

АКАДЕМИЈА ТЕХНИЧКО - УМЕТНИЧКИХ СТРУКОВНИХ СТУДИЈА  
БЕОГРАД

ОДСЕК ВИСОКА ШКОЛА ЕЛЕКТРОТЕХНИКЕ И РАЧУНАРСТВА

**Ћирић Угљеша**

**Програм за конверзију растерске графике у ASCII  
уметност**

**- завршни рад -**



Београд, март 2024.

Кандидат: **Ћирић Угљеша**

Број индекса: **РТ-8/18**

Студијски програм: **Рачунарска техника**

Тема: **Програм за конверзију растерске графике у ASCII уметност**

Основни задаци:

- 1. Опис растерске графике, ASCII уметности, C++ језика и GDI+ API.**
- 2. Имплементација програма за конверзију растерске графике у ASCII уметност**
- 3. Опис корисничког интерфејса и резултата конверзије**

Ментор:

Београд, март 2024. године.

---

Др Перица Штрбац, проф. ВИШЕР

## РЕЗИМЕ:

У раду је демонстриран и детаљно описан развој и коришћење Windows конзолне апликације за конверзију растерске графике у ASCII уметност. Разматрани су различити алгоритми који за циљ имају оптималну и веродостојну представу улазног растерског фајла као ASCII уметност. Као кључне технологије у програму се користе GDI+ API, као и Windows API, написан у програмском језику C++. Задатку је приступљено са објектно-орјентисаним развојем на уму. Један од разлога је јаснија хијерархија програма, лакша потенцијална рефакторизација или проширивање и добра основа као компонента комплекснијег система.

**Кључне речи:** *ASCII уметност, GDI+, Windows API, C++, растерска графика, објектно-орјентисан*

## ABSTRACT:

This paper demonstrates and describes in detail the development and use of a Windows console application for converting raster graphics to ASCII art. Various algorithms aimed at the most optimal and credible representation of the input raster file as ASCII art are considered. GDI+ API, as well as Windows API, written in C++ programming language, are used as key technologies in the program. The task was approached with object-oriented development in mind. One of the reasons is a clearer program hierarchy, easier potential refactoring or expansion, and a good foundation as a component of a more complex system.

**Key words:** *ASCII art, GDI+, Windows API, C++, raster graphics, object-oriented*

## САДРЖАЈ

1.	УВОД.....	4
2.	ТЕХНОЛОГИЈЕ.....	5
2.1.	GDI+ .....	5
2.2.	Windows API.....	5
2.3.	C++ std библиотека .....	6
3.	ПРИНЦИП РАДА.....	7
3.1.	Фонт приказа .....	8
3.2.	Иницијализација и провере .....	9
3.3.	Претпроцесирање улазног растерског фајла .....	10
3.3.1.	Претпроцесирање – фаличност боја .....	10
3.3.2.	Претпроцесирање – неповољне димензије растерског фајла.....	11
3.4.	Генерисање матрице на основу улазног растерског фајла.....	13
4.	АЛГОРИТМИ И СТРУКТУРЕ ПОДАТАКА .....	15
4.1.	Иницијализација и покретање програма.....	18
4.2.	Конструктор класе и имплементација програма.....	22
4.3.	Напредан алгоритам конверзије .....	23
4.3.1.	Алгоритам генерисања вектора релативне осветљености ћелије .....	24
5.	УПОТРЕБА И РЕЗУЛТАТИ .....	26
5.1.	Употреба програма .....	26
5.2.	Резултати конверзије .....	28
6.	ЗАКЉУЧАК.....	32
7.	ИНДЕКС ПОЈМОВА .....	33
8.	ЛИТЕРАТУРА .....	34
9.	ИЗЈАВА О АКАДЕМСКОЈ ЧЕСТИТОСТИ.....	35

## 1. УВОД

Апликација представља завршни рад на тему рачунарске графике и названа је **img2ascii**. Представља конвертор растерске графике у ASCII уметност који подржава групу најкоришћенијих и најпознатијих растерских типова битмапа, написан у програмском језику C++. Компајлиран је у 32-битни извршни фајл. Подржава познате растерске типове датотека попут:

- JPG
- PNG
- GIF
- BMP
- TIFF

Сама логика извршавања конверзије је реализована преко објектно-орјентисаног модела приступа решењу, где се већи део логике заправо дешава у главној класи. Позив конструктора са параметрима се дешава из главне **main** функције програма који ове параметре доставља парсирајући кориснички унос преко конзоле. Ово обезбеђује јасан ток програма и видљиву границу између логике извршавања програма као целине и саме логике конверзије која се може изоловати и применити независно од начина и приступа извршавања главног програма. Рад дефинише поглавља која су логички одвојена и говоре о технологијама које су коришћене у изради програма, принципу рада и могућностима програма, коришћеним алгоритмима и структурама података којима програм располаже, детаљан начин употребе и резултате програма и закључак који повезује и сажима свако поглавље које је наведено.

