Uvod v Računalništvo

Zbrana in popravljena poročila predavanj

Luka Šveigl, 63200301

1. PAMETNA SPECIALIZACIJA IN ANALIZA ZAHTEV:

Na predavanju **Pametna specializacija in analiza zahtev** sem izvedel veliko novega. Govorili smo o **pametnih aplikacijah, področjih pametne specializacije, o platnih poslovnih modelov in o življenskem ciklu programske opreme**.

Povedali smo, da **pametne aplikacije običajno temeljijo na internetu stvari, umetni inteligenci, računalništvu v oblaku** in podobnih tehnologijah, **računalništvo v megli** pa je skupno ime za vse te tehnologije.

Povedali smo, da so **področja pametne specializacije**:

- **Zdravje** (inteligentna infrastruktura, interakcija med vsemi stranmi v zdravstvu, nosljive naprave itd.)
- **Gospodarstvo** (umetna inteligenca kot finančni svetovalci)
- **Mobilnost** (Tesla, samovozeči avtomobili, strojno učenje v oblaku)
- **Javne storitve** (Javni avtobusi brez voznika, javne podatkovne baze)
- **Energija in trajnost** (Pametni domovi, optimizacija porabe energije, prilagajanje temperature, itd.)
- **Novice, mediji in zabava** (Pametni asistenti (Siri, Alexa, itd.), glasovno aktiviranje, nevarnosti vdora v zasebnost)
- **Industirja** (Analiza uporabe strojev, pametni vozički, boljša varnost v tovarnah)
- **Turizem** (Umetna inteligenca za priporočanje nastanitev, napovedovanje cen)
- **Izobraževanje** (Deljenje vsebin med uporabniki, kreiranje vprašanj iz učbenikov)

Pogledali smo si tudi **primere različnih pametnih aplikacij in okolji**, kot so **DECENTER** in **ENTICE** ter aplikacije za upravljanje z gradbiščem, zaznavanje video tokov itd., ter COPSYS.

Nato smo si pogledali tudi **analizo zahtev pri razvoju programov**. Začeli smo z **definicijo problema**, katere namen je **natančno določiti cilje, obseg in obliko dela ter zagotoviti upoštevanje zahtev glede podpore**. Pri definiciji problema moramo **dobro dokumentirati uporabnikove zahteve, določiti okolje v katerem bo aplikacija delovala in najti najširše soglasje o zahtevah**.

Povedali smo tudi, da lahko **uporabnikove zahteve razvrstimo v 3 kategorije:**

- **Funkcionalnost aplikacije**
- **Omejitve**
- **Podpora**

Povedali smo tudi, da je **rezultat definicje problema** dokument **Specifikacije zahtev uporabnika**, ki je **načrt za upravljanje projekta**, verifikacijo in zagotavljanje kvalitete.

Nato smo si pogledali tudi kaj je **analiza zahtev**. Povedali smo, da je njen **namen izraziti zahteve uporabnika v smislu, ki je bližje programski opremi** s stanja realizacije. Pri analizi zahtev

izdelamo implementacijsko neodvisen logični model, iz katerega lahko izpeljemo specifične zahteve programske opreme. Ta specifikacija mora vsebovati:

- Funkcionalne zahteve
- Izvedbene zahteve
- Zahteve vmesnikov
- Uporabniški vmesnik
- Zahteve virov
- Verifikacijske zahteve
- Zahteve po dokumentaciji
- Varnostne zahteve
- Zahteve po prenosljivosti

Povedali smo tudi, da so **atributi zahtev programske opreme**:

- Identifikacijska koda
- Potrebnost
- Stabilnost
- Preverljivost
- Prioriteta
- Vir

Rezultat analize zahtev pa je **Specifikacija zahtev s stališča programske opreme** in plan za naslednjo fazo. Naslednej faze so:

- Načrtovanje
- Implementacija
- Testiranje
- Pilot
- Vzdrževanje

Predavanje mi je bilo zelo zanimivo, saj sem se **naučil veliko o novih tehnologijah in njihovih uporabah v modernih pametnih aplikacijah in okoljih**. Zelo poučen se mi je zdel tudi **pregled področji, kjer se takšne aplikacije uporabljajo**. Menim, da je najpomembnejše, da smo ugotovili, da **preden začnemo programirati**, moramo poznati **prednosti in slabosti novih tehnologij**, **analiziramo poslovne priložnosti, zapišemo in analiziramo želje ter zahteve uporabnikov in izdelamo načrt**.

