Slovenská technická univerzita v Bratislave
Stavebná fakulta

Evidenčné číslo: SvF-5342-99495

Automatizácia práce v programe AutoCAD

Bakalárska práca

2022 **Roman Garkavenko**

Slovenská technická univerzita v Bratislave
Stavebná fakulta

Automatizácia práce v programe AutoCAD

Bakalárska práca

Študijný program: Matematicko-počítačové modelovanie
Študijný odbor: matematika
Školiace pracovisko: Katedra matematiky a deskriptívnej geometrie
Školiteľ: Ing. Martin Ambroz, PhD.

Bratislava 2022 **Roman Garkavenko**

Slovenská technická univerzita v Bratislave
Katedra matematiky a deskriptívnej geometrie

Stavebná fakulta
Akademický rok: 2021/2022
Evidenčné číslo: SvF-5342-99495

ZADANIE BAKALÁRSKEJ PRÁCE

Študent: **Roman Garkavenko**
ID študenta: 99495
Študijný program: matematicko-počítačové modelovanie
Študijný odbor: matematika
Vedúci práce: Ing. Martin Ambroz, PhD.
Vedúci pracoviska: Ing. Marek Macák, PhD.

Názov práce: **Automatizácia práce v programe AutoCAD**

Jazyk, v ktorom sa práca vypracuje: slovenský jazyk

Špecifikácia zadania:

Cieľom práce je vytvoriť pre program AutoCAD doplnok s vlastným užívateľským rozhraním, ktorým sa zjednodušia a urýchlia vybrané činnosti. V práci sa vytvorí dialógové okno s rôznymi možnosťami pre označenia miestností a následne sa umožní tvorba tabuľky miestností z už vytvorených označení miestností.

Termín odovzdania bakalárskej práce: 05. 05. 2022
Dátum schválenia zadania bakalárskej práce: 04. 05. 2022
Zadanie bakalárskej práce schválil: prof. RNDr. Karol Mikula, DrSc. – garant študijného programu

POKYNY
na vypracovanie bakalárskej práce

Úvodné ustanovenie

V zmysle zákona č. 131/2002 Z. z. o vysokých školách a o zmene a doplnení niektorých zákonov v znení neskorších predpisov je súčasťou štúdia podľa každého študijného programu aj záverečná práca. Jej obhajoba patrí medzi štátne skúšky. Záverečnou prácou pri štúdiu podľa bakalárskeho študijného programu je bakalárska práca. Podkladom na vypracovanie bakalárskej práce je zadanie bakalárskej práce

Štruktúra záverečnej práce

- titulný list,
- zadanie záverečnej práce,
- pokyny na vypracovanie,
- vyhlásenie autora,
- názov a abstrakt v slovenskom a v anglickom jazyku (spolu v rozsahu jednej strany),
- obsah s očíslovaním kapitol,
- zoznam príloh,
- zoznam skratiek a značiek,
- text samotnej práce (odporúčané členenie),
  - úvod,
  - súčasný stav problematiky,
  - ciele záverečnej práce,
  - vlastné riešenie členené na kapitoly podľa charakteru práce,
  - zhodnotenie dosiahnutých výsledkov resp. navrhnutých riešení,
  - záver,
- resumé v slovenskom jazyku v rozsahu spravidla 10 % rozsahu ZP (len pre práce vypracované v cudzom jazyku),
- zoznam použitej literatúry,
- prílohy (výkresy, tabuľky, mapy, náčrty) vrátane postera s rozmermi 1000x700 mm.

Rozsah a forma

1. Obsah a forma záverečnej práce musí byť spracovaná v zmysle vyhlášky MŠVVaŠ SR č. 233/2011 Z. z., ktorou sa vykonávajú niektoré ustanovenia zákona č. 131/2002 Z. z. a v zmysle Metodického usmernenia č. 56/2011 o náležitostiach záverečných prác.
2. Vyžadovaný rozsah bakalárskej práce je 20 až 30 strán. Odovzdáva sa v dvoch vyhotoveniach. Jedno vyhotovenie musí byť viazané v pevnej väzbe (nie hrebeňovej) tak, aby sa jednotlivé listy nedali vyberať. Rozsiahle grafické prílohy možno v prípade súhlasu vedúceho práce odovzdať v jednom vyhotovení.
3. Autor práce je povinný vložiť prácu v elektronickej forme do akademického informačného systému. Autor zodpovedá za zhodu listinného aj elektronického vyhotovenia.

4. Po vložení záverečnej práce do informačného systému, predloží autor fakulte ním podpísaný návrh licenčnej zmluvy. Návrh licenčnej zmluvy je vytvorený akademickým informačným systémom.
5. Odporúčaný typ písma je Times New Roman, veľkosť 12 a je jednotný v celej práci. Odporúčané nastavenie strany - riadkovanie 1,5, okraj vnútorný 3,5 cm, vonkajší 2 cm, zhora a zdola 2,5 cm, orientácia na výšku, formát A4.
6. Obrázky a vzorce sa číslujú v rámci jednotlivých kapitol (napr. obr. 3.1 je obrázok č. 1 v kapitole 3). Vzorce sa číslujú na pravom okraji riadku v okrúhlych zátvorkách - napr. (3.1).
7. Všetky výpočty musia byť usporiadané tak, aby bolo možné preveriť ich správnosť.
8. Pri všetkých prevzatých vzorcoch, tabuľkách, citovaných častiach textu musí byť uvedený prameň.
9. Citovanie literatúry vrátane elektronických materiálov sa uvádza podľa STN ISO 690 (01 0197): 2012. *Informácie a dokumentácia. Návod na tvorbu bibliografických odkazov na informačné pramene a ich citovanie.*
10. Príklad zoznamu bibliografických odkazov:
    ABELOVIČ, J. a kol.: *Meranie v geodetických sieťach.* Bratislava: Alfa 1990. 104 s. ISBN 0-1554-9173.
    MICHALČÁK, O. – ADLER, E.: Výskum stability dunajských hrádzí. In: *Zborník vedeckých prác Stavebnej fakulty SVŠT.* Bratislava: Edičné stredisko SVŠT 1976, s. 17-28. ISBN 0-3552-5214.
    ŠÜTTI, J.: Určovanie priestorových posunov stavebných objektov. *Geodetický kartografický obzor*. 2000, roč. 2, č. 3, s. 8-16. ISSN 0811-6900.
    Article 18. Technical Cooperation. http://www.lac.uk/iso/tc456 (2013-09-28)
11. Za jazykovú a terminologickú správnosť záverečnej práce zodpovedá študent.
12. Formu postera (elektronická alebo aj tlačená) určí garant študijného programu.
13. Vzor pre poster je uvedený na dokumentovom serveri v akademickom informačnom systéme univerzity.

.....
podpis garanta študijného programu

Ustanovenia týchto pokynov som vzal na vedomie. Som si vedomý(á), že ak nebude moja bakalárska práca vypracovaná v súlade s týmito pokynmi, nebude prijatá na obhajobu.

V Bratislave ....
podpis študenta

Čestné prehlásenie

Prehlasujem, že som túto záverečnú prácu vypracoval samostatne pod vedením vedúceho záverečnej práce, s použitím literatúry uvedenej v zozname použitej literatúry.

Bratislava 5. 5. 2022

Roman Garkavenko

Poďakovanie

Chcel by som sa poďakovať Ing. Martinovi Ambrozovi, PhD. za pomoc pri písaní bakalárskej práce a za to, že mi venoval čas.

Bratislava 5. 5. 2022

Roman Garkavenko

Abstrakt

Názov práce: Automatizácia práce v programe AutoCAD
Abstrakt: Hlavnou témou práce je automatizácia úloh v programe AutoCAD pomocou programovacieho jazyka AutoLISP. Práca je rozdelená do troch kapitol. V prvej teoretickej kapitole sa používateľ zoznámi s nástrojmi a nastaveniami programu AutoCAD, ktoré sú potrebné pri vytváraní textových polí, blokov s atribútmi a tabuliek. Druhá kapitola popisuje automatizáciu úloh uvedených v prvej kapitole pomocou jazyka AutoLISP. Praktické použitie vytvorených nástrojov popisujeme v poslednej kapitole, kde ukážeme, ako program funguje na výkrese pôdorysu domu. Vytvorený nástroj je možné použiť ako na vzdelávacie účely, tak aj v praxi pri vytváraní projektovej dokumentácie.

Kľúčové slová: AutoCAD, Autolisp, Automatizácia

Abstract

Title: Automating tasks in AutoCAD
Abstract: The main topic of this thesis is the automation of tasks in the AutoCAD program using the AutoLISP programming language. The thesis is divided into three chapters. In the first theoretical chapter, the tools and settings of AutoCAD necessary in creating text fields, blocks with attributes and tables are introduced. The automation of tasks presented in the first chapter using the AutoLISP is described in the second chapter. The practical use of the created tools is described in the last chapter, where we demonstrate the program functionality on the floor plan of the house. The created tool can be used both for educational purposes and in practice when creating project documentation.

Keywords: AutoCAD, Autolisp, Automatization

Obsah

Úvod **11**

1 Používanie príkazov programu AutoCAD **13**
1.1 Vytvorenie uzatvorenej lomenej čiary 13
1.2 Získanie hodnoty plochy uzatvorenej lomenej čiary 14
1.3 Vytvorenie definície bloku s atribútmi 15
1.4 Vytvorenie tabuľky 18

2 Implementácia nástroja v jazyku AutoLISP **21**
2.1 Nastavenie hladín 23
2.2 Nastavenie textu 26
2.3 Vkladanie textu, bloku a tabuľky 27

3 Práca s vytvoreným nástrojom **37**

Záver **43**

Úvod

AutoCAD je jedným z najrozšírenejších a najpoužívanejších grafických editorov pre 2D ale i 3D kreslenie. Jeho uplatnenie je veľmi široké naprieč celým radom odborov, od architektov, projektantov, strojárov alebo krajinných inžinierov a mnoho ďalších. V každom z týchto odborov sa v praxi môžu objaviť úlohy, pri ktorých konečný užívateľ len dokola opakuje určitú postupnosť príkazov. A práve takéto úlohy sú vhodným kandidátom na automatizáciu. Teda na zjednodušenie postupu z pohľadu užívateľa tak, že sa odstránenia nepotrebné (opakujúce sa) kroky a ponechajú iba podstatné kroky, v ktorých je nevyhnutný zásah užívateľa.

Automatizáciu v programe AutoCAD môžeme zabezpečiť na rôznych úrovniach. Môžeme využiť makro alebo skript, ktoré ponúkajú len obmedzené možnosti automatizácuie a pri zložitejších úlohách môžeme naraziť na ich limity. V takom prípade je nevyhnutné sa vybrať cestou použitie programovacieho jazyka. Na výber máme veľké množstvo jazykov, napr. AutoLISP, VBA, C++ alebo C#. Z týchto jazykov je s prostredím AutoCADu najviac prepojený jazyk AutoLISP, dokonca má ako jediný z týchto jazykov priamo v programe AutoCAD zabudované vývojové prostredie.

Cieľom tejto bakalárskej práce je vytvorenie nástroja v jazyku AutoLISP [**2**, **3**], ktorý bude automatizovať tvorbu označení miestností a následne aj tvorbu tabuľky miestností. Pre tento nástroj vytvoríme pomocou jazyku DCL vlastné dialógové okno, pomocou ktorého bude možné vypočítať rozmery plôch uzavretých objektov a získané hodnoty zapísať do textového objektu alebo do referencie na blok s atribútmi. Z takto vytvorených referencií na blok je následne možné vytvoriť tabuľku, ktorá sa automaticky vyplní hodnotami z atribútov vybraných refencií na blok. Inými slovami, úlohou tohto programu je čo najviac užívateľovi zjednodušiť prácu v programe AutoCAD a automatizovať celý proces získavania hodnôt plochy a ich pridania do tabuľky. Používateľ tak nestráca čas manuálnym nastavovaním všetkých potrebných príkazov. Jednoducho povedane,všetky tie procesy, ktorými sa program zaoberá, sú časovo náročné, ak nie sú automatizované, teda nastavujete si ich sami.

Práca je členená na tri kapitoly. Prvá kapitola vysvetľuje, ako by sme mali volať a konfigurovať nástroje programu AutoCAD, aby sme dosiahli požadovaný výsledok. V druhej kapitole analyzujeme funkcie, ktoré boli vytvorené v programovacom jazyku AutoLISP na automatizáciu práce v AutoCADe. V tretej kapitole prejdeme k praktickej časti a použijeme vytvorený nástroj na pôdoryse moderného domu.