У поглављу које се тиче технологија коришћених у изради програма, детаљно су описани принципи, технологије и алати коришћени у развоју програма **img2ascii**. То укључује програмски језик у коме је програм развијен, тиче се и ближе објашњава Windows API за управљање графичким елементима и приступ фајловима на рачунару, као и GDI+ API за управљање графиком у Windows оперативном систему.

Поглавље које се односи на принципе рада објашњава принцип рада програма **img2ascii** на апстрактном нивоу како би се касније детаљније објаснило на техничком нивоу шта све програм подразумева. То обухвата процес конвертовања растерске графике у ASCII уметност кроз примену одређених алгорита, као и детаље о томе како се врши подешавање произвољних параметара који могу да контролишу изглед крајњег резултата.

Поглавље „Алгоритми и структуре података“ се фокусира на алгоритме и структуре података коришћене у програму **img2ascii**. Описује како се улазна растер слика дели на матрицу и како се свака ћелија те матрице конвертује у ASCII карактер, где се описује битност избора фонта и алгорита за генерисање ASCII уметности. Описује начине функционисања алгорита и залази у тематику и позадинску логику уз објашњење чиме се програм служи и како би се могао искористити у некој широј слици.

У поглављу „Употреба и резултати“ се анализирају конверзије настале путем програма **img2ascii**, као и његова практична примена. Описује се како корисници могу користити програм и какви резултати се могу очекивати, уз евентуалне примере конверзије и приказ добијених ASCII уметности.

## 2. ТЕХНОЛОГИЈЕ

Технологије које се користе у овом програму су GDI+ API и Windows API. Такође, апликација користи стандардне функционалности и структуре података које std библиотека пружа у оквиру дефинисаног стандарда ISO C++14. Поред ових технологија апликација се не ослања на друге екстерне библиотеке или зависности, већ имплементацијом одређених алгоритама и логике омогућава извршавање. Такође, треба напоменути да се у овом програму користе макрои и дефиниције података из Windows API-а свугде где је то било могуће. Како је Windows API органски развијан кроз време где се хардвер мењао учестало, постојала је потреба да се међу компајлерима и разноврсним хардвером и подешавањима успоставе тачне одредбе заузећа меморије између одређених типова података. Данас је кроз стандардизацију хардвера и језика C++ ова проблематика решена, али због компатибилности кода уназад, овакви макрои и дефиниције које се налазе у Windows API-у и даље постоје. Наравно, њихова примена и постојање има још једну важну улогу а то је да ближе одреде намену кода кроз експлицитну дефиницију одређених типова података који имају одређену намену у име које недвосмислено и директно говори чему служи. Како би програм испратио овај дух програмирања, у коду се користе Windows дефинисани типови података где год је било могуће.

### 2.1. GDI+

GDI+ је скраћеница за „Graphic Device Interface Plus “. То је Microsoft-ов скуп библиотека за руковање графичким елементима у Windows оперативном систему. GDI+ се користи за руковање сликама, текстом и другим графичким елементима у Windows апликацијама. Он омогућава програмерима да цртају и манипулишу графиком у апликацијама. GDI+ је алат за рад са графиком који је довољно једноставан за употребу, али моћан да се користи за реализацију сложенијих графичких приказа. Са GDI+-ом је могуће руковати облицима, линијама, текстом, градијентима, трансформацијама и другим елементима, али је као такав направљен да се примарно употребљава код дводимензионе графике и приказа. GDI+ је део Windows оперативног система и доступан је у готово свим верзијама Windows-а. Као такав, GDI+ је избор технологије која ће бити задужена за манипулацијом графичког дела обраде растерске графике.

