyučujícím na střední průmyslové škole, na které byl kurz prezentován, byla i tato položka pro odevzdání domácího úkolu skryta. Jak již bylo zmíněno výše, studenty nebylo třeba více zatěžovat, neboť měli v této době několik vlastních úkolů.

6.3.12 Lekce č. 12: Závěrečný test

Ve dvanácte lekci se již studenti neučí novým věcem, je pro ně pouze připraven závěrečný test. Jak již bylo výše zmíněno, je nutné tento test absolvovat na 70% a studenti mají na splnění pouze dva pokusy. Test je složen z třiceti otázek, přičemž z každé výukové lekce jsou do testu zakomponovány tři otázky. Test je jak na teoretické bázi, tak i rovněž na praktické, kdy se zpravidla jedná o otázky, které se studentů ptají, jaký bude výpis po spuštění níže uvedeného programu. Praktické příklady jsou jednoduché,

čímž studentům není dovoleno využívat vývojových prostředí, ale musí ze zdrojového kódu sami, bez použití kompilátoru poznat, co bude program vykonávat.

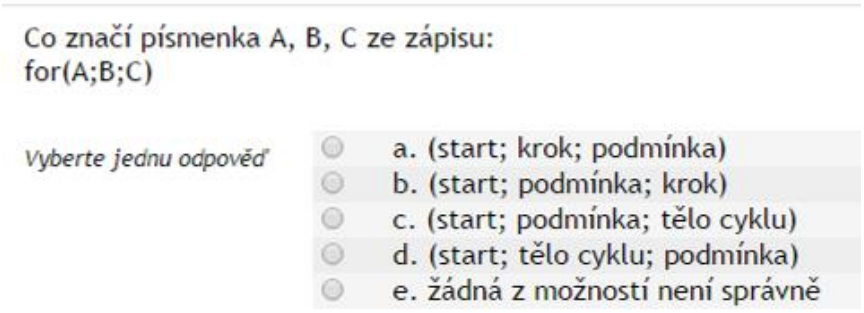

Jaký bude výpis níže uvedeného programu?

```
#include <stdio.h>

int main()
{
   int a=1;
   int b=4;
   int c=0;

   a+=b;
   c=a*(b%2);
   printf("%d",c);
   return 0;
}
```

Odpověď:

Obrázek 16: Vybraná prakticky zaměřená otázka ze závěrečného testu

Na obrázku č. 16 lze vidět vybranou otázku ze závěrečného testu, která je prakticky zaměřená. Studenti by měli poznat, jaký bude výpis níže uvedeného program zaměřeného na jednoduchou matematickou operaci s proměnnými, přičemž správná odpověď je 0.


Co značí písmenka A, B, C ze zápisu:
for(A;B;C)

Vyberte jednu odpověď

a. (start; krok; podmínka)
b. (start; podmínka; krok)
c. (start; podmínka; tělo cyklu)
d. (start; tělo cyklu; podmínka)
e. žádná z možností není správně

Obrázek 17: Vybraná teoreticky zaměřená otázka ze závěrečného testu

Na obrázku č. 17 je rovněž zobrazena vybraná otázka ze závěrečného testu, tentokrát teoretického charakteru. Studentů se ptá, co značí podmínka for(A;B;C), přičemž správná odpověď je ukrytá pod písmenem b. (start;podmínka;krok).

Dále v této kapitole jsou studenti požádáni o vyplnění zpětné vazby autorovi kurzu. Zpětná vazba poslouží k vyhodnocení kurzu a vytvoření návrhů a opatření vedoucí

k optimalizaci a realizovatelnosti samotného kurzu. Z časových důvodů byly studentům položeny pouze tři otázky:

1. Uvítali byste k prezenční formě výuky výuku prostřednictvím e-learningového kurzu? (Svou odpověď prosím odůvodněte.)

2. Přišel Vám tento kurz těžký, lehký, či přiměřený Vašim znalostním předpokladům? (Svou odpověď prosím odůvodněte.)

3. Měl(a) jste v kurzu s něčím problémy (technické problémy, neporozumění probírané látce, nevyváženost konkrétních lekcí)?

Výsledky závěrečného testu i zpětné vazby lze nalézt v kapitole č. 6. 5 a celou ukázku závěrečného testu lze dohledat na přiloženém CD/Testy/zaverecny_test.

6.3.13 Zkušební provoz

V rámci fáze vývoje byl po vytvoření všech lekcí a jejich dílčích částí zrealizován v LMS Moodle prototyp kurzu s následnou optimalizací jeho kvality. Autor poskytl přístup do kurzu třem studentům vysoké školy, kteří programovací jazyk C ovládají. Společně s nimi procházel celý kurz a zaznamenával veškeré objevené nedostatky (překlepy v textu, špatná formulace vět, nevyváženost jednotlivých lekcí, nejednoznačně formulované otázky v testech, atp.). Veškeré zmapované poznatky byly do kurzu zapracovány, což vedlo k rapidnímu zkvalitnění kurzu.

6.4 Realizace

Ve fázi realizace je vytvořený kurz podroben ostrému provozu. Jelikož cílová skupina se má skládat ze studentů střední školy, musel autor najít střední školu, na které probíhá prezenční výuka předmětu zabývajícího se programováním v jazyce C. Autor se při výběru střední školy obrátil na své bývalé pedagogy ze Střední průmyslové školy a Vyšší odborné školy v Kladně. Této prosbě vyšel vstříc pedagog Ing. Jaroslav Volšický, se kterým po e-mailové komunikaci bylo dohodnuto, že umožní tento kurz otestovat na svých současných studentech. Jednalo se o tři skupiny studentů, přičemž dvě skupiny

byly ze třetího ročníku a jedna skupina z ročníku čtvrtého. Skupiny byly cca po deseti studentech a všichni již v minulosti výše zmiňovaný předmět absolvovali.

Autor představil svůj kurz těmto jednotlivým skupinám studentů během dvouhodinové výuky. Po krátkém uvedení kurzu, byly studentům zadány autotesty i závěrečný test. Ve zbylém čase byli studenti požádáni o vyplnění krátké zpětné vazby autorovi kurzu. Výsledky tohoto šetření lze nalézt v kapitole 6. 5. 2.

6.5 Hodnocení

V poslední fázi byla posuzována kvalita kurzu Algoritmizace a programování v jazyce C úspěšnost testů. Nejoptimálnější cestou pro zhodnocení kvality kurzu by bylo, kdyby studenti celou výuku ve vytvořeném kurzu absolvovali včetně odevzdávání domácích úkolů, plnění dílčích autotestů a závěrečného testu a nakonec i zodpovězení zpětné vazby vyučujícímu. Tato výše zmíněná varianta však nemohla být bohužel realizována ze dvou důvodů. Prvním důvodem bylo, že nebyl dostatek času (jeden školní rok) na výuku. Druhým důvodem byl fakt, že autor nemůže studenty střední školy přinutit, aby vytvořený kurz absolvovali v celém jeho rozsahu. Z těchto důvodů se autor rozhodl vyhledat takové studenty, kteří již prezenční výuku předmětu zabývajícího se programovacím jazykem C absolvovali, a pro vyplňování testů by tak měli mít dostačující znalosti, které v tomto případě budou suplovat celé absolvování kurzu.

Tato kapitola se zabývá hodnocením jednotlivých autotestů a závěrečného testu, včetně identifikování nejobtížnějších a nejméně náročnějších kapitol i dílčích otázek. Zároveň jsou v této kapitole výstupy ze zpětné vazby studentů a z krátkého rozhovoru se středoškolským pedagogem Ing. Jaroslavem Volšickým.

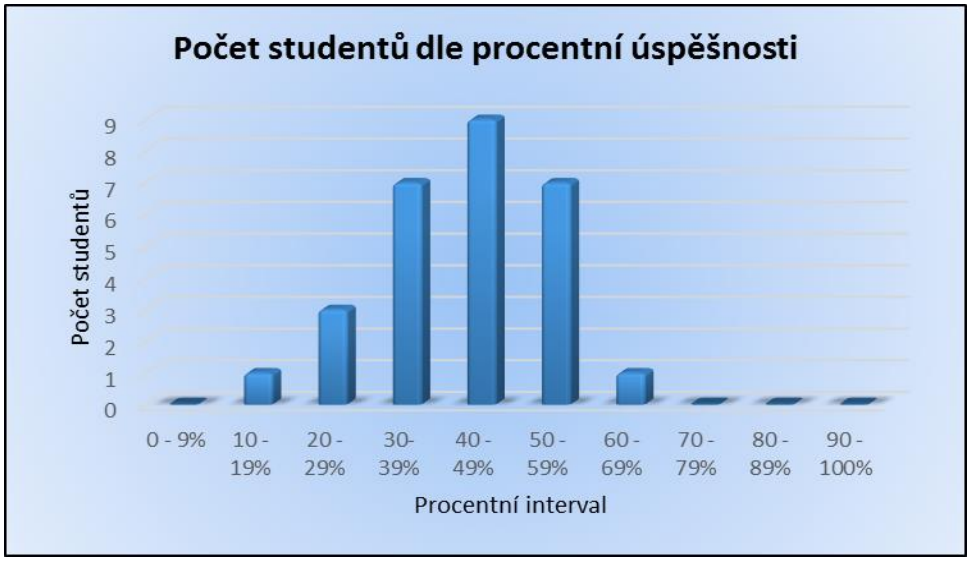
6.5.1 Hodnocení jednotlivých testů

Jak již bylo zmíněno v předchozích kapitolách, pro studenty byly připraveny tři autotesty a jeden závěrečný test. Veškeré testování bylo realizováno ve dvou třídách ze Střední průmyslové školy v Kladně. Konkrétně se jednalo o třídy EP3 a EP4, což jsou studenti třetích a čtvrtých ročníků z oboru Elektronické počítačové systémy.

6.5.1.1 Autotest č. 1

Výsledky z prvního autotestu byly velmi slabé, z 28 testovaných studentů uspěl pouze jeden student což je tedy 3,57% úspěšnost. Tento autotest byl vůbec nejméně úspěšným ze všech ostatních testů. Největší problém studentům činily otázky ze čtvrté lekce, u které dosahovali pouze 28% úspěšnosti na otázku. Naopak největší úspěch měli studenti ve třetí lekci, kde dosahovali hranice 60%. Autor tento nevalný výsledek přisuzuje skutečnosti, že lekce jsou prozatím pouze teoretického charakteru, který je pro studenty méně záživný.

V níže uvedeném grafu č. 1 je uveden počet studentů dle procentní úspěšnosti. Lze vidět, že nejvíce studentů (23 z 28) se pohybovalo v rozmezí od 30% do 59%.