Kapitola 1

Používanie príkazov programu AutoCAD

Na potvrdenie toho, že postup úlohy je časovo náročný, ukážem krok za krokom, Ako postupne vytvoríme uzavretý objekt *POLYLINE* (lomená čiara) a textový objekt obsahujúci plochu ohraničenú lomenov čiarou. Ukážeme si aj ako vytvoriť definíciu bloku s 3 atribútmi (číslo miestnosti, názov miestnosti, plocha) a ako vložiť referenciu na tento blok s automaticky vypočítanou plochou. Ako posledné si ukážeme ako vytvoriť tabuľku a vyplníme ju pomocou informácií z referencií blokov. Tento postup môžeme rozdeliť na nasledovné kroky:

1. vytvorenie uzatvorenej lomenej čiary,

2. získanie hodnoty plochy uzatvorenej lomenej čiary,

3. vytvorenie definície bloku s atribútmi,

4. vytvorenie tabuľky.

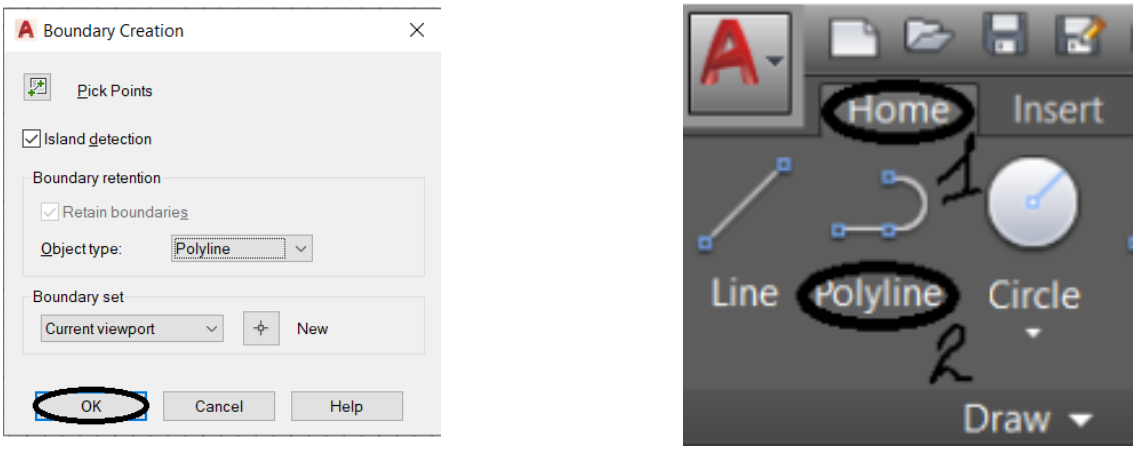
Tieto kroky postupne rozpíšeme v nasledujúcom texte.

1.1 Vytvorenie uzatvorenej lomenej čiary

Prvým krokom je vytvorenie lomenej čiary okolo uzavretého objektu (miestnosti), ktorého plochu neskôr budeme chcieť získat a vypísať. Uzatvorená oblasť sa dá vytvoriť dvoma spôsobmi:

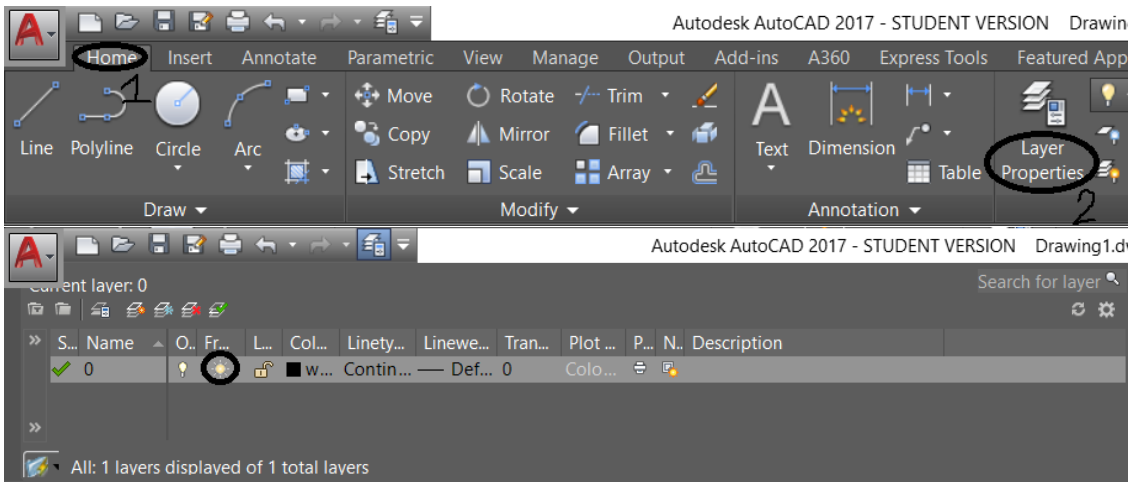
1. Pomocou príkazu **_BOUNDARY**. Po spustení príkazu sa na obrazovke objaví dialógové okno (obr. 1.1) , kde musíme vybrať ako typ vytváraného objektu *Polyline* (defaultná možnosť) a kliknime na *OK*. Okno sa zatvorí a ďalším krokom je kliknutie na ľubovoľný bod vnútri uzavretej oblasti. Následne sa okolo vybraného objektu vytvorí uzatvorená lomená čiara.

2. Pomocou príkazu **_PLINE**. Ďalej len manuálne obkreslíme uzavretý objekt ktorý potrebujeme.



Obr. 1.1: Spôsoby vytvorenia lomenej čiary.

Pri vytváraní uzatvorených oblastí môžu nastať rôzne problémy, najmä pri použití príkazu **_BOUNDARY**. Vo vnútri uzavretej oblasti môžu byť rôzne čiary, útvary. Ide o objekty, ktoré neohraničujú miestnosť a teda patria do iných hladín. Ak napríklad chceme vypočítať plochu miestností v pôdoryse bytu, musíme vyčistiť miestnosť od najrôznejšieho nábytku alebo označení, ktoré môžu prekážať pri výpočte plochy. Problém s prebytočnými objektami vieme vyriešiť tak, že zmrazíme hladiny, v ktorých sú tieto prebytočné objekty. Zmraziť hladiny môžeme v palete *Layer properties manager*, ktorú zobrazíme spustením príkazu **_LAYER**. V tejto palete je zoznam všetkých vytvorených hladín. Kliknutím na slnko pri vybranej hladine ju môžeme zmraziť. V tom prípade sa namiesto slnka objaví snehová vločka a všetky objekty vytvorené pomocou tejto vrstvy stanú sa neviditeľnými (obr. 1.2).



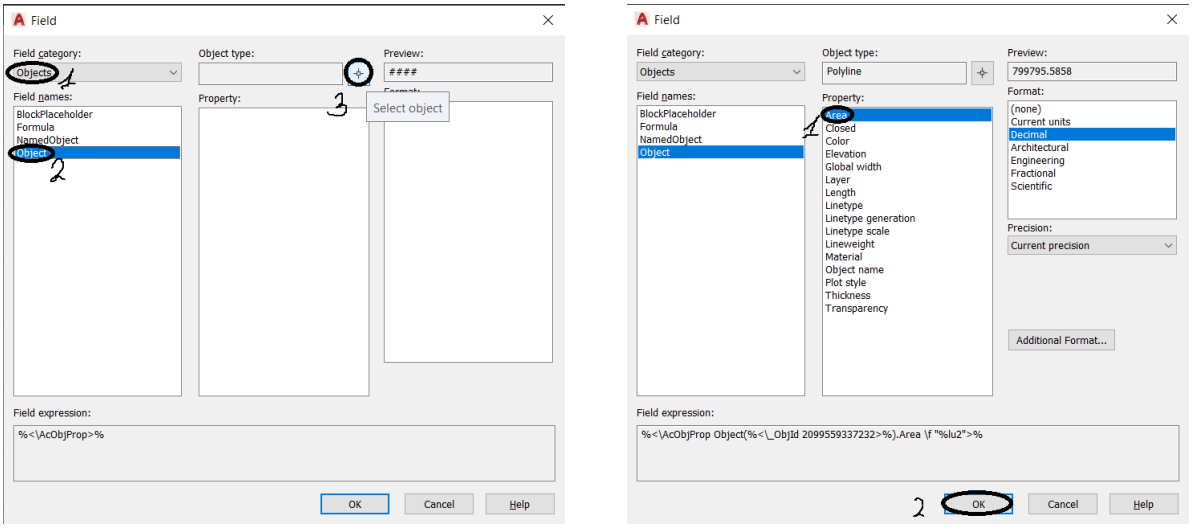
Obr. 1.2: Otvorenie palety na správu hladín a ikona pre zmrazenie alebo rozmrazenie hladiny.

1.2 Získanie hodnoty plochy uzatvorenej lomenej čiary

Po vytvorení uzatvorenej oblasti je našou ďalšou úlohou zistiť plochu ohraničenú touto oblasťou. Túto hodnotu budeme vkladať ako textové pole. Týmto zabezpečíme, že hodnota plochy bude vždy aktuálna, a to aj v prípade zmeny uzatvorenej oblasti. Textové pole môžeme vložiť do atribútu bloku (pozri koniec sekcie 1.3) alebo pomocou príkazu **_FIELD** do textového objektu nasledovným postupom:

Po spustení príkazu **_FIELD** sa na obrazovke objaví dialógové okno (obr. 1.3), kde v ľavej časti musíme vybrať *Object*, v hornej časti okna v strede klikneme na *Select object*. Tým sa nám umožní

vybrať lomenú čiaru, ktorej plochu chceme získať. Po výbere objektu sa dostupnými vlastnosťami naplní stredná časť *Property* dialógového okna, kde vyberieme *Area*. Posledný, voliteľný krok je nastavenie pre jednotky plochy, ktoré sa zobrazia v pravej časti okna, ktoré môžeme v prípade potreby zmeniť. Vytváranie ukončíme kliknutím na *OK* a už len vyberieme miesto kde chceme textový objekt umiestniť.



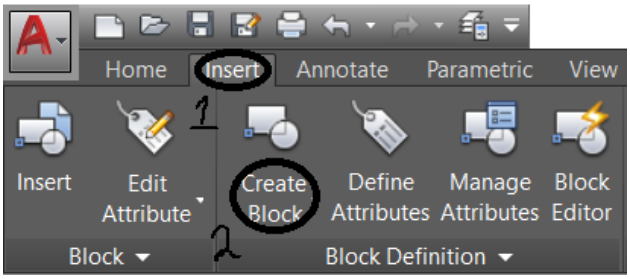
Obr. 1.3: Dialógové okno *field*.

Za zmienku stojí aj to, že ak chceme zmeniť veľkosť textu, musíme si otvoriť panel *Properties*. Urobíme to tak, že klikneme pravým tlačidlom myši na textový objekt, vyberieme *Properties* a v palete vlastností zmeníme parameter *Text height*.

1.3 Vytvorenie definície bloku s atribútmi

Našim cieľom nie je len vypočítať plochu uzavretých objektov, ale aj vytvoriť tabuľku miestností, ktorá bude obsahovať: názvy uzavretých objektov, ich rozlohu v metroch štvorcových, ako aj číslovanie všetkých objektov. Všetky tieto informácie sa do tabuľky môžu vložiť z referencií na blok s atribútmi, čím si zabezpečíme aktuálnosť údajov v tabuľke.

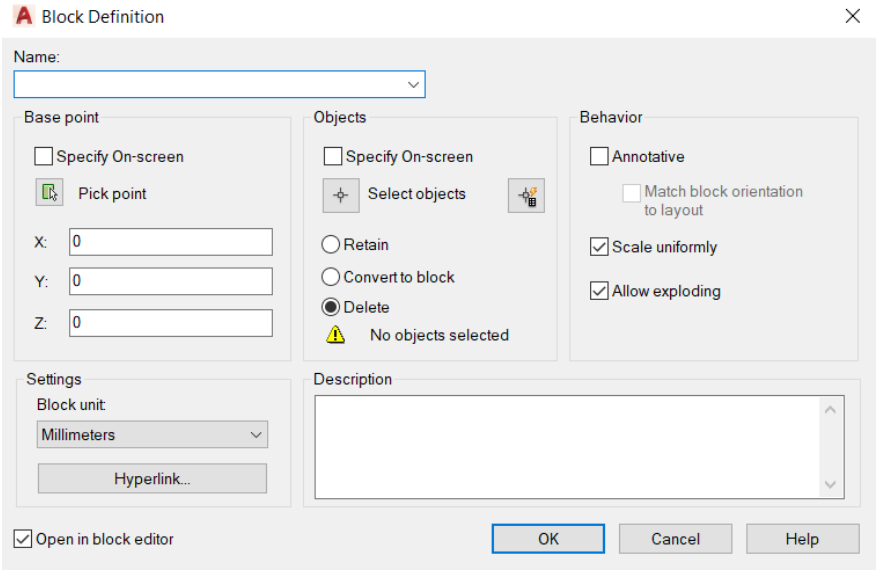
V tejto časti preto popíšeme postup definovania bloku s atribútmi, ktorý budeme potrebovať pri vytváraní tabuľky. Definíciu bloku vytvoríme pomocou príkazu **_BLOCK**. Po spustení príkazu sa na obrazovke objaví dialógové okno (obr. 1.4).