### 2.2. Windows API

Windows API (Application Programming Interface) је скуп рутина, структура података и дефиниција константи које омогућавају програмерима да приступе одређеним функцијама Windows оперативног система. То укључује приступ графичким елементима, датотекама, меморији и другим системским ресурсима. Често се користи заједно са GDI+ API-ем за стварање графички богатих Windows апликација. Windows API је интегрални део Windows оперативног система и доступан је у свим верзијама Windows-а. Он се може користити у различитим програмским језицима, укључујући C, C++, C#, Visual Basic и другим. Windows API се користи у многим различитим апликацијама, од десктоп апликација до игара. Пружа приступ многим различитим функцијама оперативног система и његовим ресурсима. То укључује, али није ограничен, на функције за рад са датотекама, рад са меморијом, рад са процесима и нитима, рад са

графичким елементима и рад са мрежама. Такође пружа могућност да се користе различити системски ресурси, као што су штампачи и уређаји за унос попут компјутерског миша и тастатуре. Такође садржи мноштво структура података и константи које се користе у апликацијама. Ове структуре података обично садрже информације о објектима у оперативном систему, као што су датотеке, процеси и уређаји. Константе се користе за дефинисање вредности за различите опције и поставке у апликацијама. Представља веома комплексан и огроман скуп алата који се могу употребити у различите сврхе. Функционалност Windows оперативног система је заснована на Windows API-у.

### **2.3. C++ STD библиотека**

C++ std је скраћеница за Standard C++ библиотеку, која је део језика C++ и садржи различите функције и класе које се користе за развој апликација. Ова библиотека се састоји од више подбиблиотека које се користе за различите намене, као што су input/output, стрингови, контејнери и алгоритми. Подбиблиотека „iostream“ садржи функције за рад са датотекама и улазним/излазним токовима података. Подбиблиотека „string“ садржи класе и функције за рад са стринговима и њихову обраду. Подбиблиотеке „vector“, „list“ и „map“ садрже различите типове контејнера, као што су вектори, листе и мапе, који се користе за обраду података. Подбиблиотека „algorithm“ садржи различите алгоритме за обраду података, као што су сортирање и претрага. Осим ових подбиблиотека, C++ std такође садржи и класе за рад са динамичком алокацијом меморије, као и за рад са exception-има. Exception-и су изузеци који се јављају током извршавања програма и користе се за обраду грешака и изузетака. C++ std се користи за развој апликација за различите платформе, као што су Windows, Linux и MacOS. Ова библиотека се стално допуњава и унапређује како би се пружио што бољи и ефикаснији приступ различитим функцијама и класама.

### 3. ПРИНЦИП РАДА

Како би апликација била што јасније објашњена, треба рашчланити њен принцип рада на најапстрактнијем нивоу. Генерална идеја је да се улазни растерски фајл подели на матрицу где би свака ћелија имала једнаке димензије. Однос ширине и висине ћелије матрице би требало да буду једнаки или приближно једнаки односу ширине и висине карактера фонта који се користи за испис ASCII уметности како би се избегла изобличеност изгледа. Свакој ћелији матрице би се доделио квантитативни број који би означавао просечну, односно релативну осветљеност која је нормализована у распону од нула до један. На основу израчунате релативне осветљености сваке ћелије матрице улазног растерског фајла можемо на веома лак начин упоредити вредности са референтном мапом претходно израчунатих вредности релативне осветљености за сваки карактер фонта који се користи. Тиме се добија директно мапирање ћелије матрице са најприближнијим карактером који би представљао ту ћелију матрице за задату релативну осветљеност. Декодирањем матрице добија се низ карактера који се исписује и то представља улазни растерски фајл конвертован у ASCII уметност.

Релативна осветљеност се дефинише као мера светлине боје у односу на референтну белу боју. У RGB систему боја, релативна осветљеност једног пиксела се може израчунати помоћу следеће формуле:

$$Y = R_{lin} * 0.2126 + G_{lin} * 0.7152 + B_{lin} * 0.0722 \quad (3.1)$$

Где би **Y** представљала релативну осветљеност пиксела, а **R<sub>lin</sub>**, **G<sub>lin</sub>** и **B<sub>lin</sub>** линеаризоване вредности црвене, зелене и плаве компоненте боје пиксела у RGB систему боја. Функција за рачунање релативне осветљености је:

```

FLOAT Section::calculate_pixel_brightness(INT x, INT y, BitmapData& imageData)
{
    // Calculating average brightness of a single pixel from the given RGB value for pixel
    // positioned at XY coordinate
    // Pixel luminance = (0.2126 * R + 0.7152 * G + 0.0722 * B)

    byte* pixel = (byte*)imageData.Scan0 + (y * imageData.Stride);

    // Watch out for actual order (BGR)!
    // 3 stands for bytes per pixel
    INT bIndex = x * 3;
    INT gIndex = bIndex + 1;
    INT rIndex = bIndex + 2;

    FLOAT red = (FLOAT)pixel[rIndex];
    FLOAT green = (FLOAT)pixel[gIndex];
    FLOAT blue = (FLOAT)pixel[bIndex];

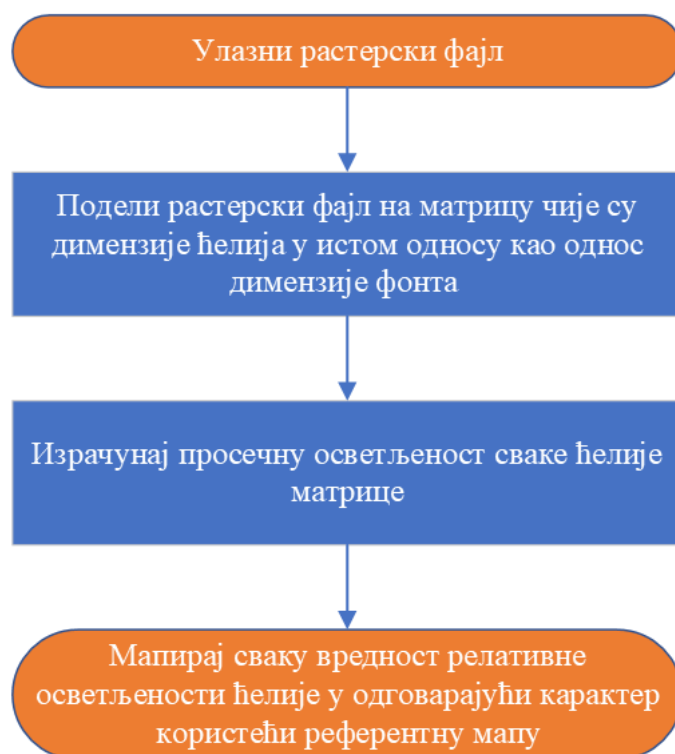
    return ((0.2126f * red + 0.7152f * green + 0.0722f * blue) / 255.f);
}

```

Слика 3.1 Функција за рачунање релативне осветљености једног пиксела



Референтну мапу претходно израчунате релативне осветљености карактера за фонт који се користи рачунамо користећи идентичан алгоритам преко пажљиво усликане слике карактера фонта која је тачних димензија и узима у обзир величину фонта и чињеницу да је сваки карактер исте димензије. Слика 3.3 представља такву слику, и као таква се користила за генерисање референтне мапе релативне осветљености, узимајући у обзир тачне димензије сваког карактера понаособ. Треба имати на уму да све калкулације које се могу извршити пре самог покретања изврше како би се спречиле редувантне калкулације у току извршавања програма. Тиме се постиже додатна брзина. Као део пројекта, у оквиру кода, се налази програм „terminal-font-brightness-map-generator“ који се користи како би се израчунала референтна мапа осветљености за фонт Terminal. Тиме се избегава динамичко рачунање референтне мапе осветљености фонта приликом сваког покретања. Ово представља парадигму познату под називом „метапрограмирање“. Програм је структуриран и направљен тако да по успешно извршеној калкулацији направи дефиницију C++ структуре која представља референтну мапу осветљености и која се може употребити у главном програму који је предмет обраде рада. На слици 3.2 се налази дијаграм једноставног тока програма.



Слика 3.2 Основни ток програма

### 3.1. Фонт приказа

Сам избор фонта је веома битан, и у многоне утиче на крајњи резултат. У овом програму се користи фонт Terminal, који је „monospaced“ растерски фонт. Terminal фонт је присутан на сваком Windows оперативном систему, где је био подразумевани фонт за командну линију на Windows 7 оперативном систему и ранијим верзијама. За креирање ASCII уметности се неретко користе фонтови фиксне ширине, односно „monospaced“

фонтови. Овакви фонтови имају сет карактера који заузимају исту ширину и висину и самим тим су погодни за предвидљив приказ и тачно центрирање исписа на екрану. Изабрана је величина фонта 8x12 пиксела (ширина x висина). Када је у питању величина фонта, што је величина мања приказ може да садржи више карактера на истој површини екрана и самим тим прави већу дефиницију у приказу. Ово може деловати привлачно приликом одабира фонта, али треба имати на уму повећање времена процесирања због обраде више ћелија матрице. Такође, уз премалу величину фонта би се изгубила поента ASCII уметности. На следећој слици (Слика 3.3) се налази фонт Terminal који ће бити коришћен за испис.