2. RAČUNALNIK KOT STROJ:

Na predavanju **Računalnik kot stroj** sem izvedel veliko novega o **zgodovini in zgradbi računalnika**. Govorili smo o **matematičnih osnovah, na katerih sloni delovanje računalnika in pa o različnih modelih računalnikov**, ki so nastali skozi zgodovino. Povedali smo tudi veliko **o mehanskih računalih, mehanskih računalnikih in raznih elektromehanskih in elektronskih računalnikih**, spoznal pa sem tudi delovanje prvih modelov računalnika. Govorili smo tudi o glavnih komponentah računalnikov.

Najprej smo na predavanju govorili o **zgodovini računalništva**, ki smo jo razdelili na različna obdobja. Povedali smo, da je **zgodnje obdobje trajalo do leta 1940**, v njem pa so se razvila **različna mehanska računala, avtomatske statve, ki so bile programibilne z luknjastimi karticami in prvi programibilni stroj**. Takrat so nastali tudi Mehanski računalniki, ena prvih sta bila diferenčni in alnalitični stroj, katera je razvil **Charles Babbage**.

Drugo obdobje, katerega smo obravnavali je **Rojstvo računalnikov, ki je trajalo od leta 1940 do 1950.** Takrat je **Konrad Zuse ustvaril Z1, ki je bil prvi programibilni računalnik**, kasneje pa je naredil tudi model Z3. V tem obdobju so na **Harvardu ustvarili tudi računalnik Harvard Mark1, ki je bil prvi računalnik z dvojiškim sistemom**. V tem obdobju so delovali tudi Atanasoff in **Alan Turing**, slednji je razbil nemško šifro Enigma. V tem obdobju je deloval tudi **John von Neumann, ki je razvil arhitekturo računalnika, ki je v uporabi še danes**.

Nato smo govorili o **modernem obdobju, ki se je začelo po letu 1950 in se deli na več generacij**. V **prvi generaciji** so bile v uporabi **elektronke in luknjaste kartice, v drugi generaciji so se razvili prvi diski, velikost se je manjšala, nastali pa so tudi prvi operacijski sistemi in visoko-nivojski programski jeziki (FORTRAN, COBOL).** V **tretji generaciji** so bila v **uporabi integrirana vezja, nastal pa je tudi prvi miniračunalnik**. V tej generaciji se je rodila industrija programske opreme, in s tem se je začela širiti uporaba računalnikov.**V četrti generaciji so nastali prvi mikroračunalniki,** v uporabo je prišel strojni jezik, **nastali so ikonični računalniki kot so ZX Spectrum in Commodore 64**. Nastala so **prva računalniška omrežja, prvi vgrajeni sistemi in prvi grafični uporabiški vmesniki**. Govorili smo tudi o **peti generaciji**, v kateri smo sedaj.

Povedali smo tudi nekaj o **razvoju stikal**, ki so zelo **pomembna računalniška komponenta** in govorili tudi o **Moorovem zakonu**, ki pravi, **da se št. elementov, ki jih lako vgradimo na določen čip podvoji vsakih 18-24 mesecev, pri čemer ostane cena nespremenjena**.

Nato smo govorili tudi o modelu računalnika. Najprej smo povedali, kaj je **hierarhija abstrakcij** funkcionalnih enot računalniškega sistema:

- Tranzistorji
- Vrata
- Vezja

Govorili smo tudi o **komponentah v von Neumannovi arhitekturi računalniškega sistema** in povedali, da vsebuje 4 glavne podsisteme:

- Pomnilnik
- Vhod / izhod
- Artitmetično-logična enota
- Krmilna enota -> Centralna procesna enota

S to arhitekturo se je rodil tudi **koncept programa, shranjenega v pomnilniku** in pa **zaporedno izvajanje ukazov.**

Nato smo si podrobneje pogledali **pomnilnik, ki je funkcijska enota za shranjevanje in branje podatkov**. Povedali smo, da **RAM deluje tako, da je čas dostopa do vseh celic enak, vsaka celica pa ima svoj naslov.** Povedali smo, da sta **2 osnovni operaciji v pomnilniku branje (fetch) in pisanje (store).** Povedali smo tudi, da sta za delovanje pomembna **MAR (Memory Address Register) in MDR (Memory Data Register).** Povedali smo tudi, da **dekodirnik pretvori vsebino MAR** v signal za specifično pomnilniško lokacijo.