Obr. 1.4: Príkaz *Create block*.

V ľavom hornom rohu dialógového okna musíme nastaviť názov bloku. Rovnako dôležité je povedať AutoCADu, aký by mal byť bod vloženia. Inými slovami, bod okolo ktorého sa blok objaví na našom kurzore pri umiestňovaní referencie na blok. Kliknime na tlačidlo *Select Objects* a

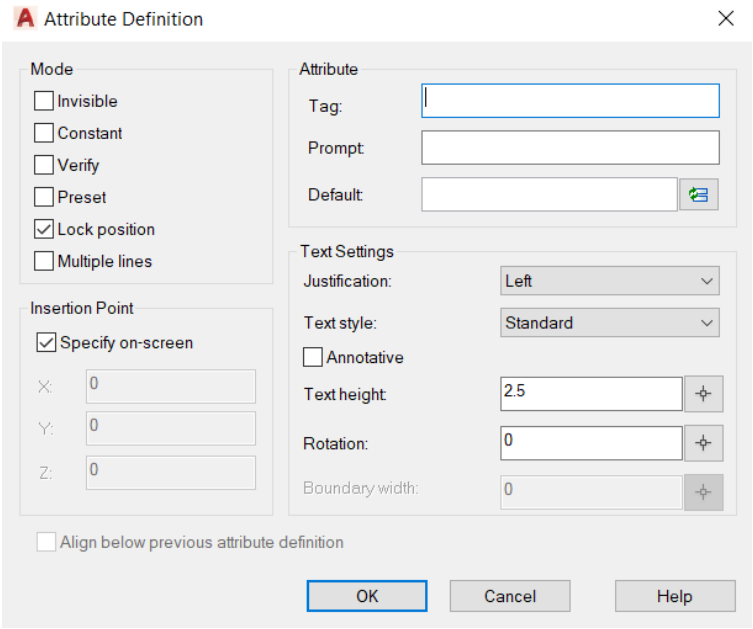
vyberieme objekty, ktoré chceme uložiť do bloku. V našom príklade to budú iba definície atribútov. Vytváranie definície bloku ukončíme kliknutím na *OK* (obr. 1.5).



Obr. 1.5: Dialógové okno *Block Definition* na vytváranie definícií blokov.

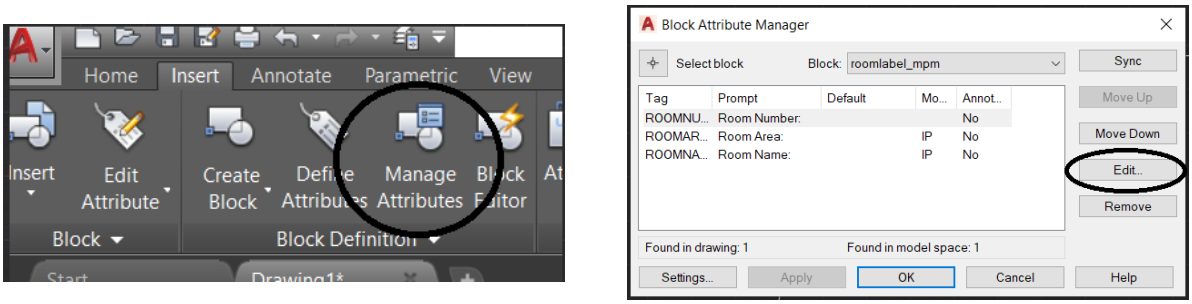
Teraz môžeme prejsť na atribúty. Najprv je dôležité vysvetliť, čo je to atribút. Atribút je štítok alebo značka, ktorá pripája údaje k bloku. Príklady údajov, ktoré môžu byť obsiahnuté v atribúte, sú čísla dielov, ceny, komentáre a mená vlastníkov. Do definícií blokov je možné pridať atribúty, aby boli informatívnejšie. Môžeme napríklad pridať atribút k dverám alebo okenným blokom výkresu s uvedením názvu výrobcu, veľkosti alebo popisu. Informácie o atribútoch extrahované z výkresu sa dajú použiť pri tvorbe tabuliek alebo databáz na vytvorenie zoznamu dielov.

Ako sa vytvárajú atribúty v Autocade? Definíciu atribútu vytvoríme pomocou príkazu **_ATTDEF**. Po spustení príkazu sa na obrazovke objaví dialógové okno (obr. 1.6). V dialógovom okne nastavíme režimy atribútov (napr. viditeľnosť) a zadáme informácie o značke atribútu, umiestnenie a možnosti textu. Vytváranie definície atribútu ukončíme kliknutím na *OK*. Keď sa zobrazí výzva na výber objektov pre definíciu bloku, zahrnieme definíciu atribútu do množiny výberu.



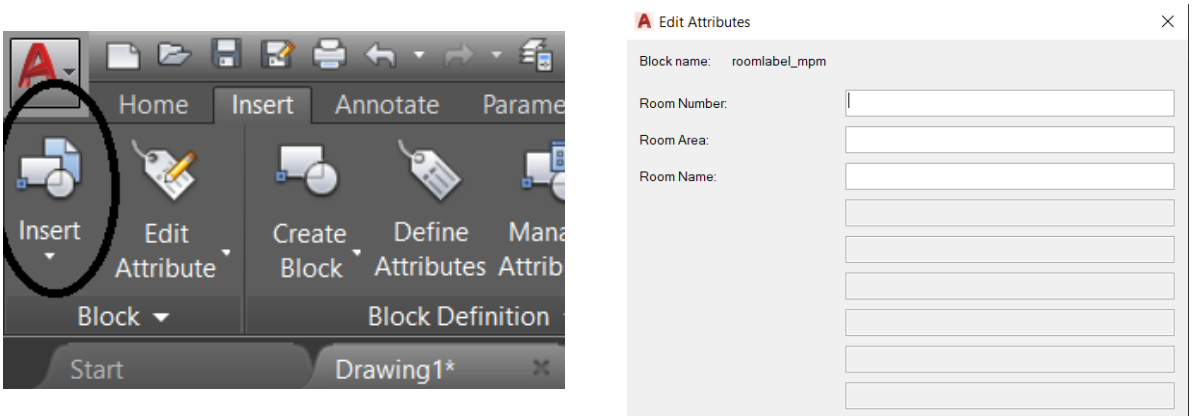
Obr. 1.6: Dialógové okno *Attribute Definition* na vytváranie atribútov.

Máme možnosť spravovať všetky vlastnosti a atribúty vybranej definície bloku. Akékoľvek zmeny atribútov v definícii bloku sa prejavia v referenciách bloku. Ak chceme zmeniť atribúty, prejdeme na *Insert* panel a potom klikneme na *Manage atributes*. Ďalej nám vyskočí okno, kde si môžeme vybrať, pre ktorý blok zmeny v atribútoch nastanú. Vyberieme ľubovoľný atribút a stlačime tlačidlo *Edit...* Po všetkých zmenách klikneme na tlačidlo *OK* (obr. 1.7).



Obr. 1.7: Správca atribútov.

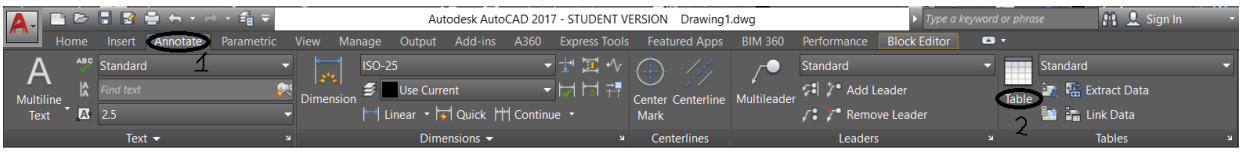
Vytvorené bloky je možné vložiť do výkresu. Po kliknutí na *Insert Block* sa nám zobrazí zoznam, v ktorom sú uložené príslušné definície bloku.Po vložení bloku sa otvorí okno, v ktorom nastavíme hodnoty atribútov (obr.1.8) manuálne alebo vložením textového poľa (pozri sekciu 1.2).



Obr. 1.8: Vkladanie referencie na blok s atribútmi.

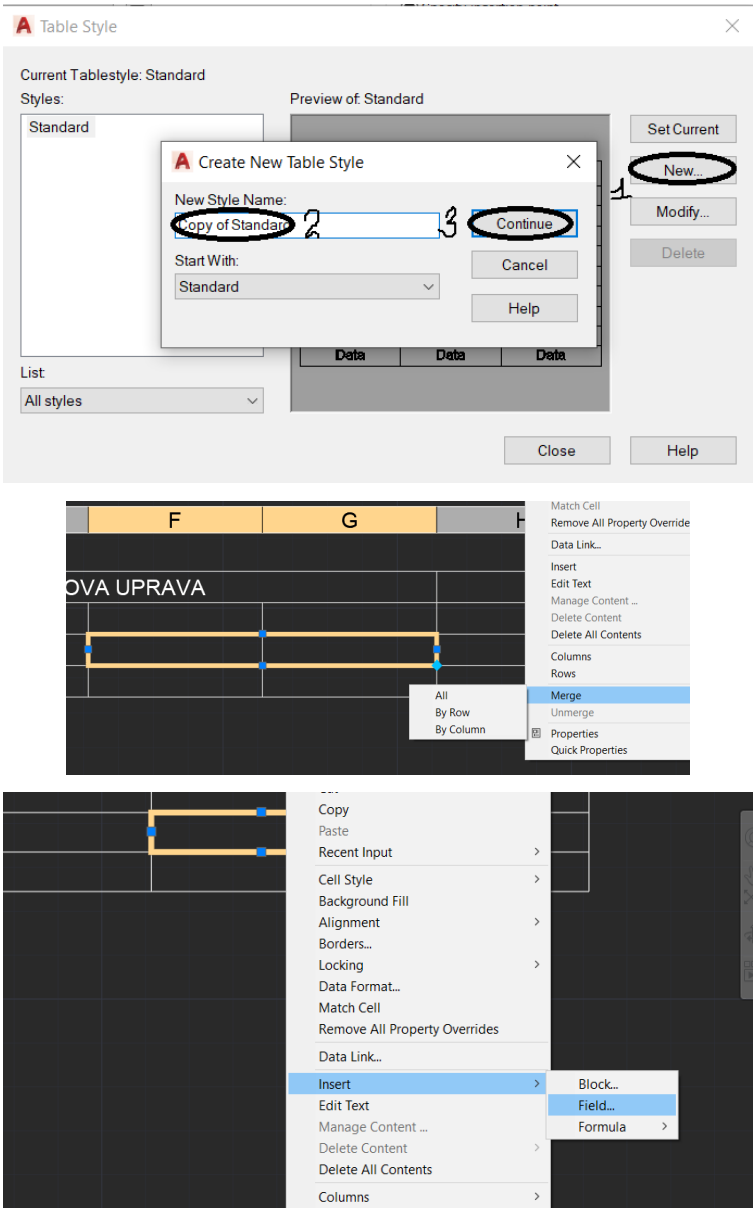
1.4 Vytvorenie tabuľky

Posledným krokom je vytvorenie tabuľky, do ktorej budeme umiestňovať informácie z referencií na blok s atribútmi.



Obr. 1.9: Príkaz *Table*.

Tabuľku vytvoríme pomocou príkazu **_TABLE** (obr. 1.9). Po spustení príkazu sa na obrazovke objaví dialógové okno *Insert table* , v ktorom vytvorime štýl pre tabuľku. Ak chceme vytvoriť samotnú tabuľku, na paneli *Column & row settings* nastavime počet riadkov a stĺpcov tabuľky, potom výšku riadkov a šírku stĺpcov a kliknime na tlačidlo *OK*. Potom kliknutím pravým tlačidlom myši na pracovnú obrazovku vytvoríme tabuľku s n riadkami a m stĺpcami. Ak chceme zadať údaje do tabuľky, musíme dvakrát kliknúť na bunku pravým tlačidlom myši a zadať údaje, ktoré potrebujeme. Bunky môžeme aj spájať. Pri výbere jednej bunky podržíme *shift* na klávesnici a vyberieme nasledujúcu bunku. Ďalej klikneme pravým tlačidlom a vyberieme *Merge*. Do bunky môžeme vložiť pole (pozri sekciu 1.2), blok alebo vzorec (obr. 1.10).