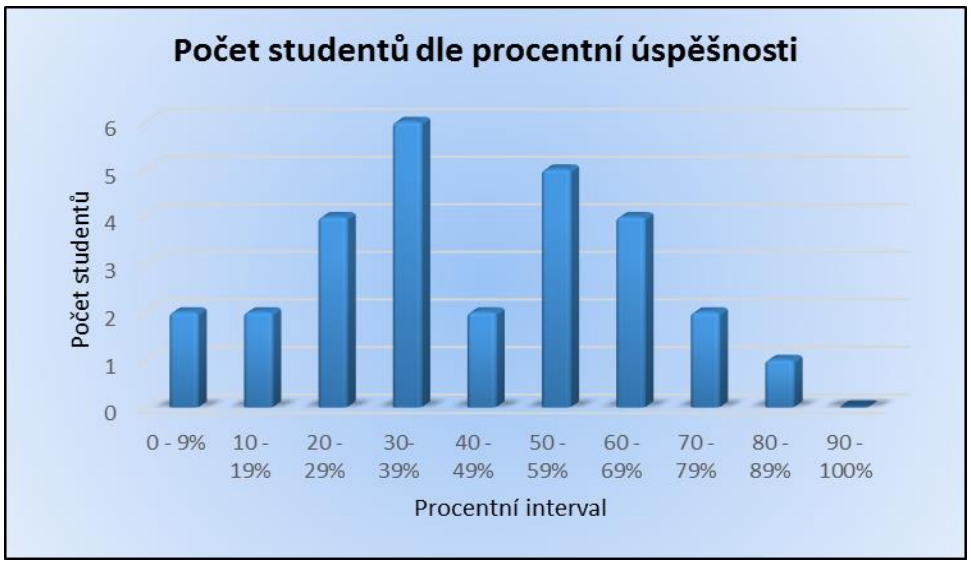


Graf 1: Počet studentů dle procentní úspěšnosti v autotestu č. 1

6.5.1.2 Autotest č. 2

Autotest č. 2 již dopadl o poznání lépe. Z 28 testovaných studentů splnilo test 7 studentů, což je 25% úspěšnost. Tento výsledek je stále slabý, avšak již nikoliv tak fatální. Tato vyšší úspěšnost by se mohla přisuzovat k faktu, že otázky již byly z velké části prakticky zaměřené. Při porovnání úspěšností jednotlivých lekcí nebyly zjištěny žádné markantní rozdíly. Každá lekce měla průměrnou úspěšnost na otázku 40% s odchylkou 6%.

V uvedeném grafu č. 2 je opět uveden počet studentů dle procentní úspěšnosti. Je možné vidět, že někteří studenti nedosáhli ani 10% a přitom měl autotest č. 2 vyšší úspěšnost, než autotest č. 1. Další zajímavostí je, že pěti studentům uniklo dosáhnutí požadované minimální procentní hranice o jednu otázku.



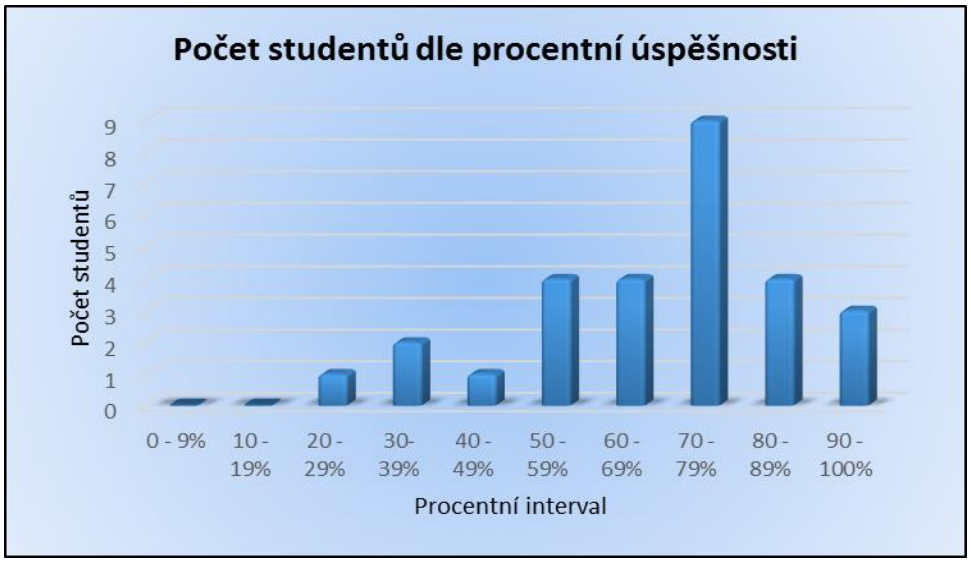
Graf 2: Počet studentů dle procentní úspěšnosti v autotestu č. 2

6.5.1.3 Autotest č. 3

Autotest č. 3 dopadl ze všech testů nejlépe. Úspěšnost tohoto autotestu byla výrazně vyšší, než u ostatních testů. Tento test splnilo 20 studentů z 28, což představuje 71,43% úspěšnost. Autotest již byl zaměřen z větší části prakticky, testoval často porozumění zdrojovým kódům a neobsahoval otázky typu multiple-choice[2]. Autor předpokládá, že vyšší úspěšnost byla z toho důvodu, že studenti v prezenční výuce probírali látku obsaženou v tomto autotestu v nedávné době. Také úspěšnost jednotlivých lekcí dokazuje výborné výsledky studentů. Nejobtížnější lekcí se studentům jevila lekce č. 10, která zaznamenala 58% úspěšnost, což je stále velmi dobrá bilance v porovnání s ostatními lekcemi.

V uvedeném grafu č. 3 je rovněž vyobrazen počet studentů dle procentní úspěšnosti. Je možné vidět, že zde je již úspěšnost studentů skutečně vyšší. Pouhých osm studentů v testu neuspělo a autor věří, že i tito studenti by dokázali uspět, kdyby byli více motivováni (reálné známkování testu).

[2] Multiple-choice značí výraz pro otázku s více možnými odpověďmi.



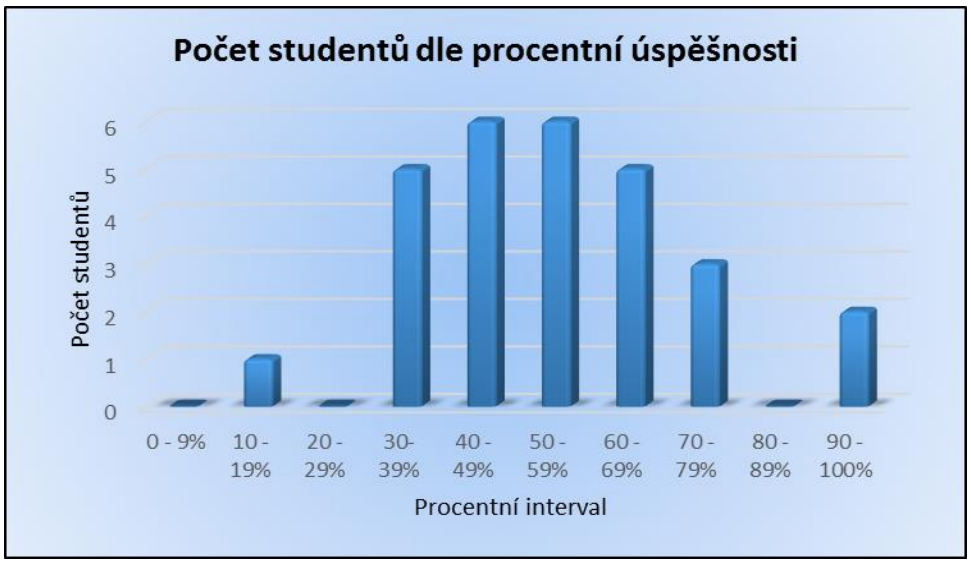
Graf 3: Počet studentů dle procentní úspěšnosti v autotestu č. 3

6.5.1.4 Závěrečný test

Posledním a nejdůležitějším testem celého kurzu je závěrečný test. Jak již bylo zmíněno výše, tento test obsahuje ve třiceti otázkách celou problematiku daného kurzu a studenti na jeho vyplnění mají třicet minut. Zásadní rozdíl mezi tímto testem a autotesty spočíval ve zvýšení minimální hranice pro úspěšné absolvování, která byla zvednuta o 10%, tzn. z 60% na 70%. Po vyhodnocení výsledků vzešlo najevo, že z 28 studentů test úspěšně absolvovalo 18% studentů, což představuje 5 osob. Kdyby tato hranice zůstala na stejné úrovni, která je nastavena v autotestech, byla by již úspěšnost závěrečného testu 36%. V závěrečném testu činily studentům největší problémy otázky z lekcí č. 2 a č 10. U lekce č. 2 tento výsledek nebyl překvapující, neboť studenti potvrdili svou neznalost teoretických základů shodně, jako v autotestu č. 1, zde dosáhli 45% úspěšnosti. U lekce č. 10 byla v závislosti na výsledcích ze třetího autotestu předpokládaná úspěšnost vyšší a ne pouhých 44%. Tuto nesrovnalost si autor vysvětluje skutečností, že studenti nerozeznali v daných otázkách tzv. chytáky, nechali se zmást, což samozřejmě vedlo k chybné odpovědi. Naopak nejúspěšnější lekcí byla jednoznačně lekce č. 11, kde studenti odpovídali správně z 82%.

V níže uvedeném grafu č. 4 je znázorněn počet studentů dle procentní úspěšnosti v závěrečném testu. Lze z něho vidět, že úspěšnost převážné části studentů se pohybovala v interval od 30 % do 69%, což je však bohužel pod minimální procentní hranicí

pro splnění závěrečného testu. Je potěšující zmínit, že dva studenti dosáhli výsledků nad 90%, přičemž jeden z nich absolvoval celý test bez jediné chyby.