Povedali smo tudi, **kaj je predpomnilnik in da je pomemben za odpravljanje Von Neumannovega ozkega grla, saj je hitrejši od RAM, ampak je zaradi tega tudi veliko dražji. Pomemben je tudi princip lokalnosti.**

Nato smo govorili tudi o **Vhodno-izhodnih napravah**, katere delimo na 2 skupini:

- Naprave za prenos informacij med CPE in pomnilnikom ter zunanjim svetom
- Pomožni pomnilnik za shranjevanje podatkov

Povedali smo tudi, kaj je **ALE (Aritmetično-logična enota) in kaj je njena vloga v delovanju CPE.** Povedali smo, da vsebuje vezja za aritmetiko in primerjanje ter logiko ter register.

Nato smo govorili tudi o **krmilni enoti, ki iz pomnilnika prebere naslednji ukaz, ki naj se izvede, ga dekodira in izvede (pošlje ustrezni enoti)**. Povedali smo, da je format ukaza v strojnem jeziku koda operacije in naslovi pomnilniških lokacij z operandi, govorili pa smo tudi o **naborih strojnih ukazov, bolj podrobno smo pogledali RISC in CISC arhitekturo.**

Na koncu smo govorili tudi o paralelnem procesiranju in modelih SIMD in MIMD.

Predavanje je bilo zelo obširno, a še vedno zanimivo saj je dobro poznati zgodovino računalništva.

3. ZAPIS PODATKOV IN LOGIČNI IZRAZI:

Na predavanju **Zapis podatkov in logični konstrukti** je bilo povedano, da so podatki lahko prikazani v **analogni(zvezni) obliki ali pa v digitalni(nezvezni) obliki**. Analogna oblika obsega neskončno št. vrednosti, digitalna pa končno. Povedali smo ključne zahteve pri predstavitvi informacij, in pa kaj je zunanja in notranja predstavidev in komu je prirejena. Učili smo se o dvojiškem številskem sistemu, o pretvarjanju iz dvojiškega v desetiški sistem in obratno, o negativnih binarnih številih, o številih s plavajočo vejico in pa tudi, kako predstavimo znake v dvojiški obliki. Veliko novega sem izvedel tudi o predstavitvi zvoka, predstavitvi slike in o stiskanju podatkov ter shranjevanju podatkov. Učili smo se tudi o Boolovi logiki, Boolovih operatorjih in o realizaciji Boolovih operatorjev z logičnimi vratmi, katera so realizirana s tranzistorji. Učili smo se tudi o algoritmih za konstrukcijo vezij in o kontrolnih vezjih(izbirnik, dekodirnik).

Najprej smo govorili o **analognih in digitalnih podatkih**, in povedali, da analogna oz. zvezna predstavitev obsega **neskončno število vrednosti**, digitalna ali diskretna pa **končno**. Povedali smo, da so **ključne zahteve pri predstavitvi informacije jasnost, nedvoumnost in zanesljivost in pa da je zunanja predstavitev prirejena človeku (desetiška št., znaki na tipkovnici), notranja pa prirejena računalniku (dvojiška št., dvojiške kode za znake).**

Nato smo govorili o **dvoijškem številskem sistemu, katerega osnova je 2**, in ima števki 0 in 1, vsako mesto pa ustreza potenci števila 2. Učili smo se tudi pretvarjati iz dvojiškega v desetiški sistem in obratno, ampak ker je to trivialno tega ne bom posebej opisoval. Učili smo se tudi o **negativnih binarnih številih** in različnih predstavitvah le teh, **ki so notacija predznak in velikost ter dvoijški komplement**. Učili smo se tudi o **številih s plavajočo vejico, katere predstavimo v načinu mantisa in eksponent.**

Govorili smo tudi o predstavitvi znakov, ki jih predstavimo tako, da jih preslikamo v binarne kode (ASCII, UNICODE itd.). Povedali smo tudi nekaj o **predstavitvi zvoka**, in sicer, da je analgoni pojav karakteriziran z amplitudo, periodo in frekvenco. Povedali smo, da moramo zvok iz analognega signala prevesti v digitalno obliko, kar dosežemo z vzorčenjem in kvantizacijo, povedali pa smo tudi nekaj o avdio formatih (WAV, MP3, MIDI, itd.).

Govorili smo tudi o **predstavitvi slike**, katero vzorčimo, da jo lahko predstavimo digitalno. Sliko **predstavimo v slikovnih elementih tj. pikslih**, pomembno pa je, da se zavedamo, da so pri obdelovanju slik udeležene ogromne količine podatkov (barva, barvna globina, intenziteta itd.).

Nato smo povedali tudi nekaj o kompresiji podatkov in podvrstah, tj. brezizgubno in izgubno.