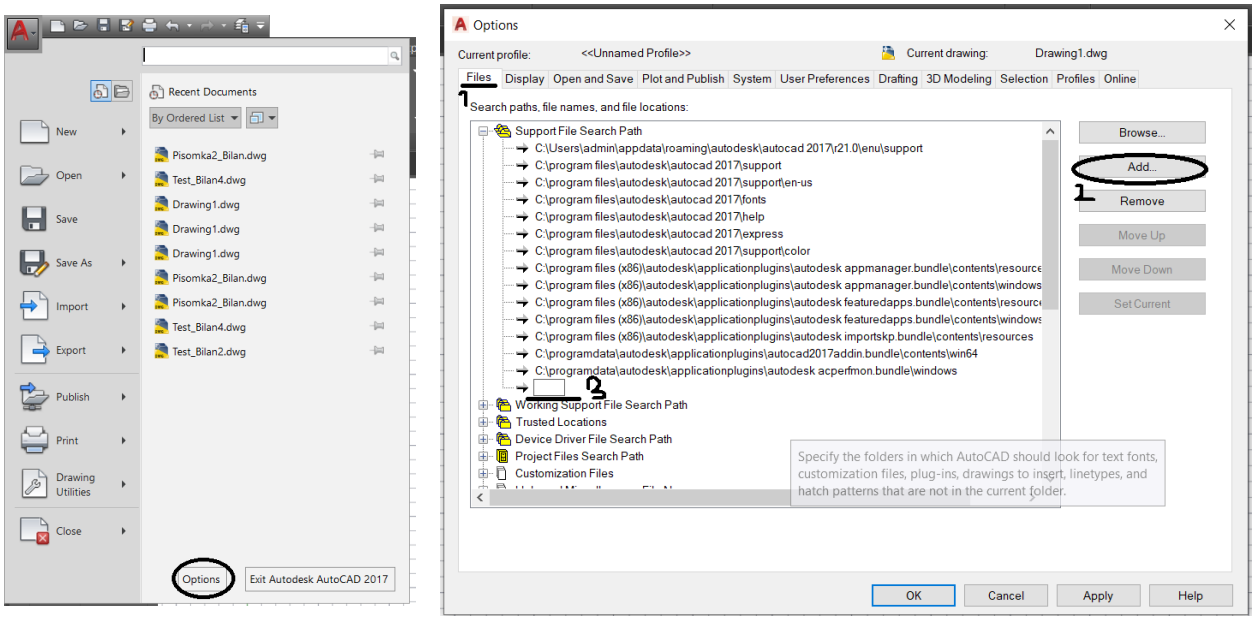
Obr. 1.10: Vytváranie, úprava a vypĺňanie tabuliek.

Kapitola 2

Implementácia nástroja v jazyku AutoLISP

Na automatizáciu označovania miestností a vytvárania tabuľky miestností bol použitý programovací jazyk AutoLISP. Našim cieľom je vytvorenie pomerne komplexného nástroja, pri ktorom má užívateľ na výber z viacerých nastavení a možností. Z toho dôvodu bolo vytvorené aj užívateľské rozhranie v podobe dialógového okna, ktoré bolo navrhnuté v jazyku DCL (Dialog Control Language).

Dialógové okno je definované v súbore s príponu .dcl. Na rozdiel od súborov v jazyku AutoLISP (s príponou *.lsp, ktoré nahráme napr. priamo z vývojového prostredia) je pre jeho používanie nevyhnutné pridať cestu k tomuto súboru v nastaveniach programu AutoCAD (obr. 2.1). Po kliknutí na písmeno A v ľavom hornom rohu programu vyberme *options*. V dialógovom okne následne na karte *files* vyberieme *Support File Search Path*, klikneme na tlačidlo *Add* a zadáme úplnú cestu k *.dcl súboru.



Obr. 2.1: Pridanie cesty k dcl s[boru.

Pozrieme sa teraz na náš súbor ROOMLABELING.dcl. Samotný kód definuje prvky dialógového okna a ich rozmiestnenie. Ako prvé definujeme dialógové okno, jeho názvom, ktorť budeme potrebovať na jeho vyvolanie jazykom AutoLISP a menovku `label`, ktorá sa zobrazí v hlavičke dialógového

okna:

```
ROOMLABELING : dialog {
    label = "Room␣Labeling" ;
```

Následne definujeme štruktúru vo forme stĺpcov a riadkov. Naše dialógové okno bude mať jednoduchú štruktúru pozostávajúcu z jedného stĺpca (`column`) a 3 pomenovaných riadkov (`boxed_row`):

```
    : column {
        : boxed_row {
        label = "&Layers␣selection";

            ...

        }

        : boxed_row {
                label = "&Text␣Settings";

            ...

        }

        : boxed_row {
                label = "&Block␣and␣Table␣creation";

            ...

        }

    }
```

kde uvádzame `...` namiesto definícií rôznych tlačidiel, textových polí a rozbaľovacich zoznamov. Ako príklad si vezmime kód na vytvorenie jedného z tlačidiel, konkrétne tlačidla, ktoré bude zodpovedné za zmrazenie vybranej hladiny:

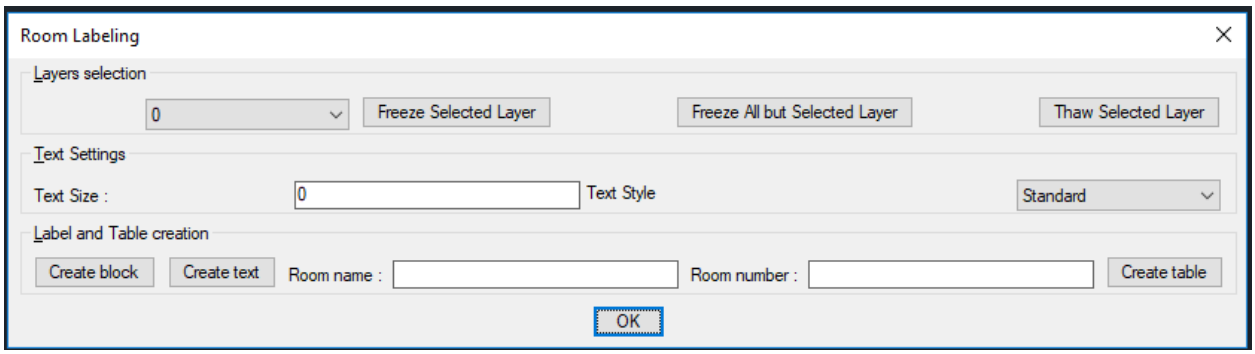
```
            : button
                {
                 key = "vyp";
                 label = "Freeze␣Selected␣Layer";
                 is_default = false;
                 fixed_width = true;
                 alignement = centered;
                }
```

Najprv určíme, aký typ elementu chceme pridať do okna dcl, urobíme to pomocou `button`. Potom vyberieme názov pre `key`, ktorý sa neskôr použije v jazyku AutoLISP ako argument pre funkcie `action_tile` a `set_tile`, pre spustenie priradených funkcií. V ďalšom riadku definujeme `label`, kde nastavíme názov pre tlačidlo, ktorý sa neskôr zobrazí v dialógovom okne. Nakoniec ešte určíme, či je tlačidlo vybrané ako predvolené (`is_default`), šírku tlačidla (`fixed_width`) a zarovnanie tlačidla (`alignment`).

Na záver nášho súboru defuinujeme tlačidlá *OK* a *Cancel* pomocou preddefinovanej funkcionality a ukončíme definíciu dialógového okna:

```
    ok_cancel ;
}
```

Aby sme mohli priradiť každému prvku dialógového okna vlastnú funkčnosť, vytvoríme sa súbor *.lsp s rovnakým názvom ako má súbor *.dcl - ROOMLABELING.lsp. V nasledujúcich sekciách uvedieme a ukážeme naviazanie prvkov okna (obr. 2.2) na vytvorené funkcie, ktoré taktiež popíšeme.



Obr. 2.2: Dialógové okno vytvoreného nástroja.

2.1 Nastavenie hladín

Nastaveniu hladín je venovaná časť dialógového okna s názvom *Layer selection*. Začnime popisom prvkov z ľavej strany. Pri otvorení dialógového okna do rozbaľovaceho zoznamu uložíme zoznam všetkých hladín vytvorených vo výkrese. To vieme urobiť pomocou kľúča `selections`, ktorý sme tomuto zoznamu priradili v jeho definícii v súbore *.dcl:

```
(start_list "selections")
(mapcar 'add_list laylist)
(end_list)
```

V uvedenom úryvku kódu sa rozbaľovaci zoznam naplňá zoznamom, ktorý sme ešte predtým uložili do premennej `laylist`. Keďže v autolisp nie je žiadny príkaz, ktorý by vedel extrahovať názvy všetkých hladín výkresu, bol vytvorený nasledujúci kód, ktorý uloží do zoznamu `laylist` názvy všetkých hladín výkresu:

```
(setq newelem (cdr (assoc 2 (tblnext "layer" T))))
(setq laylist (list newelem))
(while (/= (setq laynext(tblnext "layer")) nil)
   (setq layname (list (cdr (assoc 2 laynext))))
   (setq laylist (append laylist layname))
 )
```

Samotný cyklus `while` funguje celkom jednoducho, až kým nedorazíme na koniec zoznamu hladín výkresu pomocou príkazu `tblnext`. A práve na tento príkaz by som sa chcel zamerať, pretože pomocou výrazu `(tblnext "layer")` dostaneme úplnú charakteristiku ďalšej hladiny vo výkrese. Z tejto charakteristiky vieme pomocou príkazov `cdr` a `asoc` získať samotný názov hladiny, ktorý sa následne zapíše na koniec zoznamu `laylist` po každom prejdenom opakovaní v cykle.

Tlačidlom *Freeze Current Layer* budeme môcť zmraziť vrstvu, ktorá je aktuálne vybratá v rozbaľovacom zozname.

Na zmrazenie ľubovoľnej hladiny bola vytvorená funkcia `FreezeThawLayer`, ktorá má 2 vstupné argumenty: názov hladiny `laName` a stav vrstvy `state`. Hladiny by sme síce mohli zmraziť aj pomocou príkazu **_LAYER**, ale v tom prípade by mohli nastať problémy s operáciami na aktívnej hladine. Tieto problémy pri použití našej funkcie nastať nemôžu, pretože sme ich vo vnútri funkcie ošetrili. Je dôležité si uvedomiť, že ak sa pokúsime zmraziť aktívnu hladinu, objaví sa chyba, pretože aktívnu hladinu nie je možné zmraziť. To znamená, že prvá vec, ktorú musíme urobiť, je vytvoriť kópiu originálnej entity vybranej hladiny do premenej `origLay`:

```
(setq origLay (entget (tblobjname "LAYER" laName))
```

Funkcia **entget** vráti všeky informácie o entite, ktorej meno pošelem do funkcie ako argument. Toto meno získame pomocou funkcie `tblobjname`. Aktuálny stav hladiny s názvom *origLay* získame pomocou:

```
(setq layFreezeState (assoc 70 origLay))
```

V našom nástroji pracujeme so stavom hladiny 1 (zmrazená) alebo 0 (rozmrazená). Je tiež dôležité dodať, že keď vezmeme akúkoľvek hladinu, každá vlastnosť hladiny má svoje špeciálne číslo, ktoré sa nazýva kód skupiny [**1**]. Napríklad ak stav hladiny má kód skupiny 70, názov hladiny zasa kód skupiny 2.

Stav hladiny môžeme zmeniť a uložiť ako modifikovanú entitu do premennej `modLay` nasledovne:

```
(setq modLay (subst (cons 70 state) layFreezeState origLay))
```

Funkciou **subst** zmeníme stav kópie vrstvy na hodnotu uloženú v premennej state. Ak chceme vrstvu zmraziť, jej hodnota bude 1.