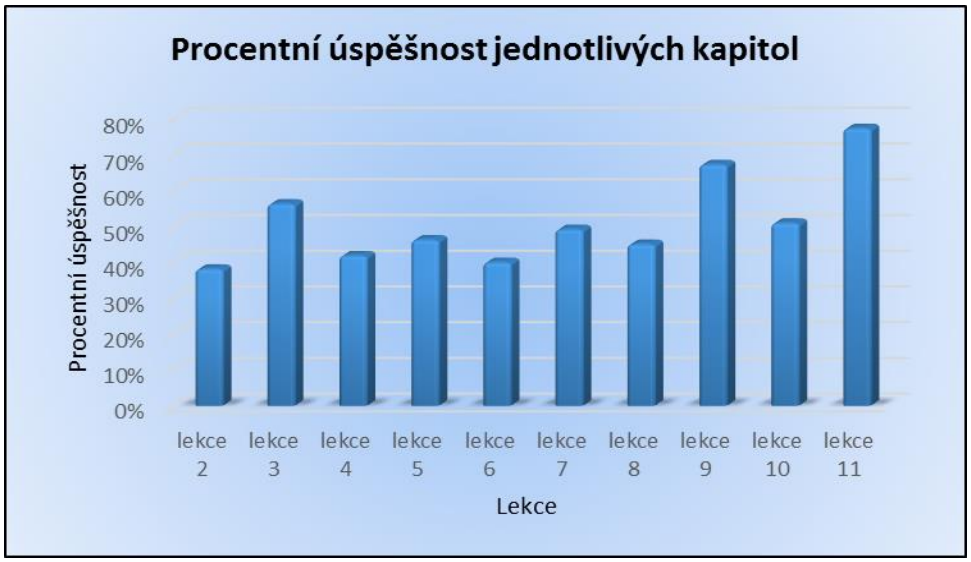


Graf 4: Počet studentů dle procentní úspěšnosti v závěrečném testu

6.5.1.5 Vyhodnocení nejkritičtějších otázek a kapitol

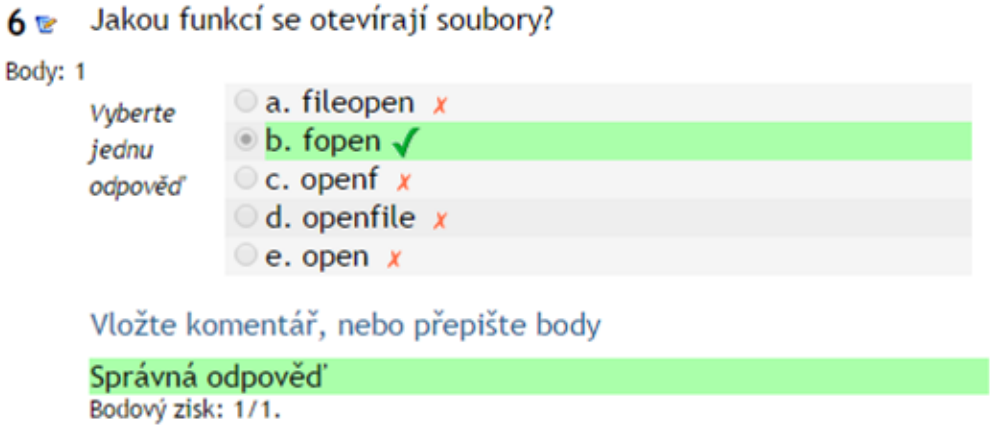
Tato část se zaměřuje na identifikaci kapitol a otázek, které studentům činily nejmenší a největší problémy v rámci všech testů.

Z grafu č. 5 je zřejmé, že znalosti studentů z jednotlivých lekcí jsou téměř vyrovnané. Průměrná úspěšnost lekcí byla 51%. Nejobtížnější lekcí se ukázala lekce č. 2 s pouhou 38% úspěšností a naopak nejmenší problémy studentům činila kapitola č. 11, která zaznamenala 78% úspěšnost.



Graf 5: Procentní úspěšnost jednotlivých kapitol

Z výše uvedeného vyplývá, že by autor kurzu měl pro další studenty, kteří se do kurzu přihlásí v příštím roce, zkvalitnit právě ty kapitoly, které mají v grafu č. 5 nejmenší procentní úspěšnost.



Obrázek 18: Otázka, která studentům činila nejmenší problémy

Na obrázku č. 18 je znázorněna otázka, která studentům činila nejmenší problémy. Otázka byla vypracována na základě podkladů z prezentace v jedenácté lekci, kdy studenti měli z pěti možností vybrat, jakou funkcí se otevírají soubory. Jak již obrázek č. 18 napovídá, správnou odpovědí je možnost b. fopen. Jako jediná měla tato otázka u studentů celkovou úspěšnost 100%.

Naopak největší potíže studentům činila otázka z osmé lekce, která je zaměřena na průběh cyklu while a v něm vnořený cyklus for. Studenti měli za úkol vypsat, jaký bude

výstup programu, který byl pod otázkou zapsán. Bohužel na rozdíl od předešlé otázky, na tuto otázku odpovědělo správně pouze 7% studentů. Neboť je otázka rozsáhlejšího charakteru, je uvedena až v příloze č. 2.

6.5.2 Zpětné vazby studentů

Následující část se zaměřuje na subjektivní názory studentů, kteří měli příležitost do kurzu nahlédnout. Touto cestou se mohli studenti vyjádřit své názory na níže uvedené otázky.

Poté co studenti odeslali vyplněné závěrečné testy, byli autorem požádáni, zda by ještě nezodpověděli tři krátké otázky, které by napomohly s vyhodnocováním této diplomové práce. Z 28 studentů tak učinilo 21 a zbylých 7 studentů se nevyjádřilo. Znění otázek bylo následující:

1) Uvítali byste k prezenční formě výuky výuku prostřednictvím e-learningového kurzu? (Svou odpověď prosím odůvodněte.)

2) Přišel Vám tento kurz těžký, lehký, či přiměřený Vašim znalostním předpokladům? (Svou odpověď prosím odůvodněte.)

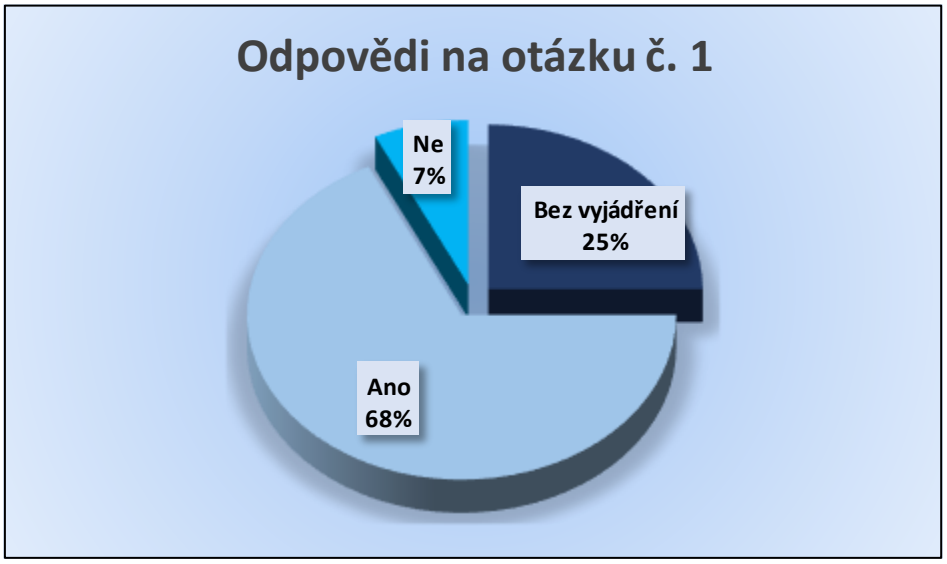
3) Měl (a) jste v kurzu s něčím problémy (technické problémy, neporozumění probírané látce, nevyváženost konkrétních lekcí)?

6.5.2.1 Vyhodnocení otázky č. 1

První otázka měla za úkol zjistit, zda by studenti uvítali jako doplněk ke klasické prezenční výuce podobný e-learningový kurz, jako je kurz Algoritmizace a programování v jazyce C.

Výsledky byly pro autora velice příznivé. 68% studentů odpovídalo kladně. Jako důvod k zavedení kurzu k prezenční výuce uváděli např.: přístup k informacím, prohloubení a ověření znalostí získaných z prezenční výuky, atp.. Naopak záporně hodnotilo kurz pouhých 7% studentů a hlavních důvodem bylo, že studenti pro vysvětlování této problematiky potřebují přímou komunikaci s vyučujícím. Zbylých

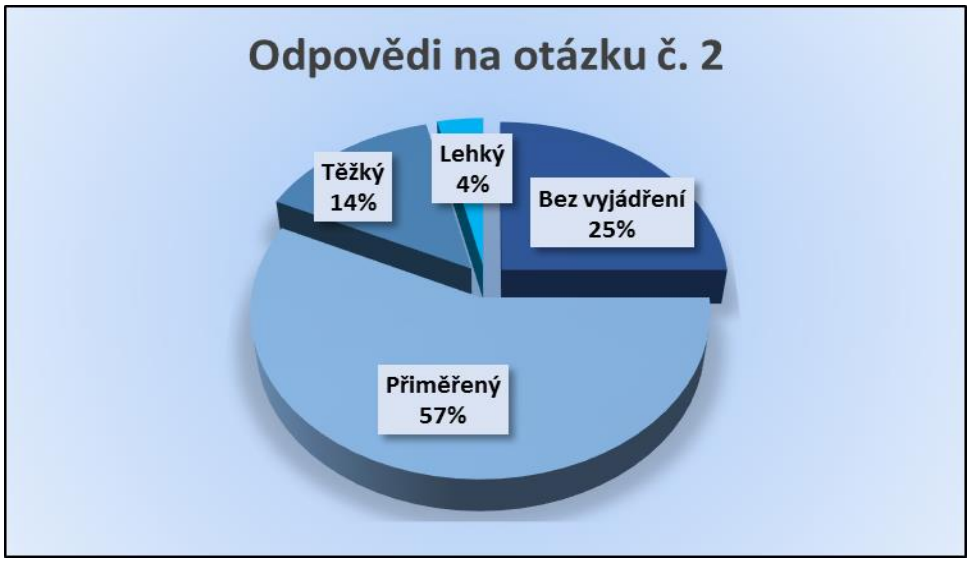
25% studentů se bohužel k této otázce nijak nevyjádřilo. Graf č. 6 reprezentuje procentuální zastoupení jednotlivých skupin odpovědí.



Graf 6: Odpovědi na otázku č. 1

6.5.2.2 Vyhodnocení otázky č. 2

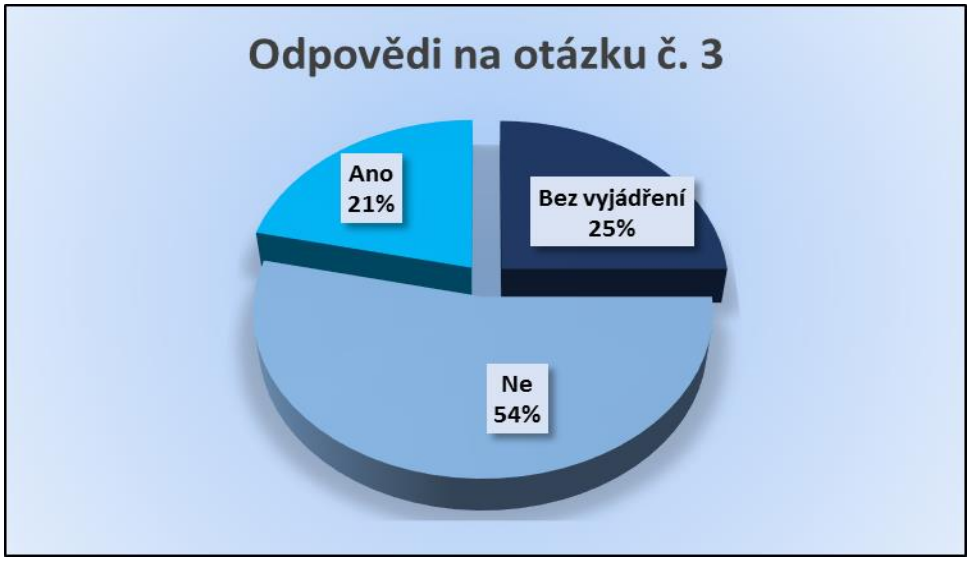
Druhou kladenou otázkou byla zjišťována studentem vnímaná obtížnost kurzu. Je nutné konstatovat, že se bohužel opět jedna čtvrtina studentů k dané otázce vůbec nevyjádřila. I přes velmi nepříznivé výsledky z jednotlivých testů hodnotilo test 57% studentů jako přiměřený. Svou neznalost odůvodňovali tím, že předmět vyučovaný na Střední průmyslové škole v Kladně neprobírá programování jazyka C do takové hloubky, aby byli schopni zodpovědět všechny položené otázky, především teoretického charakteru. 14% studentů bylo toho názoru, že daný kurz byl příliš těžký. Pouhé 4% studentů považovali kurz za lehký, avšak opět k otázce připisovali, že postrádají dostatečný teoretický základ, aby mohli být v testech úspěšnější. V grafu č. 6 lze opět procentuální vyjádření odpovědí studentů.