Слика 3.3 Сет карактера фонта Terminal 8x12

### 3.2. Иницијализација и провере

Одређене метрике од којих у многоме зависи крајњи резултат обраде су омогућене да се контролишу у виду параметара од стране корисника. Ови параметри нису обавезни да се контролишу зато што су одређене подразумеване вредности већ подешене које генеришу резултате који су задовољавајући у већини случајева. Овим се такође олакшава употреба овог програма. Од параметара који су дозвољени да се контролишу уврштавамо:

- Путања на диску до улазног растерског фајла
- Број карактера у једном реду ASCII уметности
- Ширина скалирања улазног растерског фајла у пикселима
- Корекција гаме, контраста и нивоа осветљења улазног растерског фајла
- Брзи алгоритам/прецизан алгоритам генерисања ASCII уметности

Сваким од ових параметара се такође уводи и потенцијални вектор за грешку услед погрешног уноса корисника који може да изазове непредвиђена понашања програма. Како би се шансе за непредвиђеним понашањима минимизирале, уводе се одређене провере које се изводе пре било каквог вида обраде и претпроцесирања. Провере укључују:

- Подржаност димензије улазног растерског фајла
- Успешност учитавања растерског фајла са диска у меморију програма
- Подржаност типа битмапе за обраду
- Подржаност максималног броја карактера у једном реду ASCII уметности

Ове провере су направљене како би се осигурао неометани даљи ток алгоритма и како би се смањиле шансе за непредвиђене ситуације. Другачији вид провера у току самог извршавања програма су имплементирани како би се кориснику дало до знања шта је то што је потенцијално изазвало грешку али већ у овом тренутку обраде није могуће предвидети и заобићи овакве ситуације.

### 3.3. Претпроцесирање улазног растерског фајла

Програм је дизајниран тако да узима у обзир и могуће мањкавости улазних растерских фајлова те на одговарајући начин одговори на ове проблеме. Овим претпроцесирањем и проверама се минимизира шанса да се догоди непредвиђена околност. Како корисници имају слободан унос растерских фајлова, битно је да се предвиди потенцијална фаличност дефиниције или квалитета растерског фајла и омогући да се на одговарајући начин ове фаличности минимизирају или уклоне. Две врсте мањкавости улазног растерског фајла које су обрађене у програму се огледају на следећи начин:

1. Фаличност боја (бледе боје растерског фајла)
2. Неповољне димензије растерског фајла

Претходно наведене потенцијалне фаличности улазног растерског фајла не морају нужно означавати резултујуће генерисање лоше ASCII уметност, али могу представљати разлог за смањен квалитет у односу на квалитет који би могао бити уколико би се, на одговарајући начин, ове фаличности минимизирале или уклониле.

#### 3.3.1. Претпроцесирање – фаличност боја

Када је у питању фаличност боја, односно бледе боје растерског фајла и недовољан контраст, програм приликом рачунања релативне осветљености ћелија матрице слике додељује вредности које нису довољно диверзне међусобно што резултира са ASCII уметношћу која се састоји од малог сета карактера и нејасно дефинисаним облицима и линијама. Мали сет карактера резултујуће ASCII уметности је проузроковано малим диверзитетом релативне осветљености па самим тим и малим диверзитетом карактера. Како би се одговорило на ову проблематику, кориснику је доступна опција да приликом обраде примени негативну гама корекцију и повећа контраст слике. Негативна гама корекција се односи на потамљивање средњих тонова слике. Количина корекције је фиксна и одговара за већину улазних растерских фајлова.