Učili smo se tudi **o shranjevanju binarnih podatkov**, kar počnemo z tranzistorji, ki omogočajo elektronsko preklapljanje med stanjema v zelo majhnem času. Tranzisztorji so narejeni iz polprevodnikov, so majhni in poceni.

Nato smo se učilii tudi o **logiki in logičnih konstruktih**, pri katerih smo govorili o **Boolovi logiki, logičnih operacijah in vratih** in kako z uporabo le the gradimo računalniška vezja, na koncu pa smo si pogledali tudi nekaj primerov kako izdelati različna vezja.

4. TURINGOV STROJ:

Na predavanju z naslovom **Stroj in zakaj je program za stroj cela znanost** smo govorili o kontruiranju modelov, o komponentah Turingovega stroja, o računskih agentih in se učili simulirati delovanje Turingovega stroja in napisati preproste programe za Turingov stroj in o Church-Turingovi tezi ter nerešljivih problemih.

Najprej smo obravnavali **probleme, za katere ne obstaja algoritmična rešitev** in povedali, da lahko dokažemo, da ne more obstajati algoritem, ki bi rešil dan problem. Povedali smo tudi, kaj je **"Halting problem".** Govorili smo tudi o tem, kaj so **modeli računskih agentov**. Povedali smo, da mora model:

- **Zajeti pomembne lastnosti pravega agenta (oz. fenomena)**
- **Je običajno podan v različnem (manjšem) merilu**
- **Opusti nekatere (nepomembne) podrobnosti**
- **Nima vseh funkcionalnosti**

Govorili smo tudi o napovedovanju z modeli:

- Z opazovanjem modela napovemo obnašanje
- Varneje, bolj poceni, lažje
- Omogoča testiranje, ne da bi zgradili pravi agent (fenomen)

Model računskega agenta mora:

- **Brati vhodne podatke**
- **Shranjevati in brati podatke iz pomnilnika**
- **Izvajati ukase v odvisnosti od trenutnega vhoda in trenutnega stanja**
- **Proizvesti rezultat**

Nato smo govorili o **matematičnih modelih računalnikov** kot so npr. Končni Avtomat, Skladovni računalnik, Turingov stroj. Povedali smo tudi, da je **pomembno matematično predznanje**, bolj podrobno teorija množic (unija, presek, kratezični product, korespondenca (relacije)) in osnovni jezik matematične logike (predikatni račun prvega reda).

Zatem smo govorili o **Turingovem stroju**. Povedali smo, da ima **trak, neskončen v obeh smereh, pri katerem vsako polje na traku hrani en simbol, na traku so zapisani vhodni podatki, sam trak pa služi kot pomnilnik**. Povedali smo, da so ukazi za Turingov stroj v odvisnosti od trenutnega stanja in vhoda. Ukazi so sestavljeni kot zapiši nov simbol, spremeni stanje ter se premakni eno polje levo ali desno.

Govorili smo tudi o **Turingovem stroju kot računskem modelu,** katerega množica pravil je program Turingovega stroja. Stroj vedno začne brati od **najbolj levega nepraznega polja dalje**, **ne smeta pa obstajati 2 pravili z istim stanjem in vhodnim simbolom (izogibanje dvoumnostim).** Če ne obstaja nobeno pravilo za trenutno stanje in vhodni simbol, se stroj ustavi.

Turingov stroj je zato primeren model računskega agenta, saj zadosti vsem pogojem, ki jih mora računski agent izpolnjevati.

Ker mora biti model algoritma popolnoma urejeno zaporedje nedvoumnih in učinkovito izračunljivih operacij ter proizvesti rezultat in se ustaviti v končnem času, **je Turingov stroj primeren model algoritma**. Ko smo povedali to, smo si pogledali tudi nekaj primerov algoritmov realiziranih z uporabo Turingovega stroja.

Nato smo povedali tudi kaj je **Church-Turingova** teza in sicer "Če obstaja algoritem, ki reši neko nalogo, potem obstaja tudi Turingov stroj, ki reši isto nalogo.". Ta teza ni dokazana, skoraj gotovo ni dokazljiva ampak gotovo drži.

Na koncu smo si pogledali še **nerešljive probleme** in povedali, da **Turingov stroj definira meje izračunljivosti.** Če problem ne more biti rešen s Turingovim strojem, je neizračunljiv.

To predavanje mi je bilo kar zanimivo, saj smo obdelali nekaj pomembnih tem na področju računalništva. **Najpomembnejše se mi zdi, da poznam delovanje Turingovega stroja in njegov pomen.**

5. JEZIKI IN PREVAJALNIKI:

Na predavanju z naslovom **Jeziki in prevajalniki** sem izvedel kar veliko novega.