V poslednom kroku nahradíme aktuálnu entitu hladiny modifikovanou hladinou pomocou funkcie `entmod`:

```
(entmod modLay)
```

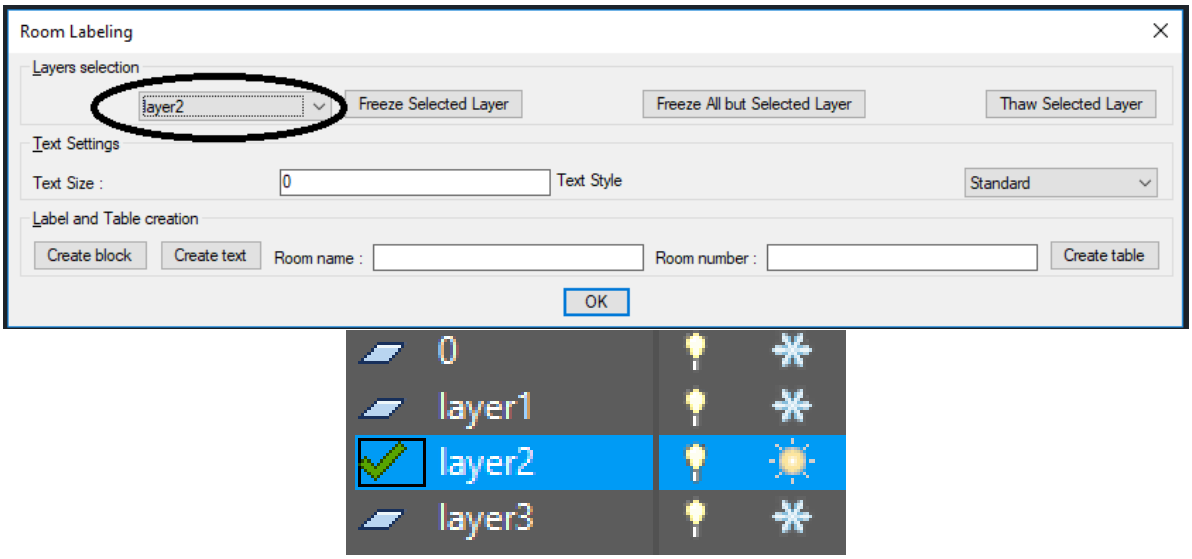
Síce sme už popísali, ako funguje funkcia `FreezeThawLayeer`, ale ešte musíme popísať ako sa po stlačení tlačidla vyberie správna hladina na zmrazenie. Ak chceme zavolať akékoľvek funkcie pomocou prvku dialógového okna, použijeme funkciu `action_tile`.

Konkrétne, na priradenie tejto funkcionalite tlačidlu *Freeze Current Layer* sme použili nasledovný kód:

```
(setq doc (vla-get-ActiveDocument (vlax-get-acad-object)))
(setq selLaName (nth (atoi (get_tile \"selections\"))␣laylist))
(action_tile
␣"vyp"
␣"(FreezeThawLayer selLaName 1) (vla-Regen doc acActiveViewport)"
)
```

Funkcii `action_tile` ako prvý argument pošleme kľúč prvku, ktorému chceme priradiť funkcionalitu. Ďalším argumentom je samotná funkcionalita, ktorá sa má po stlačení tlačidla vykonať. Konkrétne zmrazíme hladinu a obnovíme (zregenerujeme) výkres, aby sa zmena prejavila aj vizuálne. Názov hladiny, ktorú zmrazíme získame pomocou funkcie `get_tile`, teda zmrazíme aktuálne vybratú hladinu v rozbaľovacom zozname. Samotná funkcia `get_tile` nám vráti číslo sekvencie zo zoznamu, podľa ktorého vieme určiť názov hladiny zo zoznamu `laylist`. Obnovu vykreslenia výkresu, aby sa zmena stavu hladiny okamžite vykreslila, zabezpečíme funkciou `vla-regen`.

Ďalšie tlačidlo s názvom *Freeze All but Selected layer* zmrazí všetky ostatné vrstvy okrem vybranej vrstvy v rozbaľovacom zozname. V praxi to vyzera tak, ako na obrazku 2.3.



Obr. 2.3: Ukážka funkcionality tlačidla *Freeze All but Selected Layer*.

Pri stalčení tohto tlačidla sa zavolá funkcia `FreezeAll`:

```
(defun FreezeALL (laylist / a)
(setq a 0)
(setvar 'clayer
        (nth (atoi (get_tile "selections")) laylist)
)
(while (< a (length laylist))
  (if (/= (nth a laylist)
          (nth (atoi (get_tile "selections")) laylist)
      )
    (FreezeThawLayer (nth a laylist) 1)
  )
  (setq a (+ a 1))
)
(vla-Regen (vla-get-ActiveDocument (vlax-get-acad-object))
           acActiveViewport
)
(setq a 0)
)
```

Funkcia využíva funkciu `FreezeThawLayeer` vo `while` slučke. Na začiatok zmeníme aktívnu vrstvu na tú vybranú v rozbaľovacom zozname. Tento krok sa robí preto, aby sme mali možnosť zmraziť všetky ostatné vrstvy. Potom sa spustí cyklus `while`, ktorý bude prebiehať dovtedy, kým premenná `a` nedosiahne veľkosť zoznamu laylist (premenná `a` má hodnotu 0, po každom prejdenom cykle sa pridáva 1). V samotnej slučke je podmienka `if`, ktorá kontroluje každú vrstvu podľa princípu, či sa rovná aktuálne zvolenej vrstve. V prípade, že podmienka zlyhá, zavolá sa funkcia `FreezeThawLaylist` a zmrazí hladinu.

Posledné tlačidlo sa nazýva *Thaw Selected Layer*. Jeho fungovanie je založené na funkcii **FreezeThawLayer**. Princíp fungovania tohto tlačidla je teda identický s tlačidlom *Freeze Selected Layer* s jedným rozdielom: argument `state`, ktorý je vo funkcii v tomto prípade nastavený na 0. To znamená, že funkcia FreezeThawLayer, naopak, rozmrazí vybranú vrstvu.

2.2 Nastavenie textu

Pre nastavovanie textu sme v časti *Text settings* vytvorili dva prvky: textové pole a rozbaľovaci zoznam. V textovom poľom *Text Size* nastavíme veľkosť textu, ktorý sa použije pri vytváraní referencií na bloky s atribútmi alebo textového poľa s textom o veľkosti plochy. Hodnotu, ktorú zadáme do textového poľa alebo získame z rozbaľovaceho zoznamu *Text style* zapíšeme do premennej `txtsize` takto:

```
(action_tile "tSize" "(setq txtsize $value)")
```

Teda vždy pri zmene hodnoty v tomto textovom poli *tSize* sa jeho hodnota uloží do premennej `txtsize`. Samotnú hodnotu z textového poľa získame pomocou `$value`, ktoré odovzdáva hodnotu. premennú `txtsize` sa ďalej používame ako vstupný parameter pre funkcie **ROOMAREA** a **RMABLOCK**.

Prejdeme na rozbaľovaci zoznam *Text style*, ktorý obsahuje všetky vytvorené štýly textu vo výkrese. V kóde sa vytvorí zoznam, do ktorého sa zapíšu názvy štýlov textu, ktory sa používa ako premenná na vytváranie údajov v rozbaľovacom zozname. Samotný zoznam je vytvorený takto:

```
(setq newelem1 (cdr (assoc 2 (tblnext "style" t))))
(setq textlist (list newelem1))
(while (/= (setq txtstyl (tblnext "style")) nil)
   (setq listtxtstyl (list (cdr (assoc 2 txtstyl))))
   (setq textlist (append textlist listtxtstyl))
)
```

Tento zoznam sa vytvára rovnakým spôsobom ako zoznam názvov hladín. Jediným rozdielom je príkaz na získanie všetkých charakteristík štýlu textu `(tblnext "style")`.

Je veľmi dôležité zdôrazniť fakt, že textové pole je previazané s výberom textového štýlu v rozbaľovacom zozname. Presnejšie povedané, textové pole dostáva hodnotu výšky textu pri zmene výberu v rozbaľovacom zozname. Bolo tiež ošetrené, že ak je veľkosť textu v textovom štýle nastavená na 0, tak tento parameter v textovom poli je možné zmeniť. V opačnom prí je veľkosť textu fixná a preto ju nie je možné meniť a textové pole sa stane neprístupným. Podobne, keď sa spustí dialógové okno, textové pole sa nastaví na hodnotu výšky textu defaultného textoveho štýlu.

Pri spustení dialógového okna sa inicializuje hodnota textového poľa funkciou `set_tile` na číselnú hodnotu veľkostí štýlu textu prevzatých zo zoznamu `txtsizelist` nasledovne:

```
(set_tile "tSize" (setq txtsize (rtos (nth 0 txtsizelist))))
```

Pri zmene štýlov textu v rozbaľovacom zozname sa mení aj veľkosť textu a faktor, či je textové pole dostupné na zmeny alebo nie. Tu je príkaz, ktorý je zodpovedný za tento proces:

```
(action_tile "tStyles" "(switchEnableDisable textlist txtsizelist)")
```

funguje to tak, že pri výbere akéhokoľvek textového štýlu v rozbaľovacom zozname spustí sa funkcia `switchEnableDisable`. Funkcia je definovaná nasledovne:

```
(defun switchEnableDisable (textlist txtsizelist / )
  (setvar 'textstyle (nth (atoi (get_tile "tStyles")) textlist))
  (set_tile
    "tSize"
    (setq txtsize (rtos (nth (atoi (get_tile "tStyles")) txtsizelist)))
  )
  (if (= (nth (atoi (get_tile "tStyles")) txtsizelist) 0)
    (mode_tile "tSize" 0)
    (mode_tile "tSize" 1)
  )
)
```

Funkcia `switchEnableDisable` má dva vstupné parametre - dva zoznamy. Prvý zoznam `textlist` obsahuje názvy štýlov textu. Druhý zoznam `txtsizelist` obsahuje veľkosti textu textových štýlov. Funkcia funguje tak, že pri zmene aktívny štýl textu na štýl vybratý v rozbaľovacom zozname. Ďalej pomocou príkazu `set_tile` nastavíme do textového poľa číselnú hodnotu výšky zvoleného štýlu textu. Na záver funkcie sa spustí podmienka `if`, ktorá, ak je veľkosť textu 0, urobí textové pole dostupným na zmeny. V každom inom prípade sa textové pole stane needitovateľným, pretože výška textu je daná v textovom štýle.

2.3 Vkladanie textu, bloku a tabuľky

V poslednej časti dialógového okna sme vytvorili 3 tlačidlá: *Create block*, *Create text*, *Create table* a 2 textové polia: *Room name*, *Room number*.

Po kliknutí na tlačidlo *Create block* sa spustí funkcia **RMABLOCK**, v ktorej sú zadané 3 vstupné argumenty, veľkosť textu, číslo miestnosti a názov miestnosti. Samotná funkcia **RMABLOCK** je veľmi podobná funkcii **ROOMAREA** s niekoľkými rozdielmi, ktoré si popíšeme.

V tejto časti zapíšeme aktívnu hladinu do premennej a zároveň vytvoríme novú hladinu pre polyline s názvom skryta_hladina. Ak sa funkcia už používa znova a hladina s názvom skryta_hladina je už vytvorená, podmienka `if`, vyvolá vetvu na rozmrazenie tejto hladiny. Ak hladina neexistuje vytorí ju:

```
(setq oldlayer (getvar "CLAYER"))
(if (= (tblsearch "layer" "skryta_hladina") nil)
  (command "_.-layer" "_m" "skryta_hladina" "")
  (command "_.-layer" "_t" "skryta_hladina" "")
)
```

Ďalej bod, ktorý získame po kliknutí myšou na ľubovoľné miesto vvo výkrese, sa zapíše do premennej `pt`.

Ak je tento bod v akomkoľvek uzavretom objekte, vytvorí sa okolo tohto objektu pomocou príkazu **_.-BOUNDARY**, ktorý vytvára lomenú čiaru okolo uzavretého objektu. V opačnom prípade dôjde k chybe, pretože príkaz **_.-BOUNDARY** vytvorí lomenú čiaru iba vtedy, ak je bod vo vnútri uzavreteho objektu.

```
(setq pt (getpoint "\nSelect␣a␣closed␣object:"))
(command "_.-boundary" pt "")
```

Poďme analyzovať nasledujúci kúsok kódu:

```
(vl-load-com)
(setq entObject (vlax-ename->vla-object (entlast)))
```

Keďže budeme pracovať s referenciou (pamäťou) na objekt, je veľmi dôležité nastaviť zavolať funkciu `(vl-load-com)`. Táto funkcia načítava rozšírené funkcie jazyka AutoLISP, označované ako Visual LISP. Rozšírenia Visual LISP implementujú podporu ActiveX a AutoCAD reaktora cez AutoLISP a tiež poskytujú pomocné funkcie ActiveX a funkcie konverzie údajov, funkcie na prácu so slovníkmi a funkcie merania kriviek. Do premennej `entobject` pomocou príkazu `((vla-object (entlast))` zapíšeme entitu posledného vytvoreného objektu, čo je v našom prípade lomená čiara. Pozrime sa na nasledujúcu časť kódu:

```
(setq ptt(getpoint "\nSelect␣point:␣"))
(setq textbuff (getvar "Textsize"))
```

Do premennej `ptt` uložíme súradnice bodu, ktorý po výzve vyberieme vo výkrese. Tento bod je zodpovedný za to, kam sa neskôr vloží referencia na blok s atribútmi. Do premennej `textbuff` uložíme veľkosť aktívneho štýlu textu. Keďže je vhodné vytvoriť ďalšiu vrstvu pre vkladanie referencií na blok s atribútmi, popíšeme aj kód, ktorý je zodpovedný za túto úlohu:

```
(if (= (tblsearch "layer" "room_label") nil)
  (command "_.-layer" "_m" "room_label" "")
  (progn (command "_.-layer" "_t" "room_label" "")
    (setvar "CLAYER" "room_label")
  )
)
```

kde v poslednom riadku sa používa príkaz `(setvar "CLAYER")`, ktorý zmení aktívnu vrstvu na vrstvu s názvom room_label.