Graf 7: Odpovědi na otázku č. 2

6.5.2.3 Vyhodnocení otázky č. 3

Poslední otázka vyhodnocovala, zda studenti během absolvování kurzu objevili nějaké nedostatky. Jako u předchozích dvou otázek, se opět čtvrtina studentů vůbec nevyjádřila. 54% studentů nenalezlo v kurzu žádné nedostatky a zbylých 21% zmínilo, že se v kurzu s některými problémy přece jen potýkali. Vždy se však jednalo o neporozumění probírané látky. V grafu č. 7 lze opět vidět procentuální vyjádření odpovědí studentů na otázku č. 3.



Graf 8: Odpovědi na otázku č. 3

6.5.3 Zpětná vazba pedagogického pracovníka

Tato kapitola je zaměřena na hodnocení kurzu z jiného pohledu, tj. z pohledu pedagogického pracovníka, který na Střední průmyslové škole v Kladně vyučuje předmět týkající se dané problematiky kurzu. Pohled pedagogického pracovníka je důležitý, neboť již má v oboru učitelství několikaletou praxi a jeho připomínky a názory jsou tak pro autora velmi inspirativní a cenné. Panu Ing. Jaroslavu Volšickému byl umožněn vstup do kurzu na LMS Moodle již několik dnů před samotným testováním studentů, aby si mohl udělat o kurzu ucelený obraz. Poté, co toto testování studentů proběhlo, zodpověděl Ing. Volšický autorovi několik otázek týkajících se kurzu. Níže je uveden krátký dialog, kdy je nejprve položena otázka autora a následně pod ní zaznamenána odpověď Ing. Jaroslava Volšického.

- Jak dlouho vyučujete?
    - *Na SPŠ Kladno učím 10 let.*
- Co si myslíte o doplnění prezenční výuky o E-learningový kurz?
    - *E-learningový kurz je vhodný pro doplnění výuky.*
- Považujete autorem vytvořený kurz za přínosný?
    - *Ano, umožňuje prověření znalostí a je zpětnou vazbou pro vyučujícího.*
- Shledáváte v kurzu nějaké nedostatky?
    - *Ne.*
- Myslíte si, že náročnost testů a ostatní podmínky jsou přiměřené?
    - *Ano, náročnost testu odpovídá výuce programování na naší škole ve 3. ročníku.*
- Jeví se Vám lekce v kurzu dostatečně srozumitelné pro studenty?
    - *Ano, studentům je vše srozumitelně vysvětleno, i testy proběhly bez problému.*

Dle odpovědí výše zmíněných lze konstatovat, že byl pedagog s vytvořeným e-learningovým kurzem spokojen, neměl žádné výhrady ani ke kurzu, ani ke způsobu testování studentů a doporučil by ho ke kombinaci s prezenční výukou.

7 Doporučení a závěr

Cílem teoretické části diplomové práce bylo charakterizovat teoretické principy elektronického vzdělávání. Nejprve byl samotný e-learning definován a poté byly nastíněny nejpodstatnější historické milníky ve vývoji e-learningu. Dále bylo provedeno rozčlenění e-learningu do jednotlivých forem a následně vymezeny veškeré výhody a nevýhody ze třech základních pohledů (tj. z pohledu studenta, učitele a instituce). Po charakteristice této vzdělávací formy byl představen model ADDIE a jak dle tohoto modelu vytvářet e-learningové kurzy. Poslední teoretická kapitola se věnovala LMS, jeho výhodám a nevýhodám a zejména poté jednomu jeho zástupci, tj. LMS Moodle.

Na základě informací vycházejících z teoretické části mohla být zrealizována praktická část. Cílem praktické části diplomové práce bylo v prostředí LMS Moodle vytvořit e-learningový kurz Algoritmizace a programování v jazyce C pro studenty střední školy. Kurz byl otestován vybranými studenty třetích a čtvrtých ročníků oboru Elektronické počítačové systémy na Střední průmyslové škole v Kladně. Bohužel z časových důvodů a nekompetentnosti autora vůči studentům bylo studentské absolvování kurzu suplováno pouze znalostmi, které studenti získali již během prezenční výuky. Vedení SPŠ Kladno však povolilo, aby studenti vyplnili připravené testy, které jsou zakomponovány ve vytvořeném kurzu.

Výsledky testů jsou nejobjektivnějším nástrojem k hodnocení kvality kurzu. Jelikož studenti kurz neabsolvovali celý, nelze hodnocení zakládat pouze na těchto výsledcích. Pro zjištění efektivnosti kurzu by tedy bylo vhodné v příštím školním roce nechat studenty absolvovat celý kurz od začátku se všemi náležitostmi a výsledky z testování porovnat s aktuálními výsledky, získané touto prací. Bylo by vhodné, aby na střední škole probíhal daný kurz zároveň s prezenční výukou, což umožní studentům utřídit své poznatky a eliminovat případné nesrovnalosti.

Hodnocení kurzu je v této diplomové práci realizováno ze tří různých pohledů. První pohled se zaměřuje na výsledky studentů v testech. Druhý je zaměřen na studentské názory na vytvořený e-learningový kurz a na obtížnost testů. Poslední pohled je pohled pedagogického pracovníka.

Testy ukázaly, že vybraní studenti mají velmi zásadní znalostní nedostatky. Nejslabší úspěšností (38%) disponovala kapitola teoretického charakteru, v níž byly vysvětleny základní pojmy. Pokud by se kurz měl pro studenty v dalším roce realizovat paralelně s prezenční výukou, bylo by vhodné, aby právě ty kapitoly, které studentům dělají největší potíže, byly zkvalitněny a bylo jim ve výuce věnováno více času.

Velmi pozitivní výsledky pro autora kurzu přinesla zpětná vazba od studentů i od pedagogického pracovníka. Ukázalo se, že studenti i pedagogický pracovník by takovýto typ e-learningového kurzu k dosavadní prezenční výuce vřele uvítali.

Úplným závěrem by chtěl autor navrhnout, aby Střední průmyslová škola Kladno ve svém technickém zázemí zrealizovala LMS Moodle a vytvořila pro studenty e-learningové kurzy na vybranou skupinu předmětů. Tyto kurzy přidat k prezenční výuce a sledovat, zda se během jednoho školního roku u studentů projevila nějaká zlepšení. Pokud ano, měla by dále rozvíjet své kurzy a doporučit tuto kombinaci prezenční výuky s e-learningovými kurzy dalším středním školám.

8 Seznam použitých zdrojů

Abromitis online learning. *ADDIE*. [online]. [cit. 2014-10-25]. Dostupné z: <http://abromitis.com/about-us/portfolio/addie/>

BAREŠOVÁ, Andrea. 2011. *E-learning ve vzdělávání dospělých*. Praha: Vox, 197 s. ISBN 978-808-7480-007.

BURIAN, Pavel. 2014. *Internet inteligentních aktivit*. Vyd. 1. Praha: Grada, 332 s. Průvodce (Grada). ISBN 978-80-247-5137-5.

CASTAGNOLO, Chuck. *Ezine articles: The ADDIE Model - Why use it?* [online]. 2007 [cit. 2014-12-29]. Dostupné z: <http://ezinearticles.com/?The-ADDIE-Model---Why-Use-It?&id=859615>

DRLÍK, Martin. 2013. *Moodle: kompletní průvodce tvorbou a správou elektronických kurzů*. 1. vyd. Brno: Computer Press, 344 s. ISBN 978-80-251-3759-8.

EGER, Ludvík. 2012. *Vzdělávání dospělých a ICT: aktuální stav a predikce vývoje*. 1. vyd. Plzeň: Nava, 120 s. ISBN 978-807-2114-283.

GOERZ, Michael. *Complete ASCII table.* [online]. [cit. 2015-01-12]. Dostupné z: <http://www.cheat-sheets.org/saved-copy/ascii.png>

HAVLÍČEK, Zdeněk. 2007. *Podpora elektronického vzdělávaní na ČZU v Praze.* Závěrečná zpráva projektu, 1. vyd. Praha: Reprografické studio PEF ČZU, ISBN 978-80-213-1620-1.

HEROUT, Pavel. 2004. *Učebnice jazyka C.* 4. přeprac. vyd. České Budějovice: Kopp, 271 s. ISBN 80-7232-220-6.

JOHNSON, Steven. 2011. *Where Good Ideas Come From The Natural History of Innovation*. S.l.: Penguin Group US, ISBN 978-110-1444-207.

KOPECKÝ, Kamil. 2006. *E-learning (nejen) pro pedagogy*. Olomouc: Hanex, 125s. ISBN 80-85783-50-9.

Moodle. *Co je Moodle* [online]. 2006 [cit. 2015-01-03]. Dostupné z: <https://docs.moodle.org/archive/cs/Co_je_Moodle>

PEJŠA, Jan. *LCMS A LMS, vývoj kurzů*. Praha: Kontis, 2004 [cit. 2014-12-09]. Dostupné z: <http://www.e-learn.cz/soubory/LMS_LCMS.pdf>

PODLAHOVÁ, Libuše. 2012. *Didaktika pro vysokoškolské učitele*. 1. vyd. Praha: Grada, 154 s. ISBN 978-802-4742-175.

ROUBAL, Pavel. 2012. *Informatika a výpočetní technika pro střední školy: praktická učebnice*. 1. vyd. Brno: Computer Press, 112 s. ISBN 9788025132272.