Učili smo se o Hierarhiji po Chomskemu, ki opredeljuje **tipe gramatike** (Type 0, Type 1, Type 2, Type 3) in opisuje **njihove lastnosti** (Neomejena gramatika, Kontekstno odvisna gramatika, Kontekstno neodvisna gramatika, Regularni izrazi) in povedali, kateri **tip avtomata pripada določenemu tipu gramatike** (Turing Machine, Linear-bounded automaton, Pushdown automaton, Finite state automaton). Pomembno je, da vemo, da **imajo jeziki ekspresivno moč**. Pogledali smo si tudi, kako s pomočjo produkcijskih pravil preidemo iz neterminalov v terminale in tako dobimo zaključne besede (npr. P1: S -> AB). Pogledali smo si tudi **Backus-Naur gramatiko,** ki je gramatika s katero opisujemo računalniške jezike (primer: <postal-address> ::= <name-part><street-address><zip-part>).

Pogledali smo si tudi **programske jezike**, pri čemer smo povedali, da je na vrhu hierarhije naravni jezik, zj. jezik, ki ga govorimo ljudje. Programski jeziki se delijo na različne kategorije:

- **Nizko-nivojski programski jeziki:** strojni jezik, zbirni jezik
- **Visoko-nivojski programski jeziki:** npr. Java, C, Python, ...
- **Jeziki za opisovanje pomena:** XML
- **Jeziki semantičnega spleta:** Web Ontology Language

Povedali smo, da je programskih jezikov več sto in podali tudi nekaj primerov skriptnih in simboličnih jezikov (awk, Mathematica), jezikov za generiranje formul (Lagra Mge) in jezikov za opazovanje pomena (XML, OWL2)

Kot zadnji sklop smo predelali tudi poglavje **Prevajalniki** in opisali glavne **zahteve prevajalnikov** in sam **proces prevajanja.** Glavni zahtevi za prevajalnike sta **pravilnost** (strojni ukazi naredijo točno to, kar pomenijo visoko-nivojski ukazi) in **učinkovitost** (strojna koda mora biti optimizirana in se izvajati hitro). Povedali smo, da proces prevajanja sestoji iz 4 sklopov:

1. **Leksikalna analiza:** Leksikalni analizator (scanner) združuje znake v lekseme (tokens), tako, da izpusti nepomembne znake kot so komentarji in presledki
2. **Sintaksna analiza:** Sintaksni analizator (parser) preveri sintakso in zgradi notranjo predstavitev programa. To stori s pomočjo BNF notacije za opis pravil.
3. **Semantična nalaliza:** Analizira se pomen in generirajo se ukazi, gleda se semantične zapise in primerja podatkovne tipe
4. **Optimizacija kode:** Z lokalno ter globalno optimizacijo se izboljša časovna ali prostorska kompleksnost proizvejene kode. Lokalna optimizacija preverja majhne dele kode v zbirniku, medtem pa globalna optimizacija gleda večje bloke. Za globalno optimizacijo smo odgovorni programerji, ima pa večji učinek.

Menim, da je bilo na predavanju najbolj pomembno, da sem se naučil, da je **gramatika množica pravil, ki definirajo jezik** in pa, **da imajo jeziki ekspresivno moč**. Pomembno je tudi, da sem se naučil, kako zapisovati **Backus-Naur gramatiko, saj tako opisujemo računalniške jezike**, kar mi bo v prihodnosti pri razvoju programske opreme zagotovo prišlo prav. Kar pa se tiče

prevajalnikov, mi je bila snov nadvse všeč, saj poznavanje postopkov za proizvajanje in optimizacijo kode nikoli ne škodi.

6. PROGRAMSKI JEZIKI IN ALGORITMI:

Na predavanju z naslovom **Jeziki, algoritmi, računalništvo** smo obdelali veliko pomembnih tem iz področja razvoja strojne opreme in računalništva kot znanosti.

Najprej smo si pogledali poglavje **Programski jeziki,** kjer smo povedali, **da je jezikov ogromno,** in da je vsak programski jezik primeren za določeno nalogo, nekateri so primernejši za računanje realnih števil, nekateri za oblikovanje in realizacijo grafičnih vmesnikov, nekateri za komunikacijo s podatkovnimi bazami, itd. Pomembno je, da se zavedamo, da jezika ni pametno izbirati glede na naše preference, ampak moramo **izbrati pravilno orodje za določeno opravilo.** Seveda lahko določen problem reši več podobnih jezikov, takrat lahko izbiramo jezik, ki ga najbolj poznamo.