Pozrime sa na poslednú časť kódu, v ktorej už budeme vytvárať samotneé objekty a vrátime hladinu a výšku textu na pôvodné hodnoty:

```
(command-s "_.TEXTSIZE" txts)
(GenerateBlock "roomlabel_mpm")
(LabelInsert (entlast) rnumber rname "roomlabel_mpm" ptt (atof txts))
(setvar "CLAYER" oldlayer)
(command "_.-layer" "_f" "skryta_hladina" "")
(setvar "Textsize" textbuff)
```

Pred vytváraním referencií na blok nastavíme `_.TEXTSIZE` na veľkosť textu, ktorý nám odovzdáva premenná `txts`. Ďalej sa spustia dve funkcie `GenerateBlock` a`LabelInsert`, ktoré sú zodpovedné za vytvorenie definície bloku a vkladanie referencií naň, pričom sa im podrobnejšie budeme venovať nižšie. Na záver zmrazíme hladinu, ktorá bola vytvorená pre lomenú čiaru a vrátime veľkosť textu do pôvodného stavu.

Funkcia `GenerateBlock` má 1 vstupný argument `name`, čo je názov blokui. Ešte skôr než začneme definovať blok, musíme urobiť kontrolu názvu bloku:

```
(if (not (tblsearch "BLOCK" name))
...
)
```

Ak už existuje blok s rovnakým názvom, nič sa nevytvorí. Ak blok s rovnakým názvom neexistuje, vytvorí sa nový blok. Kód na vytvorenie bloku vyzerá nasledovne:

```
    (progn
      (entmake
        (list
          (cons 0 "BLOCK")
          (cons 2 name)
          (cons 70 2)
          (cons 10 '(0. 0. 0.))
        )
      )
      (entmake
        (list
          (cons 0 "ATTDEF")
          (cons 8 "0")
          (cons 10 '(0.0 0.0 0.0))
          (cons 70 0)
          (cons 1 "")
          (cons 2 "ROOMNUMBER")
          (cons 3 "Room␣Number:␣")
          (cons 40 2.0)
        )
```

```
)
(entmake
  (list
    (cons 0 "ATTDEF")
    (cons 8 "0")
    (cons 10 '(0.0 0.0 0.0))
    (cons 70 9)
    (cons 1 "")
    (cons 2 "ROOMAREA")
    (cons 3 "Room␣Area:␣")
    (cons 40 2.0)
  )
)
(entmake
  (list
    (cons 0 "ATTDEF")
    (cons 8 "0")
    (cons 10 '(0.0 0.0 0.0))
    (cons 70 9)
    (cons 1 "")
    (cons 2 "ROOMNAME")
    (cons 3 "Room␣Name:␣")
    (cons 40 2.0)
  )
```

Pre vytvorenie prvku pre blok sa používa funkcia `entmake`, ktorá vytvorí novú entitu. Samotné charakteristiky, ktoré sme nastavili, priamo súvisia s referenciou dxf. Formát DXF™ je reprezentácia označených údajov všetkých informácií obsiahnutých v Výkresový súbor AutoCAD®. Označené údaje znamenajú, že pred každým údajovým prvkom v súbore je celé číslo, ktoré sa nazýva skupinový kód. Hodnota skupinového kódu označuje, aký typ dátového prvku nasleduje [**1**]. V kóde sme pre blok vytvorili 3 entity definicii atributov, ktoré majú názvy: ROOMNUMBER, ROOMAREA, ROOMNAME. Na záver vytvárenia definície bloku je potrebné vytvoriť aj entitu s názvom ENDBLk, ktorá ukončí definíciu bloku:

```
(entmake
  (list
    (cons 0 "ENDBLK")
    (cons 8 "0")
  )
```

Funkcia `LabelInsert` má 6 vstupných argumentov:

- `PolylineEnt`, entita vybranej lomenej čiary,
- `RoomNumber` a `RoomName`, tieto argumenty získavame z 2 textových polí *Room name* a *Room number*,
- `BlockName` je názov bloku, ktorého referenciu chceme vložiť,
- `Pt`, bod vloženia atribútu. txtSize, nastavená v textovom poli *Text size*, výška textu.

Pozrime sa na kód funkcie:

```
(setq Ename (tblobjname "block" BlockName))
```

Do premennej Entname zapíšeme entitu bloku, ktorého referencie ideme vkladať do výkresu

```
(setq NextEnt (entnext Ename))
(while
    NextEnt
(setq Data (entget NextEnt))
(if (= "ATTDEF" (cdr (assoc 0 Data)))
        (setq Attdefs (cons Data Attdefs))
)
(setq NextEnt (entnext NextEnt))
)
```

V slučke `while` prechádzame cez všetky objekty vnorené v definícii bloku. Ak vnorený objekt sa rovná definícii atribútu `ATTDEF` tak jeho entita sa pripojí k zoznamu `attdefs`.

```
(entmake
  (list
    (cons 0 "INSERT")
    (cons 66 1)
    (cons 67 0)
    (cons 2 BlockName)
    (cons 10 Pt)
  )
)
```

V tejto časti kódu sa vytvorí referencia na blok pomocou príkazu **entmake**.

```
     (foreach x (reverse Attdefs)
         (setq text "")
         (if (= (cdr (assoc 2 x)) "ROOMAREA")
           (progn
             (setq obj (vlax-ename->vla-object PolylineEnt))
             (setq str (itoa (vla-get-ObjectID obj)))
             (setq text
                    (strcat
                      "%<\\AcObjProp␣Object(%<\\_ObjId␣"
                      str
                      ">%).Area␣\\f␣\"%lu2%pr1%ps[]%ct8[0.000001]\">%"
                    )
             )
           )
         )
         (if (= (cdr (assoc 2 x)) "ROOMNUMBER")
           (if (/= RoomNumber nil)
             (setq text RoomNumber)
           )
         )
         (if (= (cdr (assoc 2 x)) "ROOMNAME")
           (if (/= RoomName nil)
             (setq text RoomName)
           )
         )
```

V slučke `foreach` sa pozrieme na to, aké tagy majú atribúty v definícii bloku a podľa toho nastavíme premennú text. Potom priradime tagu každého z atribútov správnu textovú hodnotu uloženú v premennej `text`.

```
        (entmake
          (list
            (cons 0 "ATTRIB")
            (cons 100 "AcDbEntity")
            (cons 62 0)
            (cons 100 "AcDbText")
            (cons 10 Pt)
            (cons 40 txtSize)
            (cons 1 text)
            (assoc 7 x)
            (assoc 71 x)
            (cons 100 "AcDbAttribute")
            (assoc 2 x)
            (cons 70 x)
          )
        )
      )
```

V tejto časti kódu pridávame hodnoty k vlastnostiam atribútov. Je dôležité poznamenať, že máme dve možnosti, ako to môžeme urobiť. A to buď pomocou funkcie `cons` (dvojicou vlastnosť a hodnota vlastnosti) alebo funkciou `assoc` (prevezmeme vlastnosť atribútu z x kde x je definícia atribútu).

```
      (entmake '((0 . "SEQEND") (8 . "0")))
```

Týmto výrazom vo funkcii `entmake` ukončíme vytváranie atribútov.

```
 (vla-Regen      (vla-get-ActiveDocument (vlax-get-acad-object))
                 acActiveViewport
     )
```

Výkres autocadu sa aktualizuje pomocou funkcie `vla-regen`

Tlačidlom *Create text* voláme funkciu **ROOMAREA**, ktorá má 1 vstupný argument `txtsize`. Tento argument je zodpovedný za výšku textu v poli, ktoré sa vytvorí po skončení funkcie. Funkcia **ROOMAREA** je takmer totožná s funkciou **RMABLOCK**. Jediný rozdiel je v tom, že vo funkcii **RMABLOCK** sa volajú dve ďalšie funkcie na vytvorenie bloku s atribútmi a vo funkcii **ROOMAREA** sa namiesto bloku vytvorí pole s textom, ktorý označuje plochu uzavretého objektu. V kóde boli pridané dve veci:

```
 (setq ad (vla-get-ActiveDocument (vlax-get-acad-object)))
```

Do premennej ad sa načítaju ukazovatele na aktívny dokument a priestor, ktoré sú potrebné ako argument funkcie na vytváranie textového objektu. Druhá vec je vytvorenie textového poľa, ktoré umiestnime na miesto, kam sme klikli myšou.

```
(vla-addMText
  (if (= 1 (vla-get-activespace ad))
    (vla-get-modelspace ad)
    (if (= (vla-get-mspace ad) :vlax-true)
      (vla-get-modelspace ad)
      (vla-get-paperspace ad)
    )
  )
  InsertionPoint
  0.0
  (strcat
    "%<\\AcObjProp␣Object(%<\\_ObjId␣"
```

```
    entObjectID
    ">%).Area␣\\f␣\"%lu2%pr1%ps[,␣m\\\\U+00B2]%ct8[0.000001]\">%"
  )
)
```

Vytvoríme teda objekt **MText** v obdĺžniku definovanom bodom vloženia a šírkou ohraničovacieho rámčeka. Príkaz obsahuje 3 argumenty: bod vloženia, šírka, text. Kód na generovanie textového poľa s plochou vybraného objektu lomenej čiary bol prevzatý z programu AutoCAD. Tento kód sa generuje pri vytváraní textového poľa a v našom príkaze sme do tohto kódu potrebovali vložiť identifikačné číslo entity, ktorú máme uloženú v premennej `entObjectID`.

V dialógovom okne nasledujú dve textové polia, *Room name* a *Room number*, do ktorých zapisujeme informácie o názve a čísle miestnosti, ktoré používame ako argumenty vo funkcii **LabelInsert** na označenie atribútov.

Tlačidlo *Create table* volá funkciu `ROOMTABLE`, ktorá následne vytvorí tabuľku s parametram ako je názov miestnosti číslo miestnosti a plocha miestnosti.pozrime sa na kód, ktorý vytvára tabuľku:

```
        (setq
        i 0
        modelspace
         (vla-get-modelspace
           (vla-get-activedocument
             (vlax-get-acad-object)
           )
         )
  )
```

Táto časť kódu vytvorí prázdny zoznam, nastaví premennú počítadla a nastaví odkaz na aktuálny modelový priestor.

```
(princ "\n␣Select␣block␣insertions")
(if (setq sset (ssget '((0 . "INSERT") (2 . "roomlabel_mpm"))))
```

Funguje ako filter, ktorý pri výbere objektu umožní vybrať iba referencie na blok s názvom roomlabel_mpm.

```
    (progn
        (setq   lst '())
      (repeat (setq cnt (sslength sset))
        (setq obj  (vlax-ename->vla-object (ssname sset i))
              atts (vlax-invoke obj 'getattributes)
        )
        (setq lst (cons atts lst));
        (setq i (1+ i))
      )
```

Cyklus repeat beží toľkokrát, koľko je prvkov v sset (koľko je vybraných objektov).Do premennej `obj` uložíme `ename` i-teho aktuálneho objektu. Pomocou funkcie `getattributes` získame zoznam atribútov z referencie bloku `obj` a zapíše sa do premennej `atts`. Nakoniec sa do premennej `lst` pripojí zoznam z premennej `atts`, čím získame 2-rozmerný zoznam, v ktorom bude `cnt` trojíc atribútov.

```
      (setq pt1 (getpoint "\nPick␣point␣for␣table␣"))

      (setq myTable (vla-AddTable
                      modelspace
                      (vlax-3d-point pt1)
```

```
                (+ 4 (sslength sset))
                9
                5
                50
              )
)
```

V tejto časti kódu vyberieme bod, kam sa tabuľka vloží a nastavíme jej parametre ako počet riadkov, stĺpcov, šírku a dĺžku každej bunky.

```
(vla-setcolumnwidth myTable 0 20)
(vla-setcolumnwidth myTable 4 15)
(vla-mergecells myTable 1 2 0 0)
(vla-mergecells myTable 1 1 3 6)
(vla-mergecells myTable 1 2 1 1)
(vla-mergecells myTable 1 2 2 2)
(vla-mergecells myTable 1 2 0 0)
(vla-mergecells myTable 2 2 3 4)

(vla-setText mytable 0 0 "LEGENDA")
(vla-setText mytable 1 0 "C.M.")
(vla-setText mytable 1 1 "PLOCHA")
(vla-setText mytable 1 2 "Nazov␣M")
(vla-setText mytable 1 3 "POVRCHOVA␣UPRAVA")
(vla-setText mytable 2 3 "PODLAHA")
```