SEDLÁČKOVÁ, Barbora. E-LKA2: E-learningový kurz knihovnické angličtiny.: Část I. *Inflow - information journal* [online]. 2011 [cit. 2014-11-12]. Dostupné z: <http://www.inflow.cz/elearningovy-kurz-knihovnicke-anglictiny-2-elka2#_ftn10>

STÁRKOVÁ, Dagmar. *Model ADDIE při vytváření koncepce výuky a jeho aplikace* [online]. 2012 [cit. 2015-12-09]. Dostupné z: <http://it.pedf.cuni.cz/strstud/edutech/2012_Addie_Starkova/>

ZOUNEK, Jiří, SUDICKÝ, Petr. 2012. *E-learning: učení (se) s online technologiemi*. 1. vyd. Praha: Wolters Kluwer Česká republika, 226 s. ISBN 978-80-7357-903-6.

9 Seznam schémat

10 Seznam obrázků

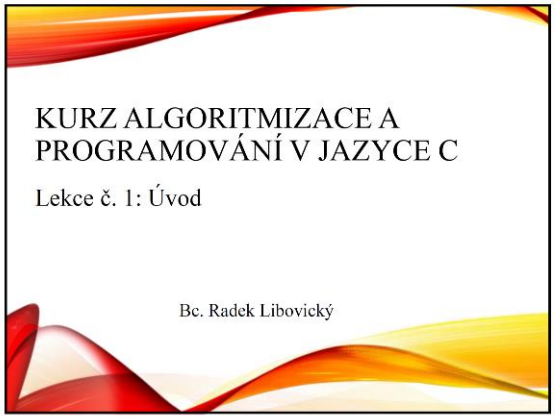
11 Seznam tabulek

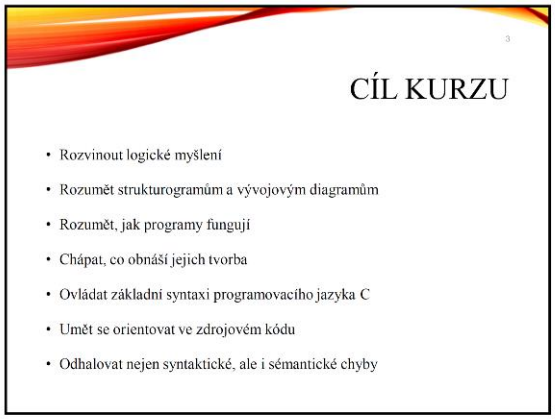
12 Seznam grafů

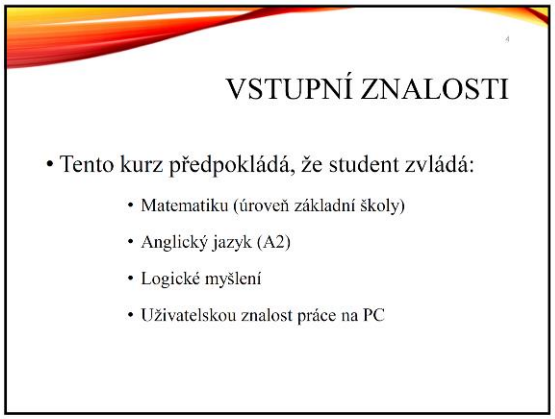
13 Seznam příloh

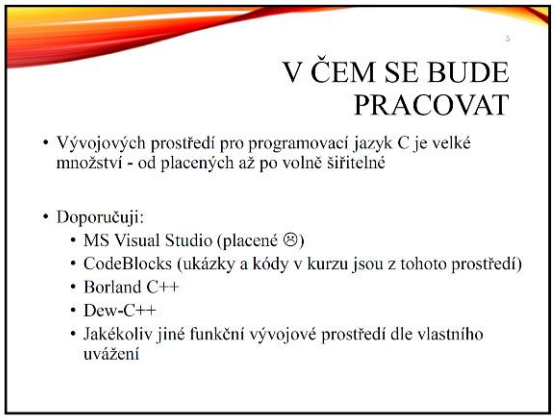
14 Přílohy

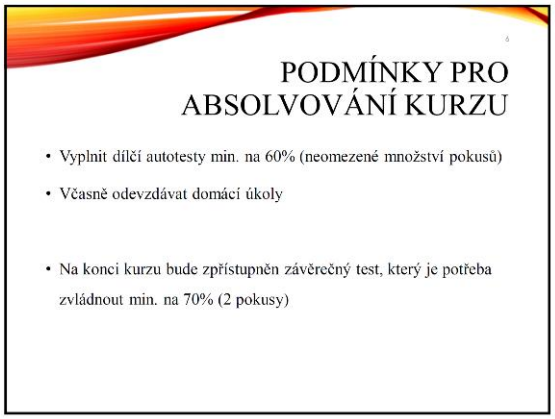
Příloha 1: Jednotlivé lekce kurzu Algoritmizace a programování v jazyce C











7

OSNOVA KURZU

1. Úvod
2. Základní pojmy
3. Způsoby zaznamenávání algoritmů
4. Programovací jazyk C
5. Hello world
6. Proměnné, jejich typy a práce s nimi
7. Podmínky (if, switch)
8. Cykly (while, do while, for)
9. Pole a práce s nimi
10. Tvorba vlastních funkcí
11. Práce se soubory
12. Závěrečný test

KURZ ALGORITMIZACE A PROGRAMOVÁNÍ V JAZYCE C

Lekce č. 2: Základní pojmy

Bc. Radek Libovický

9

LOGICKÁ HÁDANKA

- V týpí sedí dva členové kmene Apačů. Jeden je velký a druhý malý. Malý je syn velkého, ale velký není otec malého. Jak je to možné?

Odpověď na konci prezentace

10

ALGORITMUS

- **Algoritmus –** lze definovat jako jednoznačně určenou posloupnost konečného počtu elementárních kroků vedoucí k řešení daného problému (úlohy), přičemž musí být splněny základní vlastnosti každého algoritmu.

  (Algoritmus nemusí vždy souviset s programováním, i recept v kuchařce lze považovat za algoritmus)

- **Program** – realizuje konkrétní algoritmus prostřednictvím počítače

11

ŽIVOTNÍ CYKLUS PROGRAMU

1. Analýza problému
2. Stanovení podmínek, za kterých má program fungovat
3. Sestavení algoritmu
4. Sestavení programu
5. Testování (ladění) programu

12

VLASTNOSTI ALGORITMŮ

- **Konečnost** – musí mít konečný počet kroků
- **Správnost** – výsledek vydaný algoritmem musí být správný
- **Obecnost** – neřeší jeden konkrétní problém (2*9), ale obecnou třídu obdobných problémů (součin dvou čísel)
- **Rezultativnost** – po zadání vstupních dat vždy vrátí výsledek (může to být i chybové hlášení)

13

VLASTNOSTI ALGORITMŮ

- **Jednoznačnost -** V každé situaci musí být naprosto zřejmé, co a jak se má provést, jak má provádění algoritmu pokračovat
- **Opakovatelnost –** při stejných vstupních hodnotách musí vyjít vždy shodný výsledek
- **Srozumitelnost –** musí být srozumitelný i pro uživatele, který jej nevytvořili

14

ZÁKLADNÍ ŘÍDÍCÍ STRUKTURY ALGORITMU

- **Sekvence -** posloupnost prováděných operací
- **Selekce –** větvení algoritmu/programu
- **Iterace** – opakování (cyklus)

15

SYNTAXE A SÉMANTIKA

- **Syntaxe** - syntaxe programovacího jazyka je soubor pravidel, který definuje správnou kombinaci symbolů
- **Syntaktická chyba** – znamená, že je špatně zapsaný kód. Dobré je, že takovouto chybu lze snadno najít (kompilátor na ní upozorní) a opravit
- **Sémantika** - popisuje procesy, které řídí počítač při vykonávání programu v daném programovacím jazyce
- **Sémantická chyba** – kód je syntakticky správně, ale program nedělá co má

16

ODPOVĚĎ NA LOGICKOU HÁDANKU

- Velký člen kmene Apačů je totiž indiánka - je to maminka malého.

17

ZDROJE

- ROUBAL, Pavel. *Informatika a výpočetní technika pro střední školy: praktická učebnice*. Vyd. 1. Brno: Computer Press, 2010, 112 s. ISBN 9788025132272.

KURZ ALGORITMIZACE A PROGRAMOVÁNÍ V JAZYCE C

Lekce č. 3: Způsoby zaznamenávání algoritmů

Bc. Radek Libovický

19

LOGICKÁ HÁDANKA

- Policejní komisař si předvolá na stanici pět podezřelých z vraždy. Ví, že jeden z nich je určitě pachatel a právě tři výpovědi jsou pravdivé:
  Adam: "Dominik je vrah."
  Bořivoj: "Jsem nevinný."
  Cyril: "Nebyl to Eduard."
  Dominik: "Adam lže."
  Eduard: "Bořivoj říká pravdu."
  Kdo spáchal vraždu?

Odpověď na konci prezentace

20

PREZENTACE ALGORITMŮ

- **Popisem**
  - Pseudokód
- **Graficky**
  - Vývojový diagram
  - Strukturogram
- **Programovacím jazykem**

21

PSEUDOKÓD

- **Pseudokód** – je možností zápisu algoritmu pomocí slov bez závislosti na programovacím jazyku. Tvůrce pseudokódu se nemusí zabývat syntaxí konkrétního programovacího jazyka.

Popíjení piva:

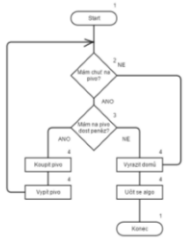
Cyklus: pokud mám chuť na pivo
  Pokud mám dost peněz
    Koupit pivo
    Vypít pivo
  Konec podmínky
  Návrat na začátek cyklu
Jinak konec cyklu
Vyrazit domů
Učit se algo

22

VÝVOJOVÝ DIAGRAM

- **Vývojový diagram** – zobrazení algoritmu v grafické podobě

Popíjení piva:



Legenda:
1 Začátek/konec
2 Iterace
3 Selekce
4 Sekvence

23

STRUKTUROGRAM

- **Strukturogram** zobrazení algoritmu v tabulkové podobě

Popíjení piva:

Mám chuť na pivo?	1
Mám na pivo dost peněz?	2
ANO / NE	
Koupit pivo 3	
Vypít pivo 3	
Učit se algo	3
Vyrazit domů	3

Legenda:
1 Iterace
2 Selekce
3 Sekvence

24

ODPOVĚĎ NA LOGICKOU HÁDANKU

- Vraždu spáchal Eduard a lže Adam a Cyril.