Pogledali smo si **proceduralne programske jezike,** kot so FORTRAN, COBOL, C/C++, Java, Python in mnogi drugi. Proceduralno programiranje pomeni, da se **ukazi izvajajo zaporedno, eden za drugim.** Podrobneje smo si pogledali tudi različne programske jezike, in o vsakem povedali nekaj dejstev (za kaj je pomemben , uporaba in drugi zanimivi podatki). Specifično smo pogledali jezike FORTRAN, COBOL, C, C++, Java, C# in Python. Navedli smo tudi nekatere uporabne lastnosti oz. orodja teh jezikov, kot je naprimer **Garbage Collection,** ogrodje .NET in mnogi drugi, med sabo pa smo primerjali tudi podobne jezike.

Poleg proceduralnih jezikov smo si pogledali tudi **namenske programske jezike**, kot so SQL, HTML in Javascript. Kasneje smo si ogledali tudi **alternativne programerske paradigme, kot so funkcijsko programiranje, logično programiranje in paralelno programiranje**. Navedli smo tudi njihove lastnosti, uporabo in jezike, ki spadajo v te paradigme.

Poleg programskih jezikov smo si pogledali tudi poglavje **Algoritmi**. Na začetku smo definirali, kaj algoritem sploh je, in navedli nekaj vsakdanjih primerov algoritmov. Povedali smo tudi, da obstajajo 3 vrste operacij (**zaporedne operacije, pogojne operacije, iterativne operacije**). Učili smo se, kako **algoritmično rešujemo probleme** in povedali, da moramo razmišljanje prilagoditi algoritmom. Povedali smo, da je pomembno, da se naučimo **analizirati problem in poiskati proceduro za njegovo rešitev.**

Kasneje smo pogledali tudi poglavje **računalništvo** in najprej navedli nekaj zmotnih idej, kaj računalništvo je. Povedali smo, da je program samo orodje oz. sredstvo za dosego cilja. Povedali smo, da moramo računalničarji znati **specificirati, načrtovati in testirati programske pakete in računalniške sisteme.** Povedali smo tudi, kaj je strojna in kaj lingvistična realizacija algoritmov.

To predavanje mi je bilo nasploh všeč, saj je **teorija programskih jezikov ena mojih najljubših tem** v svetu računalništva. Zelo zanimiv mi je bil pregled zgodovine ter uporabe različnih jezikov, in pregled različnih programerskih paradigm. Zelo zanimivo mi je bilo tudi poglavje o algoritmih, saj menim, da je pomembno poznati načine kako reševati algoritme.

Menim, da je pomembno, da se zavedamo, da **noben jezik ni boljši od drugega,** pomemben je le problem, ki ga rešujemo. Ko izbiramo jezik za rešitev nekega problema, **moramo biti aktualni in poznati prednosti in slabosti posameznih jezikov.** Menim , da je pomembno poznati različne

programerske paradigme, saj je dobro razumeti različne perspektive in ideje. Pomembno je tudi, da vemo, da je algoritem **popolnoma urejeno zaporedje nedvoumnih in učinkovito izračunljivih operacij, ki, ko se le te izvedejo, proizvede rezultat in se ustavi v končnem času.**

7. ALGORITMI IN PSEVDOKODA:

Na predavanju **Algoritmi in psevdokoda** sem se naučil veliko novega. Pogledali smo si kaj je psevdokoda, kaj so algoritmi in kako se jih načrtuje, kako jih predstaviti z psevdokodo. Povedali smo tudi nekaj o različnih stavkih, ter kako abstrahirati problem ter ilustrirati delovanje vzorčnih algoritmov.

Najprej smo povedali nekaj o **algortimih**. Začeli smo z **načini predstavitve**, ki so **naravni jezik, programski jezik in psevdokoda.** Navedli smo prednosti in slabosti teh predstavitev:

- **Naravni jezik:** zelo ekspresiven, bogat, enostaven za uporabo in pogosto uporabljan, problem pa je v tem, da je redundanten, nestrukturiran, dvoumen in kontekstno odvisen. (Primer: Robot za podajanje soli lahko zaradi nepoznavanja konteksta narobe razume vprašanje "Kje je sol?")
- **Programski jezik:** strukturiran, načrtovan za računalnike in neredundanten, slabost pa je v tem, da je gramatično "tečen", preveč precizen, kriptičen in vsebuje veliko tehničnih podrobnosti.
- **Psevdokoda:** je nekje vmes med naravnim in programskim jezikom, je enostavna, berljiva a brez strogih pravil, enostavna za vizualizacijo in enostavna za prevod v programske jezike.