Tu sa formátuje tabuľka. nastavíme text do buniek. naformátujeme šírku buniek a niektoré bunky zlučujeme dokopy, aby tabuľka mala požadovaný vzhľad.

```
(setq row 3)

(foreach item lst
  (setq i 0)
  (foreach att item
    (vla-setText
      mytable
      row
      i
      (strcat "%<\\AcObjProp␣Object(%<\\_ObjId␣"
              (itoa (vla-get-ObjectID att))
              ">%).TextString␣>%"
      )

    )

    (setq i (1+ i))
  )

  (setq row (1+ row))
)

(princ (itoa cnt))
(setq i 5)
(setq str "B4")
```

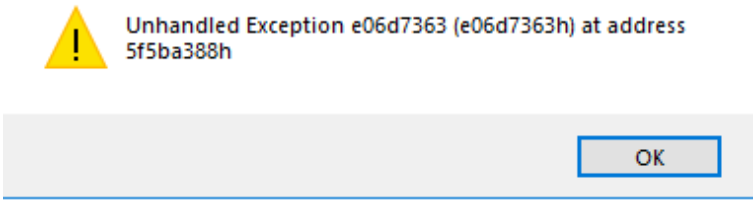
```
        (repeat (- cnt 1)
          (setq str (strcat str " + B" (itoa i)))
          (setq i (1+ i))
        )


        (vla-setText
          mytable
          row
          1
          (strcat "%<\\AcExpr ("
                  str
;                 ")>%"
                  " \\f \"%lu2%pr1\")>%"
          )
        )
```

Táto časť kódu je zodpovedná za vyplnenie tabuľky hodnotami atribútov.Cykly `repeat` prechádzajú každým atribútom a podľa toho, za čo je atribút zodpovedný, dostaneme takúto hodnotu do tabuľky. Premenná `str` obsahuje súčet všetkých ploch.

```
        (vlax-release-object myTable)
        (vlax-release-object modelspace)
      )
      (princ "\nNo objects selected. ")
    )
    (princ)
)
```

Na záver ešte uvoľníme ukazovateľe na tabuľku a modelový priestor.

Pri kliknutí na tlačidlá *Create block*, *Create text*, *Create table* sa spustia funkcie, ktoré vyžadujú interakciu s výkresom AutoCADu. Inak povedané, ak máme otvorené dialógové okno, po kliknutí na jedno z týchto tlačidiel sa vyskytne chyba (obr. 2.4), pretože funkcie nemôžu fungovať, keď je otvorené dialógové okno.



Obr. 2.4: Chyba pri spúšťaní príkazu počas zobrazeného dialógového okna.

Aby sme sa tejto chybe vyhli počas behu funkcie, musíme dialógové okno skryť. Aby sme skryli dialógové okno, musíme pochopiť, že keď zavoláme funkciu `(done_dialog)`, do funkcie `(start_dialog)` sa vráti argument stavu. Ak chceme skryť dialógové okno `(done_dialog)`, vrátené číslo sa musí rovnať 4. V úvode kódu nastavíme premennú `flag` na 4.

```
(setq flag 4)
```

Potom vložíme všetky príkazy, ktoré majú niečo spoločné s tlačidlami v dialógovom okne, do cyklu `while`:

```
(while (> flag 2)
  ...
)
```

V súlade s tým sa dialógové okno nezatvorí, kým premenná `flag` nebude mať hodnotu menšiu ako dva. To sa stane iba vtedy, ak klikneme na tlačidlo *OK* alebo *Cancel*. V príkazoch `action_tile`, ktoré sú pre tlačidlá, ktoré vyžadujú, aby bolo dialógové okno skryté, aby sa predišlo chybe, pridáme `(done_dialog 4)`. Argument 4 znamená, že dialógové okno sa skryje. Potom, čo `(start_dialog)` príjme argument funkcie `(done_dialog)`, zapíšeme toto číslo do premennej `flag`.

```
(setq flag (start_dialog))
```

Potom sa spustí dvojitá podmienka `if` prvá podmienka dáva informáciu o tom, ktorú funkciu chceme volať. druhá podmienka skontroluje, či sa premenná `flag` rovná 4 a iba v tom prípade volá funkciu.

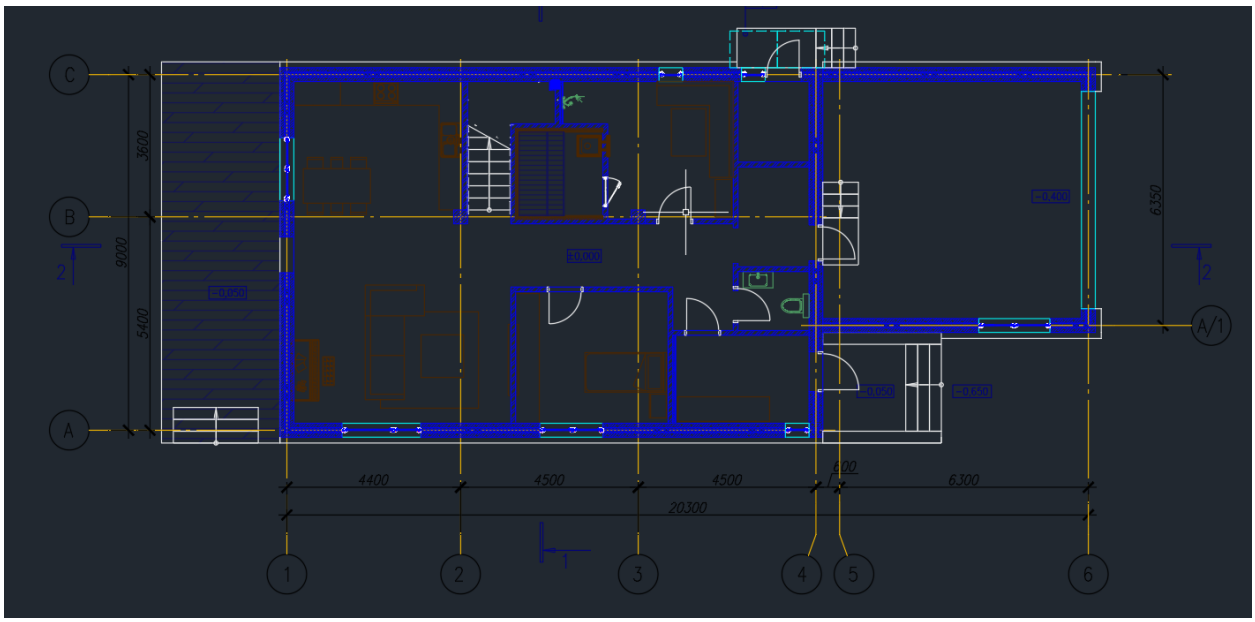
```
(if(= swich 1)(if (= flag 4)(c:ROOMAREA txtsize)))
(if(= swich 0)(if (= flag 4)(c:rmablock txtsize rnumber rname)))
(if(= swich 2)(if (= flag 4)(c:ROOMTABLE)))
```

V prípade, že premenná `flag` má číselnú hodnotu menšiu ako 2, slučka `while` sa preruší a funkciou `(unload_dialog dc_id)` zatvoríme dialógové okno.

Kapitola 3

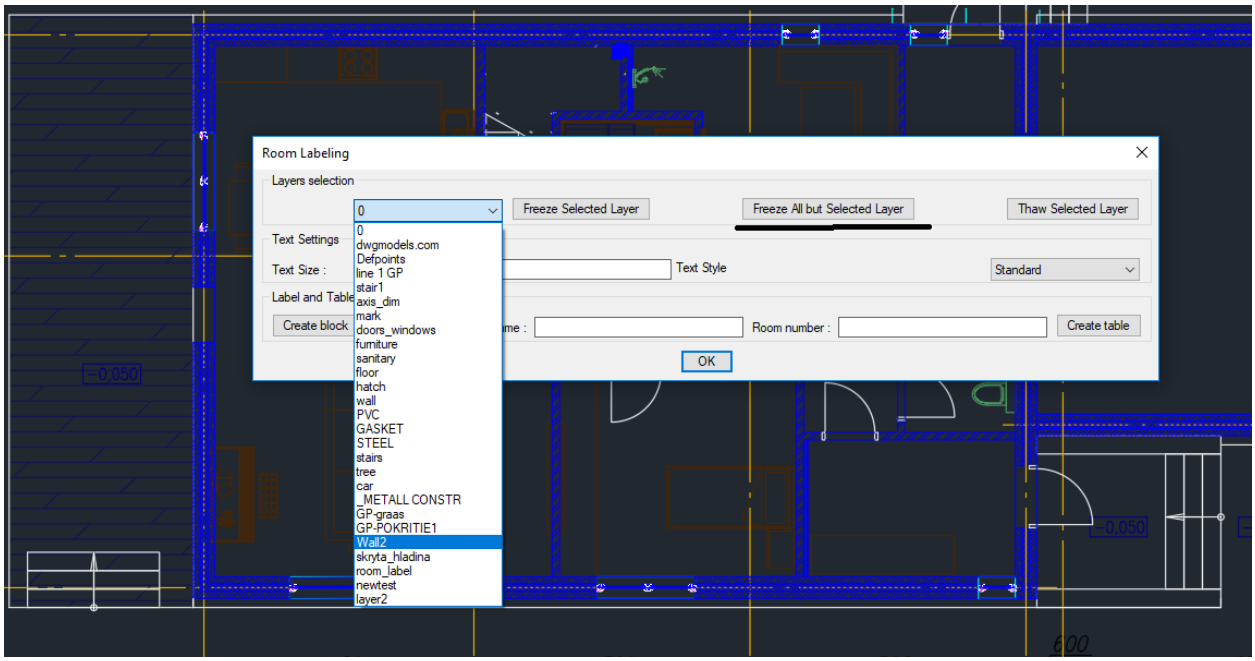
Práca s vytvoreným nástrojom

V tejto kapitole sa pozrieme na to, ako vytvorený nástroj funguje v praxi. Ako príklad použijeme pôdorys domu [**4**] na obr. 3.1. Výkres pozostáva z mnohých hladín, ktoré sa používajú na kóty, nábytok, okná a dvere.



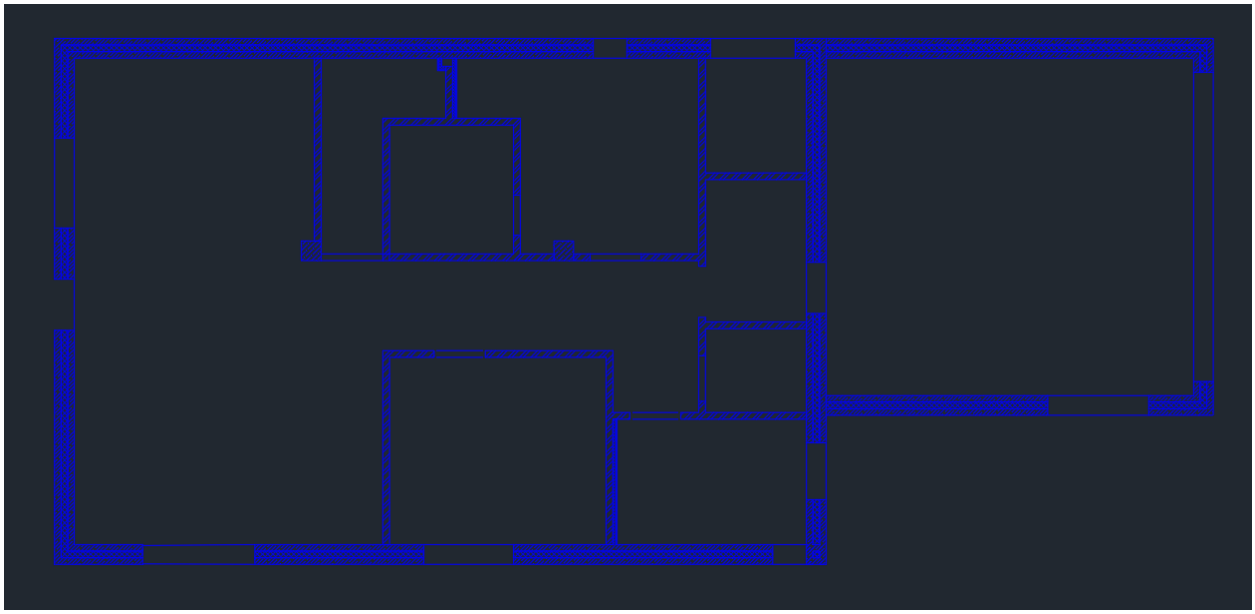
Obr. 3.1: Ukážka pôdorysu spodného poschodia domu.