KURZ ALGORITMIZACE A PROGRAMOVÁNÍ V JAZYCE C

Lekce č. 4: Programovací jazyk C

Bc. Radek Libovický

LOGICKÁ HÁDANKA

- V hořícím domě je skupina lidí. Chtějí se dostat za každou cenu ven, neboť dům za 12 minut spadne. Musí proběhnout chodbou která je celá v plamenech. Pokud skrz ni chce někdo projít, tak musí mít u sebe hasící přístroj a plameny alespoň trochu krotit. Problém je, že přátelé mají jen jeden.
  Chodbou mohou jít zároveň maximálně dva lidé. Pak se někdo musí vrátit s přístrojem a mohou jít další dva.
  Mezi přáteli je jeden hasič, který se v plamenech pohybuje běžně, a tak dokáže chodbou proběhnout během minuty. Student proběhne za minuty dvě. Pak je tam ještě jeden důchodce, kterému to trvá čtyři minuty, a ožrala, který se bude chodbou motat pět minut. Pokud jde dvojice, pohybuje se rychlostí pomalejšího.
  Jak budou postupovat, aby se dostali ven do 12 minut, než dům spadne?

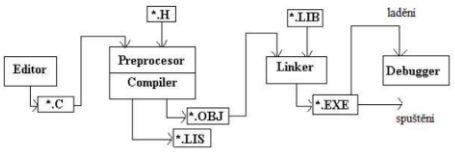
Odpověď na konci prezentace

PROGRAMOVACÍ JAZYK C

- **Vznik** - 70. léta minulého století.
- **Nízkoúrovňový** - poskytuje malou nebo žádnou abstrakci od toho, jak funguje procesor počítače. Rozdíl mezi daným programovacím jazykem a strojovými instrukcemi procesoru je minimální.
- **Kompilovaný** – před spuštěním musí být zdrojový kód nejprve zkompilován do strojového kódu, který je již spustitelný.
- **Strukturovaný a procedurální –** při tvorbě algoritmu či programu se postupuje shora dolů, používají se jenom tři základní řídicí struktury (sekvence, selekce, iterace)
- **Nespecializovaný** - není specializovaný pouze na jednu oblast používání
- **Efektivní** - velmi vysoká efektivita přeloženého kódu (téměř srovnatelný s Assemblerem)
- **Case sensitive –** rozlišuje velká a malá písmena (promenna, Promenna, PROMENNA – pokaždé se jedná o jinou proměnnou)

PRINCIP ZPRACOVÁNÍ PROGRAMU

Od zdrojového kódu po spustitelný program (strojový kód)



PRINCIP ZPRACOVÁNÍ PROGRAMU

- **Zdrojový kód** - zápis textu počítačového programu v některém programovacím jazyce
- **Editor** – v něm se zdrojový kód vytváří
- **Preprocesor** – součást compileru, která předzpracovává zdrojový kód tak, aby compiler měl jednodušší práci (např. odstraňování komentářů a mezer, či vložení hlavičkových souborů (.H), atd.)
- **Compiler (překladač, kompilátor)** – vytváří ze zdrojového kódu téměř hotový program, kterému ještě chybí přiřadit absolutní adresy pro proměnné a funkce. Dále odesílá zprávu o nalezených chybách
- **Linker –** přidělí relativnímu kódu absolutní adresy. Vzniká spustitelný soubor (.exe)
- **Debugger –** ladící program sloužící pro hledání a následné odstraňování chyb v programu
- **Strojový kód** - posloupnost strojových instrukcí prováděných procesorem počítače

ODPOVĚĎ NA LOGICKOU HÁDANKU

• Hasič a student ven	2min
Hasič zpět	1min
Ožrala a důchodce ven	5min
Student zpět	2min
Hasič a student ven	2min

ZDROJE

- ROUBAL, Pavel. *Informatika a výpočetní technika pro střední školy: praktická učebnice*. 1. vyd. Brno: Computer Press, 2010, 112 s. ISBN 9788025132272.
- HEROUT, Pavel. *Učebnice jazyka C*. 4. přeprac. vyd. České Budějovice: Kopp, 2004, 271, viii s. ISBN 80-7232-220-6.

KURZ ALGORITMIZACE A PROGRAMOVÁNÍ V JAZYCE C

Lekce č. 5: Hello world!

Bc. Radek Libovický

LOGICKÁ HÁDANKA

- Máte neomezené množství vody. Jak naberete 6 litrů, když máte jen 4- a 9-litrovou nádobu.

Odpověď na konci prezentace

PŘÍPRAVA VÝVOJOVÉHO PROSTŘEDÍ

- Tento návod je pro vývojové prostředí CodeBlocks, nicméně princip je stejný pro všechna prostředí

1. Zapnout CodeBlocks
2. File – New – Project
3. Vybrat console application (v tomto kurzu se budou vytvářet pouze konzolové aplikace)
4. Vyplnit název souboru a umístění souboru
5. Po levé straně v sekci Management otevřít Projects – Sources – main.c

Pokud se vše povedlo, můžete začít s programováním

ZAČÍNÁME PROGRAMOVAT

- Pokud je editor připravený, je na čase vytvořit Váš první program
- **Zadání**:
  - Vytvořit program, který vypíše: „Hello world!“

HLAVIČKOVÉ SOUBORY

- Pro zpřístupnění konkrétních funkcí je třeba k programu připojit knihovny
- Pro začátek bohatě postačí knihovna stdio.h (standard input/output)
- Připojení knihovny:
- `#include <stdio.h>`
- Po připojení všech potřebných knihoven se může začít psát hlavní funkce programu

37

HLAVNÍ FUNKCE PROGRAMU

Hlavní funkce programu:

```
int main()
{
    return 0;
}
```

- Hlavní funkce programu má vždy název **main**
- Je to funkce, která se spouští jako první
- int před **main** znamená, že hlavní funkce je datového typu integer (neboli celé číslo)
- Závorky za main jsou pro vstupní parametry funkce. Hlavní funkce nemusí mít vstupní parametry (v tomto případě by závorky zůstaly prázdné)
- return 0; vyjadřuje jakou hodnotu hlavní funkce vrací (celé číslo - 0)
- Středníkem se ukončují příkazy

38

VÝSTUP

- **Funkce pro výpis do konzole**:

```
printf("Libovolný text");
```

- Tato funkce vrací celé číslo, které je rovno počtu vypsaných znaků do konzole

39

BÍLÉ ZNAKY

- Bílé znaky:
  Tyto znaky nejsou vidět, ale jsou velmi důležité (nový řádek, tab, konec souboru, návrat na začátek řádku,…)

Nový řádek	\n
Pípnutí	\a
Návrat na začátek řádku	\r
Tabulátor	\t
Posun doleva	\b

Např.: `printf("\n \a \t");`

40

KOMENTÁŘE

- **Komentář** – slouží pro zpřehlednění kódu, či zapsání nějaké myšlenky
- **Jednořádkové**:

```
//nějaká poznámka
```

- **Víceřádkové:**

```
/*    nějaká poznámka
      nějaká poznámka
      nějaká poznámka
      nějaká poznámka */
```

41

HELLO WORLD

```
#include <stdio.h> //připojení hlavičkového souboru

int main() //hlavní funkce
{ //začátek těla hlavní funkce
    printf("Hello world!\n"); /*funkce pro výpis na konzoli*/
    return 0; //hlavní funkce vrací hodnotu 0
} //konec těla hlavní funkce
```

42

ODPOVĚĎ NA LOGICKOU HÁDANKU

- Naplníme 9-litrovou nádobu a odlijeme z ní dvakrát 4 litry do vedlejší nádoby. Zůstane nám tedy 1 litr a ten přelijeme do 4-litrové. Znovu naplníme 9-litrovou a doplníme obsah 4-litrové (jen 3 litry). Tím pádem nám v 9-litrové zůstane 6 litrů vody.

KURZ ALGORITMIZACE A PROGRAMOVÁNÍ V JAZYCE C

Lekce č. 6: Proměnné, jejich typy a práce s nimi

Bc. Radek Libovický

LOGICKÁ HÁDANKA

- Těžce nemocný starý pán byl u doktora a ten mu předepsal 2 červené a 2 modré prášky. Přesně o půlnoci si musí vzít právě jeden prášek modrý a právě jeden červený. Pokud si je nevezme, jeho srdce to nevydrží. Pokud si jich vezme víc, zemře na předávkování.
  Protože byl z cesty k doktorovi unavený, šel si hned po příchodu lehnout (takže si prášky nenachystal). Spal až do půlnoci, kdy ho probudil budík. Ke svému zděšení zjistil, že došlo k výpadku elektřiny a všude je naprostá tma. Zašmátral v krabičce na léky a zjistil, že ty 4 prášky od sebe hmatem nerozezná a žádná možnost k posvícení také neexistuje.
  Zachoval chladnou hlavu, zamyslel se a celou situaci v mžiku bravurně vyřešil. Jak?

Odpověď na konci prezentace

TVORBA PROMĚNNÝCH

- **Jednoduché proměnné**:

Celé číslo	`int`
Reálné číslo	`float`
Znak	`char`

- **Vytvoření proměnných**:

```
int pocet_lidi;
float objem_valce;
char pismeno;
```


OPERÁTORY

- **Operátory**:

sčítání	+
odečítání	-
násobení	*
dělení	/
dělení modulo (zbytek po dělení)	%

OPERACE S PROMĚNNÝMI

Příklady:	**Hodnoty**:
`int a;`	a nemá přiřazenou hodnotu
`int b = 7;`	b je 7
`a = 4;`	a je 4
`a = b;`	a je 7
`a = a + b;`	a je 14
`a = a * (a - b - 5);`	a je 28
`a+=b;`	Zkrácená forma zápisu (`a = a + b;`) a je 35
`a-=b;`	a je 28
`a%=b;`	a je 0
`a++;`	a je 1 – inkrementace `a = a + 1;`
`a--;`	A je 0 – dekrementace `a = a - 1;`

OPERACE S PROMĚNNÝMI

Příklady:	**Hodnoty**:
`char a;`	a nemá přiřazenou hodnotu
`char c = '5';`	c je znak 5 – pozor není to číslo 5
`int d;`	d nemá přiřazenou hodnotu
`a = 'a';`	a je znak a
`a = a - 32;`	a je znak A //posun v ASCII tabulce
`d = a - 32;`	d je číslo 65 //ASCII hodnota pro a je 97
`c = c - 'a';`	c je znak ď

49

VSTUP

- **Funkce pro načítání dat z klávesnice:**

```
scanf("%d", &promenna);
```

%d – pro celá čísla
%f – pro reálná čísla
%c – pro znaky

- Tato funkce vrací celé číslo, které je rovno počtu správně načtených položek

50

ZADÁNÍ PROGRAMU

- **Zadání**:

  Program, který načte 3 reálná čísla a vypíše jejich aritmetický průměr

51

ŘEŠENÍ

```
#include <stdio.h>

int main()
{
//vytvoření proměnných
    float a;
    float b;
    float c;
    float prumer;

    printf("Zadejte 3 reálná čísla\n");
//naplnění proměnných
    scanf("%f",&a);
    scanf("%f",&b);
    scanf("%f",&c);
    prumer = (a+b+c)/3; //výpočet aritmetického průměru
    printf("Aritmetický průměr zadaných 3 čísel je :%f",prumer); //výpis textového řetězce a proměnné průměr
    return 0;
}
```

52

ODPOVĚĎ NA LOGICKOU HÁDANKU

- Od každého ze 4 prášků si vzal půlku.