Pogledali smo si tudi primere algoritmov, zapisanih v različnih načinih zapisa.

Pogledali smo tudi različne **zaporedne operacije** (**računske, vhod, izhod**) in povedali, **da je zaporedni algoritem sestavljen samo iz zaporednih operacij**. Učili smo se tudi o **nadzornih operacijah** (**pogojni stavki ali vejitve in iterativni stavki**). Pogledali smo si, kaj je pogojni stavek in povedali, da so i**teracije blok operacij, ki se iterativno izvaja v zanki dokler je izpolnjen pogoj za nadaljevanje**. Povedali smo tudi, da poznamo **dva načina za testiranje pogoja** (**pretest loop, posttest loop**) in da se moramo **izogibati neskončnim zankam**.

Na koncu smo pogledali tudi veliko primerov, kar se mi zdi dobro, saj je tako omogočena lažja predstava naučene snovi.

Menim, da je pri tej snovi najbolj pomembno, da poznamo različne načine zapisa algoritmov, da poznamo različne vrste operacij, saj so le te osnovni gradniki programov/algoritmov. Nasploh mi je bila snov zelo zanimiva in poučna, čeprav sem te operacije že poznal. Sploh mi je bilo zanimivo te operacije pogledati bolj iz "teoretičnega" stališča.

8. ALGORITMI IN KOMPLEKSNOST:

Na predavanju z naslovom **Algoritmi in kompleksnost** smo se učili o pomembnih **atributih algoritmov** in **kako jih razumeti, kako analizirati učinkovitost algoritmov,** pogledati smo si tudi različne **časovne rede časovnih kompleksnosti** in si pogledali tudi nekaj primerov. Poleg tega smo se tudi učili o **nerešljivih problemih ter aproksimaciji.**

Najprej smo povedali, da **za določen problem lahko obstaja več rešitev** (algortimov) in povedali, da je pomembno da **znamo oceniti algoritme.** Povedali smo, da je za dober algoritem pomembno, da je:

- Enostaven za razumeti
- Eleganten
- Časovno in prostorsko učinkovit

Poleg tega pa smo tudi povedali, da so **najpomembnejši atributi algoritmov**:

- Pravilnost (pomembno poznati problem in prave rezultate, ki se lahko od problema do problema spreminjajo)
- Razumljivost (namen mora biti jasen, algoritem mora biti enostaven za vzdrževanje in nadgrajevanje, prav tako je lažje preverjati pravilnost rešitev)
- Eleganca (VČASIH nasprotujoča z razumljivostjo)
- Učinkovitost (učinkovita izraba virov, prostroska in časovna učnikovitost, pomembna analiza in primeranje algoritmov)

Pogledali smo si tudi primer zaporednjega iskanja.
Nato smo si pogledali še različne **rede velikosti**. Začeli smo z **razredom n (O(n))**, kar pomeni, da je **časovna kompleksnost linearna** in **odvisna od velikosti problema** (št. vhodnih podatkov), **sprememba pa je konstanta z večanjem n**. Nato smo si pogledali tudi **razred n^2** in povedali, da **ima vsaka taka funkcija od neke točke naprej večje vrednosti od linearne funkcije ne glede na konstanti faktor**. Pogledali smo si tudi **razred logn**, pri katerem **vrednosti funkcij naraščajo zelo počasi** in pa r**azred k^n**, pri katerem velja, da teh algoritmov **ne moremo rešiti v polinomskem času** (npr. Hamiltonov cikel). Pri razredu k^n smo povedali tudi, da vsebuje **ogromno št. pregledovanj**, in da so take rešitve praktično nedosegljive. Nato smo vse razrede časovne kompleksnosti tudi našteli:

- Konstantna O(1)
- Lograitemska O(logn)
- Linearitmična O(n*logn)
- Linearna O(n)
- Kvadratična O(n^2)
- Kubična O(n^3)
- Polinomska O(n^p), p > 0
- Eksponentna O(2n)

Na koncu smo povedali tudi pravila za **določanje redu časovne kompleksnosti** in kako štejemo operacije, kako računamo produkt, vsoto in množenje s konstanto. Nato smo si pogledali samo še **približne algoritme**, in povedali da lahko r**ešijo problem le približno, kar pa je ponavadi dovolj dobro (in v sprejemljivem času).**