Samotný nástroj spustíme pomocou príkazu **ROOMLABELING**. Ako prvý krok na ceste k vypočítaniu plochy miestností, musíme v našom dialógovom okne zmraziť všetky hladiny, ktoré nám prekážajú. Najprv otvorme zoznam vrstiev (obr. 3.2).



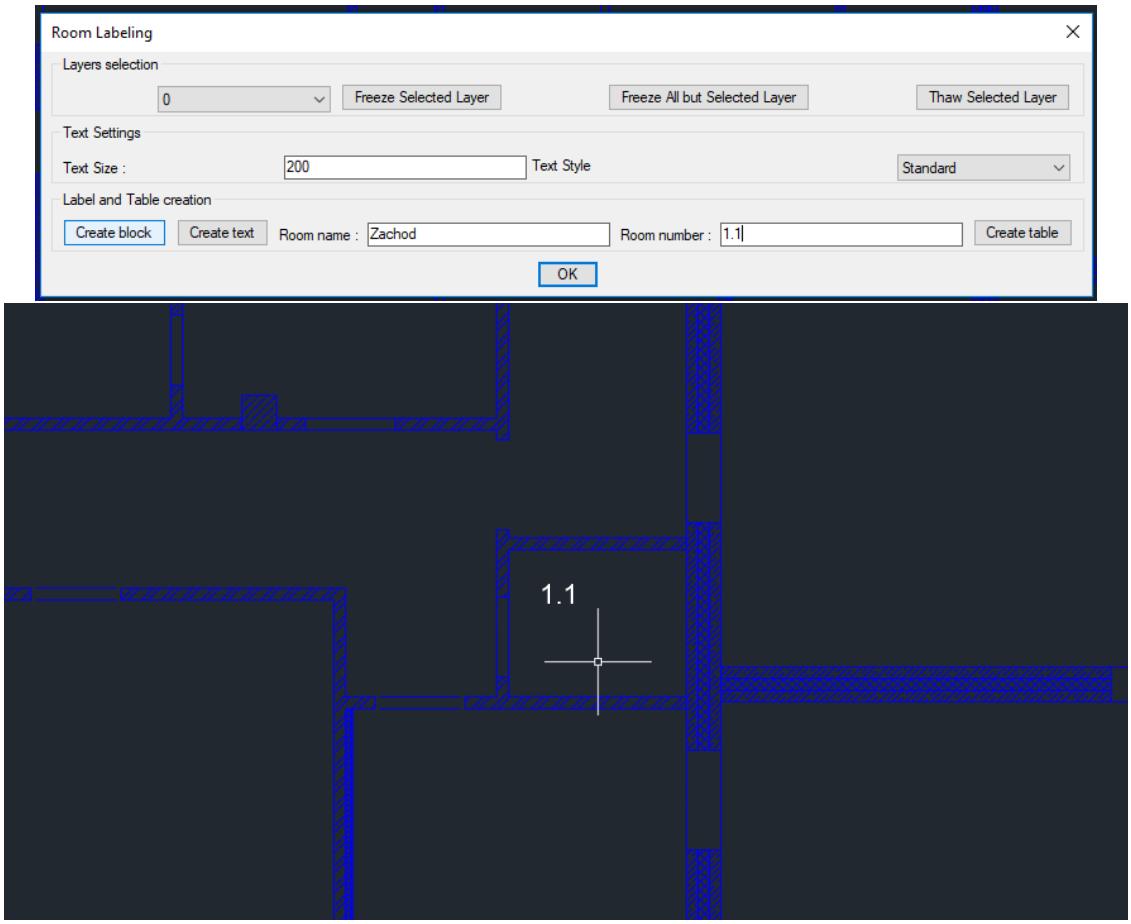
Obr. 3.2: Rozbaľovaci zoznam s hladinami.

Pomocou tlačidla *Freeze Selected Layer* zmrazíme hladiny, bez ktorých sa vytvorí najvhodnejší model na výpočet plôch miestností (obr. 3.3). Taktiež, ak chceme ponechať len jednu hladinu, klikneme na tlačidlo *Freeze All but Selected Layer*. Kvôli veľkému počtu hladín použijeme tlačidlo *Freeze All but Selected Layer* na hladine s názvom Wall2, aby sme ponechali iba holé steny. V prípadoch, keď bola požadovaná hladina nechtiac zmrazená, stačí ju vybrať v zozname vrstiev a kliknúť na tlačidlo *Thaw Selected Layer*.



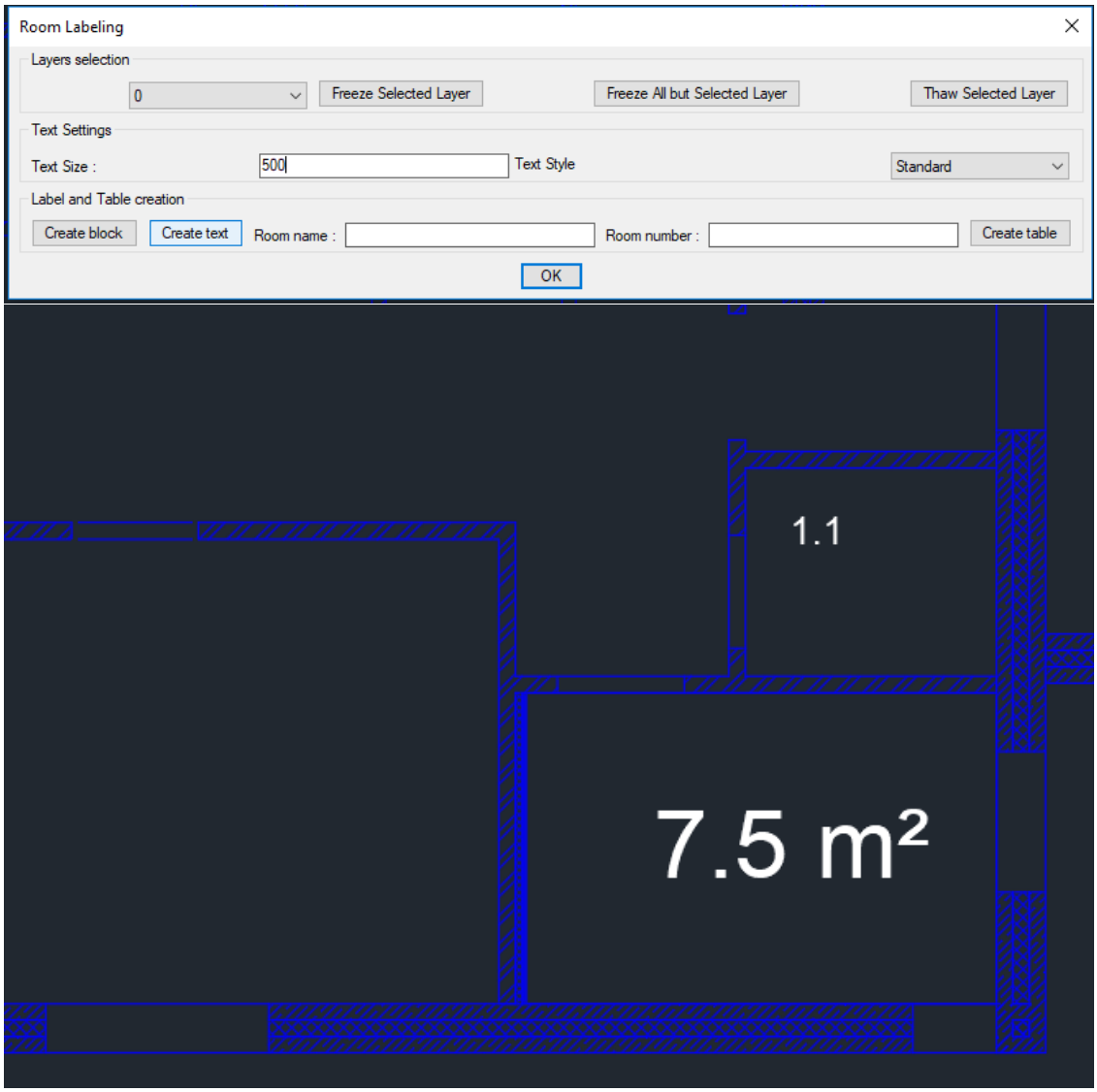
Obr. 3.3: Model pôdorysu bez nepotrebných hladín.

Po nastavení veľkosti textu, štýlu textu, názvu a čísla miestnosti kliknite na tlačidlo vytvoriť blok, čím sa vytvorí blok roomlabel_mpm (obr. 3.4) a po umiestnení erferencie na blok sa zobrazí atribút Room number, ostatné atribúty sú neviditeľné.



Obr. 3.4: Ukážka funkcionality tlačidla *Create Block*.

Keď klikneme na tlačidlo *Create text* (po nastavení textu), vložíme textové pole s veľkosťou plochy miestnosti (obr. 3.5), ktorú sme zvolili ako aktuálny objekt.



Obr. 3.5: Ukážka funkcionality tlačidla *Create Text*.

Aby sme ukázali, ako funguje vytváranie tabuľky, bolo vytvorené 5 referencií na blok pre päť rôznych miestnosti (obr. 3.6). 1,1 - `living room`, 1,2 - `bedroom`, 1,3 - `garage`, 1,4 - `kitchen` , 1,5 - `WC`.

Obr. 3.6: Miestnosti označené v pôdoryse.

Kliknutím na tlačidlo *Create table* dostaneme možnosť vybrať si referencie na blok, z ktorých chceme vytvoriť tabuľku. Potom, čo sme vybrali všetky objekty, ktoré by sme chceli pridať do tabuľky, stlačime na klavesnici *enter* a vytvorí sa tabuľka (obr. 3.7), ktorá zobrazuje údaje ako je názov miestnosti, číslo miestnosti, plocha miestnosti a súčet všetkých plôch.

LEGENDA

C.M.	NAZOV M	PLOCHA	POVRCHOVA UPRAVA			
			PODLAHA	STENY	STROPY	POZNAMKA
1.1	living room	76.0				
1.2	bedroom	12.9				
1.3	garage	39.3				
1.4	kitchen	76.0				
1.5	WC	2.7				
		206.9				

Obr. 3.7: Tabuľka vytorená automaticky z vybraných označení miestností (referencií na blok s atribútom).

Záver

V tejto bakalárskej práci sme sa venovali vytvoreniu nástroja v jazyku AutoLISP pre automatizované využitie príkazov programu AutoCAD na vytváranie označení miestností a následne aj tvorbe tabuliek miestností s cieľom čo najviac zjednodušiť prácu užívateľa a ušetriť čas. V prvej kapitole bolo podrobne vysvetlené, ako pomocou nástrojov programu AutoCAD krok za krokom vytvoriť lomené čiary, získať plochu uzavretých objektov a vytvoriť textové pole, vytvorenie definície bloku s atribútmi a vytvorenie tabuľky a jej nastavení. V druhej kapitole boli analyzované funkcie, ktoré slúžia na automatizáciu práce, ako aj podrobné vysvetlenie kódu na ich tvorbu. Aby používateľ mohol volať funkcie čo najpohodlnejšie, bolo vytvorené dialógové okno pomocou jazyka DCL. V tomto dialógovom okne sme popísali každé tlačidlo z hľadiska kódu a jeho funkčnosti. V poslednej kapitole bol uvedený príklad použitia vytvoreného nástroja na pôdoryse spodného podlažia domu.

Stanovené ciele boli dosiahnuté, program funguje bez chýb. Napriek tomu je ešte niekoľko oblastí, ktoré by sa dali rozšíriť alebo lepšie vypracovať. Napríklad môžeme do blokov pridať ďalšie atribúty a podľa toho rozšíriť výslednú tabuľku. Môžeme tiež vylepšiť algoritmus na výpočet plochy, pretože v prípadoch, keď sú pri odstraňovaní nepotrebných vrstiev v miestnostiach sa objavia malé medzery, funkcia výpočtu plochy nebude fungovať.

Bibliografia

[1] DXF reference. AutoCAD 2012. https://images.autodesk.com/adsk/files/autocad_2012_pdf_dxf-reference_enu.pdf

[2] Togores, R.N., 2018. AutoCAD expert's Visual LISP, release 2019 edition.

[3] Harkow, R., 2013. Essential AutoLISP®: With a Quick Reference Card and a Diskette. Springer.

[4] DWG models.
https://dwgmodels.com/1034-modern-house.html