KURZ ALGORITMIZACE A PROGRAMOVÁNÍ V JAZYCE C

Lekce č. 7: Podmínky (if, switch)

Bc. Radek Libovický

54

LOGICKÁ HÁDANKA

- Obžalovaný stojí u soudu a je odsouzen k trestu smrti. Jeho obhájci se však nakonec podaří vyjednat následující podmínku:
  "Můžete říci jednu oznamovací větu na svou obranu. Pokud nám zalžete, tak Vás oběsíme. Pokud řeknete pravdu, tak Vás zastřelíme."
  Co obžalovaný řekne, aby si zachránil život?

Odpověď na konci prezentace

55

TVORBA LOGICKÉHO VÝRAZU

- **Logické výrazy mohou obsahovat tyto operátory:**

==	rovnost	(c=='A')
!=	nerovnost	(c!=7)
<	menší	(c<7)
>	větší	(c>6)
>=	větší nebo rovno	(c>=3)
<=	menší nebo rovno	(c<=12)
!	negace výrazu	(!(c==14))

- **Logické výrazy lze skládat pomocí:**

&&	AND – a současně	((c>6) && !(c==12))
||	OR – nebo	((c<=6) || (c==28))

56

PODMÍNKA - IF

- **Větvení programu:**

Používá se v situaci, kdy je potřeba, aby se program choval rozdílně v závislosti na vyhodnocené podmínce

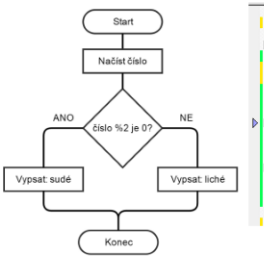
Zápis podmínky if:

```
//část programu, který vypisuje větší číslo ze dvou načtených
if(a>b); //vyhodnocení podmínky (v tomto případě a>b)
{ //blok příkazů pro situaci, kdy je a větší než b
    printf("%d",a); //výpis proměnné a, protože je větší než b
} //konec bloku
else // nepovinná část podmínky if s blokem, který se má vykonat v případě nesplněné podmínky
{//blok příkazů pro situaci, kdy je b větší než a
    printf("%d",b); //výpis proměnné b, protože je větší než a
} //konec bloku
```

57

DALŠÍ PŘÍKLAD PODMÍNKY - IF

Program na rozeznání sudého či lichého čísla:




```
#include <stdio.h>

int main()
{
    int cislo;

    printf("Zadejte cislo\n");
    scanf("%d",&cislo);

    if(cislo%2==0)
    {
        printf("Zadane cislo je sude\n");
    }
    else
    {
        printf("Zadane cislo je liche\n");
    }
    return 0;
}
```

58

VÍCENÁSOBNÁ PODMÍNKA SWITCH

- Používá se v situaci, když se program rozsáhle větví a kód by se využíváním podmínky if stal velmi nepřehledným
- **Zápis vícenásobné podmínky switch:**

```
switch (cislo)
{
    case 1:
        printf("Varianta 1");
    break;
    case 2:
        printf("Varianta 2");
    break;
    case 3:
        printf("Varianta 3");
    break;

    default:
        printf("Zadna varianta");
    break;
}
```

Switch přijímá jako vstupní parametr zadanou proměnnou. Dle hodnoty v proměnné proběhne skok na case, který má shodnou hodnotu. Pokud takový case není, proběhne skok na default.

Příkaz break způsobí skok ven z těla switche a program pokračuje dál.

59

ODPOVĚĎ NA LOGICKOU HÁDANKU

- "Určitě mě oběsíte"

KURZ ALGORITMIZACE A PROGRAMOVÁNÍ V JAZYCE C

Lekce č. 8: Cykly (while, do while, for)

Bc. Radek Libovický

61

LOGICKÁ HÁDANKA

- Stojíte u tří vypínačů. Víte, že patří ke třem žárovkám, které jsou v místnosti, kam vede dlouhá a klikatá chodba - tzn. že ze svého místa vůbec nemůžete vidět, zda některá svítí nebo ne. Všechny tři vypínače jsou nyní v poloze vypnuto.
  S vypínači můžete manipulovat jak chcete, pak jednou projít chodbou a podívat se do místnosti. Tam musíte říci, který vypínač je od které žárovky.
  Jak na to?

Odpověď na konci prezentace

62

CYKLUS WHILE

- **Cyklus while:**
  Nejdříve se vyhodnotí podmínka, pokud je splněna, vykoná se tělo cyklu a vrátí se zpět na vyhodnocování podmínky. Takto neustále dokola, až do chvíle, kdy se podmínka vyhodnotí jako nepravdivá.

Zápis cyklu while:

```
while(a>b)//vyhodnocení podmínky (v tomto případě a>b)
{ //začátek těla cyklu
    i++;
    b++;
} //konec těla cyklu
printf("Hodnota B se musela %d krát zvětšit, aby se B=A",i);
```

63

CYKLUS DO WHILE

- **Cyklus do while:**
  Jediný rozdíl od cyklu while je ten, že se nejdříve provede tělo cyklu a teprve poté se vyhodnocuje podmínka. Znamená to tedy, že každý cyklus do while proběhne minimálně jednou.

Zápis cyklu do while:

```
do//začátek cyklu bez vyhodnocení podmínky
{ //začátek těla cyklu
    i++;
    b++;
} while(a>b); //konec těla cyklu a vyhodnocení podmínky
```

64

CYKLUS FOR

- **Cyklus for:**
  Tento cyklus se používá v situacích, kdy autor přesně ví, kolikrát se má cyklus zopakovat.

Zápis cyklu for:

```
for(i=0;i<10;i++)//(start; podmínka; krok)
{
      //tělo cyklu - konkrétně proběhne 10krát
}
```

Start – nastavení proměnné při prvním vstupu do cyklu (i=0)
Podmínka – stejná jako u while – vyhodnocuje se, zda se bude cyklus opakovat (i<10)
Krok – po každém průchodu těla se proměnná i zvýší (i++)

65

ODPOVĚĎ NA LOGICKOU HÁDANKU

- Zapnete první vypínač, chvíli počkáte. Poté vypnete první a zapnete druhý. Jděte do místnosti a jedna žárovka bude svítit, jedna bude horká a poslední zhasnutá.

KURZ ALGORITMIZACE A PROGRAMOVÁNÍ V JAZYCE C

Lekce č. 9: Pole a práce s nimi

Bc. Radek Libovický

67

LOGICKÁ HÁDANKA

Dostali jste devět na pohled stejných mincí, jedna je falešná - lehčí. K dispozici máte rovnoramenné váhy. Jak na dvě vážení zjistíte, která je falešná?

Odpověď na konci prezentace

68

POLE

- **Pole se používá:**

  Pokud je potřeba hromadně definovat nějaký počet proměnných určitého typu.

Zápis pro vytvoření pole:

```
char pole_znaku[10];
int pole_cisel[5];
```

Index pole:

- Definování pole je shodné jako u klasických proměnných, akorát parametr uvedený v hranatých závorkách udává, kolik znaků či čísel bude pole obsahovat
- Pro určení položky v poli se používají indexy. Pole začíná vždy indexem 0!

69

OPERACE S POLI

Příklady:	**Hodnoty:**
`int a[3]={1,2,3};`	a je pole celých čísel o velikosti 3 s hodnotami 1, 2, 3
`int b[5];`	b je pole celých čísel o velikosti 5 s nedefinovanými hodnotami
`int i = 0;`	i je 0
`char c[2]={'Q','w'};`	c je pole znaků (řetězec) o velikosti 2 s hodnotami Q,w
`a[i] = 4;`	a na indexu 0 zapíše hodnotu 4, tím se smaže hodnota 1, která tam byla předtím
`i++;`	Inkrementace i, i je 1
`a[i+1] = 27;`	a je na indexu 2 nahradí hodnotu 3 za hodnotu 27

70

ZADÁNÍ PROGRAMU

- **Zadání**:

  Program, který bude načítat čísla tak dlouho, dokud se nezadá 0. Poté vypíše 5 nejvyšších zadaných čísel v pořadí od největšího po nejmenší.

71

ŘEŠENÍ

```
#include <stdio.h>

int main()
{
    int PetNejvetsich[5]= {0,0,0,0,0};
    int nactene;
    int pomocna;
    int i;
    printf("Zadavejte cisla k razeni\nzadavani cisel se ukonci znakem \"0\"\n\n");
    while(1)
    {
        scanf("%d",&nactene);
        if(nactene==0)
        {
            break;
        }
        for(i=0; i<5; i++)
        {
            if(nactene>PetNejvetsich[i])
            {
                pomocna=PetNejvetsich[i];
                PetNejvetsich[i]=nactene;
                nactene=pomocna;
            }
        }
    }
    printf("\nPada 5m:\n");
    for(i=0; i<5; i++)
    {
        printf("\n %d",PetNejvetsich[i]);
    }
    return 0;
}
```

72

ODPOVĚĎ NA LOGICKOU HÁDANKU

- Tři mince vlevo, tři vpravo, tři jsou vedle. Váhy jdou buď do leva, doprava nebo zůstanou v rovnováze.
- Už se možnosti omezili na tři mince. Jedna doleva, jedna doprava, jedna mimo....

KURZ ALGORITMIZACE A PROGRAMOVÁNÍ V JAZYCE C

Lekce č. 10: Tvorba vlastních funkcí

Bc. Radek Libovický

LOGICKÁ HÁDANKA

Stojíte před třemi dveřmi: za jedněmi je nové auto, za zbývajícíma dvěma je koza.

Máte právo jedny dveře otevřít a získat, co za nimi je. Vy samozřejmě chcete auto.

Když si vyberete své dveře, tak ještě než je definitivně otevřete, moderátor (který ví, kde je auto) vždy otevře jedny ze zbývajících dveří a ukáže, že je tam koza. Nyní Vám dá šanci svou volbu změnit.

Stojíte tedy před dvěmi zavřenými dveřmi. Změníte svou volbu?

Odpověď na konci prezentace

FUNKCE

- **Funkce**:

  Hlavičkové soubory nabízejí využívat mnoho předdefinovaných funkcí, avšak v případě potřeby si programátor může nějakou funkci vytvořit sám.

  Při volání funkce program skočí do těla funkce a po jejím provedení se vrátí za místo, kde byla funkce volána. V situaci, že je daná funkce volána vícekrát, je šetřena nejen paměť, ale i práce daného programátora.

  Funkce vždy vrací nějakou hodnotu, pokud není typu void (prázdný).