Predavanje mi je bilo zelo všeč, prav tako je bila snov zelo poučna in uporabna, saj nam je kot razvijalcem programske opreme pomembno, **kako dobri so naši algoritmi in če obstajajo boljše rešitve**. Menim, da je najbolj pomembno, da **znamo določiti rede učinkovitosti naših programov** in da znamo svoje algoritme **analizirati** in **primerjati z potencialno boljšimi rešitvami.**

9. PROGRAMSKO INŽENIRSTVO:

Na predavanju z naslovom **Programsko inženirstvo** sem izvedel veliko zanimivega. Na predavanju smo se učili o **prednostih višje-nivojskih programskih jezikov**, o procesu **kako se ta višje-nivojska programska koda prevaja v objektno kodo** in se naučili primerjati **način izražanja osnovnih operacij** v 3 proceduralnih programskih jezikih (C++, Java, Python). Na koncu pa smo si še pogledali **življenski cikel razvoja programske opreme** in razložili, zakaj je pomemben.

Začeli smo z **zbirnim jezikom.** Je **bolj razumljiv kot strojni jezik,** saj vsebuje simbolične kode ukazov, naslovov ukazov in podatkov, ter vsebuje psevdoukaze. Povedali smo, da se ukazi neposredno preslikajo v strojne ukaze v razmerju 1:1. A kljub temu **ima zbirni jezik tudi pomanjklivosti**. Te pomanjkljivosti so, da **mora programer skrbeti za premikanje podatkov** med pomnilnikom in registri, da je pogled na naloge mikroskopski, da **koda ni prenosljiva** in, da je **daleč od naravnega jezika.**

Nato so sledili **visoko-nivojski programski jeziki,** ki so **bolj razumljivi in lažji za uporabo kot zbirni jezik.** Povedali smo, da so pri teh jezikih **programski konstrukti bližje naravnemu jeziku in matematični notaciji,** da so programi **bolj prenosljivi,** da je pogled na naloge makroskopski in pa da se programer ne rabi ubadati z premikanjem podatkov med pomnilnikom in registri, omogočajo pa tudi **višji nivo abstrakcije.**

Nekaj smo povedali tudi o **prevajanju iz visoko-nivojskih programskih jezikov.**

- Prevajalnik (compiler) prevede izvorno kodo v zbirni jezik oz. v podobno obliko.
- Zbirnik to kodo pretvori v objektno kodo
- Programske knjižnice vsebujejo uporabno preverjeno kodo.
- Povezovalnik (linker) združi več datotek objektne kode v en izvršljivi modul.

Nato smo govorili o **proceduralnih jezikih** in povedali, da so zelo priljubljeni, **programi pa sestojijo iz zaporedji ukazov**. Za primere smo vzeli jezike C++, Java in Python in povedali, da imajo **podobno filozofijo,** ampak se r**azlikujejo v sintaksi (zapis programskih stavkov) in v semantiki (pomen stavkov).** Nato smo si pogledali tudi nekaj zanimivih problemov, rešenih v vseh 3 jezikih, kar je bil zanimiv prikaz razlik med jeziki.

Nekaj smo povedali tudi o **prenosljivosti programov.** Povedali smo, da se o**bičajno distributira izvršljiva koda,** ne pa izvorna. **Izvorno kodo je treba prevesti na vseh platformah, na katerih jo nameravamo izvajati.** Povedali smo. da ima C++ prevajalnik v strojno kodo, Java prevajalnik v nizko-nivojsko prenosljivo (zložno kodo), Python pa ima interpreter, ki sproti interpretira izvorno kodo.

Nato smo povedali tudi, kaj je **programsko inženirstvo** in povedali, da je uporabno pri razvoju kompleksne programske opreme. Navedli smo tudi 3 faze v ciklu razvoja programske opreme:

- **Pred implementacijo:** študija izvedljivosti, specifikacija problema, načrt programa in izbor ali razvoj ter analiza algoritmov

- **Implementacija:** kodiranje, razhroščevanje
- **Po implementaciji:** testiranje, verifikacija, primerjanje, dokumentiranje, vzdrževanje

Na koncu smo vse te faze tudi podrobno opisali, in navedli različne načine razvoja programske opreme (kaskadni, agilni) in povedali kaj nam omogočajo modelna razvojna okolja.

Predavanje mi je bilo zanimivo, saj moramo kot računalničarji poznati **razlike med najpopularnejšimi jeziki** in pa **postopek razvoja programske opreme**, da smo lahko bolj učinkoviti v programiranju. Na predavanju se mi je najpomembnejša zdela **primerjava visoko-nivojskih programskih jezikov in kaj z njimi lahko dosežem**, in pa **razlaga pojma programskega inženirstva.**