TVORBA VLASTNÍ FUNKCE

Zápis funkce:

```
int obsah_ctverce(int a)
{
    a=a*a;
    return a;
}
```

1. U funkce musí být zvolen konkrétní datový typ, nebo nebude vracet žádnou hodnotu a bude obsahovat na začátku klíčové slovo void
2. Zvolit libovolný název funkce (bez mezer)
3. Do závorek definovat vstupní parametry (proměnné, které budou přijímány z jiné funkce)
4. Tělo funkce
5. Return vrací nějakou hodnotu, která je stejného datového typu jako samotná funkce
6. Volání funkce je název fce a v závorkách jsou hodnoty, které se do funkce odešlou. Konkrétní volání fce může vypadat takto: `obsah = obsah_ctverce(strana);`

REKURZIVNÍ FUNKCE

- **Rekurzivní funkce:**
  Funkce, která volá samu sebe. Využití například u výpočtu faktoriálu.

ODPOVĚĎ NA LOGICKOU HÁDANKU

- Lepší je dveře změnit.

  Vyhrajete vždy, pokud jste si napoprvé vybrali špatné dveře (2 ze 3).

  Pokud dveře nezměníte, tak vyhrajete, jen když jste si napoprvé vybrali dveře správné (1 ze 3).

KURZ ALGORITMIZACE A PROGRAMOVÁNÍ V JAZYCE C

Lekce č. 11: Práce se soubory

Bc. Radek Libovický

LOGICKÁ HÁDANKA

Jedna topinka se smaží deset minut - pět minut z každé strany. Na pánev se vejdou dva chleby vedle sebe. Za jak dlouho nejrychleji osmažíte na jedné pánvi tři topinky?

Odpověď na konci prezentace

DATOVÝ TYP FILE

- **Datový typ FILE**:
  Slouží pro práci se souborem

Zápis definování datové proměnné:

FILE *fr; pro čtení ze souboru (anj. read)

FILE *fw; pro zápis do souboru (anj. write)

OTEVŘENÍ SOUBORU

- Když jsou již nadefinované proměnné pro práci se soubory, je nutné k nim přiřadit adresu souboru a soubor otevřít

Zápis pro otevření souboru:

```
fr=fopen("adresa_souboru","r");
fw=fopen("adresa_souboru","w");
```

- Funkce fopen má dva parametry, první je celý název souboru včetně adresy, v případě, že se nachází soubor ve shodném adresáři jako program, pak stačí jen název souboru. Druhý parametr označuje, co se se souborem bude dělat (r – číst, w – zapisovat)
- Pokud se při otevírání zjistí, že cílový soubor pro zápis neexistuje, bude touto funkcí vytvořen

VÝPIS A ZÁPIS DO SOUBORU

- Pro zapisování a vypisování ze souboru dostatečně postačí:

Výpis: `znak=getc(fr); //načte znak ze souboru pro čtení`

Zápis: `putc(znak,fw); //zapíše znak do souboru pro zápis`

Tyto funkce se používají ve spojení s cyklem while – Např. načítej znaky tak dlouho, dokud znak nebude EOF (end of file)

ZAVŘENÍ SOUBORU

- Když už je veškerá práce se soubory hotová, je ještě nutné používané soubory uzavřít

Zápis pro uzavření souboru:

```
fclose(fr); // uzavření souboru, ze kterého se četlo
fclose(fw); // uzavření souboru, do kterého se zapisovalo
```

Pokud programátor zapomene tyto soubory uzavřít, mohly by se vyskytnout komplikace při budoucí práci s těmito soubory.

85

ODPOVĚĎ NA LOGICKOU HÁDANKU

- Za 15 minut. Po pěti minutách jednu topinku sundáte a druhou otočíte. Po deseti minutách je jedna hotová a dvě je třeba osmažit ještě z jedné strany.

Příloha 2: Otázka, která studentům činila největší potíže

5 Jaký bude výstup níže uvedeného programu?

Body: 1

```
#include <stdio.h>

int main()
{
   int i;
   int a=7;
   int b=1;

   while(a>b)
   {
      for(i=0;i<=3;i++)
      {
        a=a-1;
        printf("A");
        i++;
      }
      printf("_");
   }
   return 0;
}
```

Odpověď:

AA_AA_AA_

✓

Vložte komentář, nebo přepište body

Správná odpověď

Bodový zisk: 1/1.

Příloha 3: ASCII tabulka

REGULAR ASCII CHART (character codes 0 – 127)

000d	00h	␀	(nul)	016d	10h	►	(dle)	032d	20h	␣	048d	30h	0	064d	40h	@	080d	50h	P	096d	60h	`	112d	70h	p
001d	01h	☺	(soh)	017d	11h	◄	(dc1)	033d	21h	!	049d	31h	1	065d	41h	A	081d	51h	Q	097d	61h	a	113d	71h	q
002d	02h	☻	(stx)	018d	12h	↕	(dc2)	034d	22h	"	050d	32h	2	066d	42h	B	082d	52h	R	098d	62h	b	114d	72h	r
003d	03h	♥	(etx)	019d	13h	‼	(dc3)	035d	23h	#	051d	33h	3	067d	43h	C	083d	53h	S	099d	63h	c	115d	73h	s
004d	04h	♦	(eot)	020d	14h	¶	(dc4)	036d	24h	$	052d	34h	4	068d	44h	D	084d	54h	T	100d	64h	d	116d	74h	t
005d	05h	♣	(enq)	021d	15h	§	(nak)	037d	25h	%	053d	35h	5	069d	45h	E	085d	55h	U	101d	65h	e	117d	75h	u
006d	06h	♠	(ack)	022d	16h	▬	(syn)	038d	26h	&	054d	36h	6	070d	46h	F	086d	56h	V	102d	66h	f	118d	76h	v
007d	07h	•	(bel)	023d	17h	↨	(etb)	039d	27h	'	055d	37h	7	071d	47h	G	087d	57h	W	103d	67h	g	119d	77h	w
008d	08h	◘	(bs)	024d	18h	↑	(can)	040d	28h	(	056d	38h	8	072d	48h	H	088d	58h	X	104d	68h	h	120d	78h	x
009d	09h		(tab)	025d	19h	↓	(em)	041d	29h	)	057d	39h	9	073d	49h	I	089d	59h	Y	105d	69h	i	121d	79h	y
010d	0Ah	◙	(lf)	026d	1Ah	→	(eof)	042d	2Ah	*	058d	3Ah	:	074d	4Ah	J	090d	5Ah	Z	106d	6Ah	j	122d	7Ah	z
011d	0Bh	♂	(vt)	027d	1Bh	←	(esc)	043d	2Bh	+	059d	3Bh	;	075d	4Bh	K	091d	5Bh	[	107d	6Bh	k	123d	7Bh	{
012d	0Ch		(np)	028d	1Ch	└	(fs)	044d	2Ch	,	060d	3Ch	<	076d	4Ch	L	092d	5Ch	\	108d	6Ch	l	124d	7Ch	|
013d	0Dh	♪	(cr)	029d	1Dh	↔	(gs)	045d	2Dh	-	061d	3Dh	=	077d	4Dh	M	093d	5Dh	]	109d	6Dh	m	125d	7Dh	}
014d	0Eh	♫	(so)	030d	1Eh	▲	(rs)	046d	2Eh	.	062d	3Eh	>	078d	4Eh	N	094d	5Eh	^	110d	6Eh	n	126d	7Eh	~
015d	0Fh	☼	(si)	031d	1Fh	▼	(us)	047d	2Fh	/	063d	3Fh	?	079d	4Fh	O	095d	5Fh	_	111d	6Fh	o	127d	7Fh	⌂

EXTENDED ASCII CHART (character codes 128 – 255) LATIN1/CP1252

128d	80h	€		144d	90h			160d	A0h	\	176d	B0h	°	192d	C0h	À	208d	D0h	Ð	224d	E0h	à	240d	F0h	ð
129d	81h			145d	91h	‘		161d	A1h	¡	177d	B1h	±	193d	C1h	Á	209d	D1h	Ñ	225d	E1h	á	241d	F1h	ñ
130d	82h	‚		146d	92h	’		162d	A2h	¢	178d	B2h	²	194d	C2h	Â	210d	D2h	Ò	226d	E2h	â	242d	F2h	ò
131d	83h	ƒ		147d	93h	“		163d	A3h	£	179d	B3h	³	195d	C3h	Ã	211d	D3h	Ó	227d	E3h	ã	243d	F3h	ó
132d	84h	„		148d	94h	”		164d	A4h	¤	180d	B4h	´	196d	C4h	Ä	212d	D4h	Ô	228d	E4h	ä	244d	F4h	ô
133d	85h	…		149d	95h	•		165d	A5h	¥	181d	B5h	µ	197d	C5h	Å	213d	D5h	Õ	229d	E5h	å	245d	F5h	õ
134d	86h	†		150d	96h	–		166d	A6h	¦	182d	B6h	¶	198d	C6h	Æ	214d	D6h	Ö	230d	E6h	æ	246d	F6h	ö
135d	87h	‡		151d	97h	—		167d	A7h	§	183d	B7h	·	199d	C7h	Ç	215d	D7h	×	231d	E7h	ç	247d	F7h	÷
136d	88h	ˆ		152d	98h	˜		168d	A8h	¨	184d	B8h	¸	200d	C8h	È	216d	D8h	Ø	232d	E8h	è	248d	F8h	ø
137d	89h	‰		153d	99h	™		169d	A9h	©	185d	B9h	¹	201d	C9h	É	217d	D9h	Ù	233d	E9h	é	249d	F9h	ù
138d	8Ah	Š		154d	9Ah	š		170d	AAh	ª	186d	BAh	º	202d	CAh	Ê	218d	DAh	Ú	234d	EAh	ê	250d	FAh	ú
139d	8Bh	‹		155d	9Bh	›		171d	ABh	«	187d	BBh	»	203d	CBh	Ë	219d	DBh	Û	235d	EBh	ë	251d	FBh	û
140d	8Ch	Œ		156d	9Ch	œ		172d	ACh	¬	188d	BCh	¼	204d	CCh	Ì	220d	DCh	Ü	236d	ECh	ì	252d	FCh	ü
141d	8Dh			157d	9Dh			173d	ADh		189d	BDh	½	205d	CDh	Í	221d	DDh	Ý	237d	EDh	í	253d	FDh	ý
142d	8Eh	Ž		158d	9Eh	ž		174d	AEh	®	190d	BEh	¾	206d	CEh	Î	222d	DEh	Þ	238d	EEh	î	254d	FEh	þ
143d	8Fh			159d	9Fh	Ÿ		175d	AFh	¯	191d	BFh	¿	207d	CFh	Ï	223d	DFh	ß	239d	EFh	ï	255d	FFh	ÿ

Hexadecimal to Binary

0	0000	4	0100	8	1000	C	1100
1	0001	5	0101	9	1001	D	1101
2	0010	6	0110	A	1010	E	1110
3	0011	7	0111	B	1011	F	1111

Groups of ASCII-Code in Binary

Bit 6	Bit 5	Group
0	0	Control Characters
0	1	Digits and Punctuation
1	0	Upper Case and Special
1	1	Lower Case and Special

Zdroj: Michael Goerz, 2009