Слика 3.4 Демонстрација претпроцесирања фаличности боја улазног растерског фајла

### 3.3.2. Претпроцесирање – неповољне димензије растерског фајла

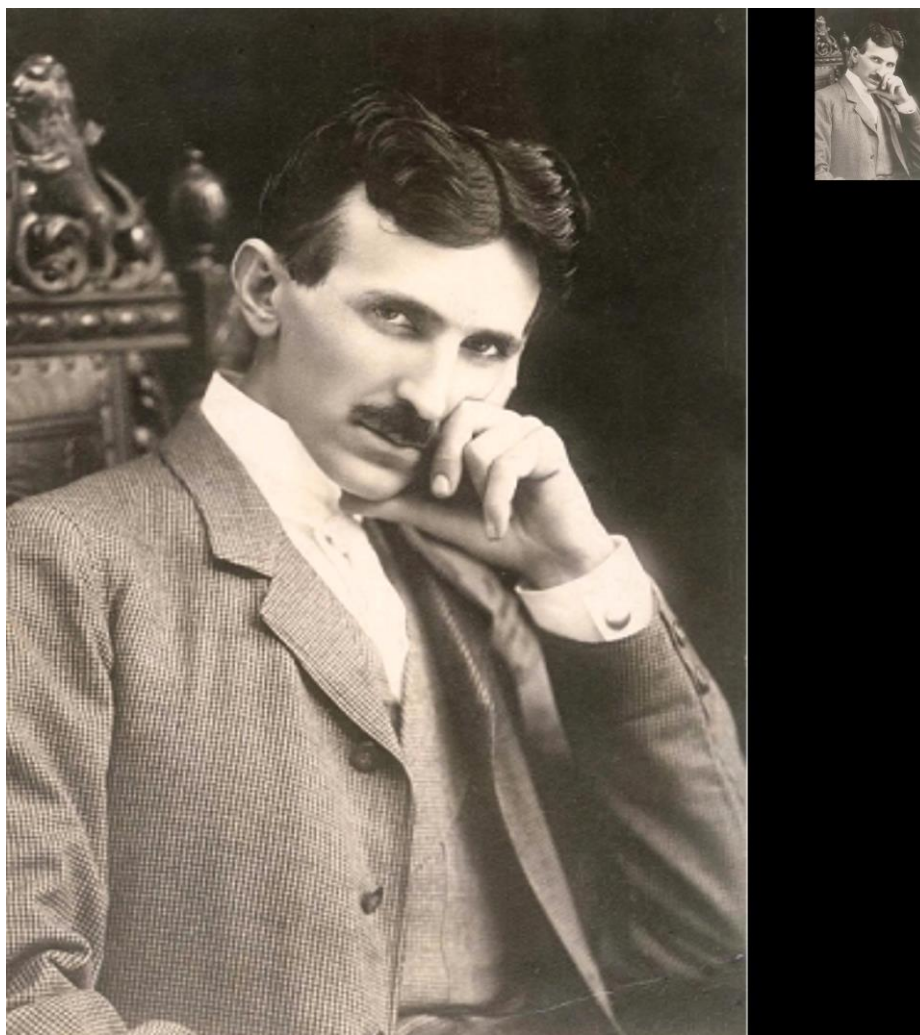
Улазни растерски фајл, попут разних растерских типова датотека, може бити и разних димензија. Ово уводи додатну компоненту и ниво комплексности програма јер су димензије потенцијално променљиве за сваки други улазни растерски фајл. Иако сама променљивост димензија улазног растерског фајла јесте изазов на који се мора одговорити на одговарајући начин, један од изазова је такође и чињеница да непредвидивост димензија компликује алгоритам који је задужен за поделу улазног растерског фајла на матрицу. Број ћелија матрице у ширини је заправо још један од могућих улазних параметара програма (али не и обавезан) и заслужан је заправо за број карактера у једном реду ASCII уметности. Ово означава саму „величину“ ASCII уметности и тиме што је омогућен као параметар смо корисницима омогућили да имају велику контролу на изглед и величину резултујуће обраде.

Први изазов што се тиче димензија улазног растерског фајла јесте неповољна димензија и то може ићи у два екстрема. Први екстрем је превелики фајл који је прескуп за обраду. После одређених димензија растерског фајла нема бенефита у виду јасноће дефиниције резултујуће обраде. Овакав фајл би временски пролонгирао обраду без предности и бенефита. Као противмера је у програму уведена провера да улазни растерски фајл не сме превазилазити максимално дозвољене димензије које се контролишу у изворном коду програма. У овом случају, одлучено је да су ове димензије **четири хиљаде пиксела за висину и ширину** улазног растерског фајла максимално што програм дозвољава за обраду. Овим смо решили један од екстрема. Други екстрем је улазни растерски фајл који нема довољно пиксела услед премалих димензија. Ово резултује више проблема у виду нејасне дефиниције слике па самим тим ни алгоритам не може да поуздано конвертује овакав фајл у ASCII уметност и проблеме у виду неподжане жеље корисника да се фајл конвертује у велику дефиницију ASCII уметности која подразумева велики број карактера у једном реду упркос малој количини пиксела улазног растерског фајла. Како би се одговорило на овакву проблематику, у програму је додат параметар за контролу скалирања улазног растерског фајла у веће или мање димензије како би фајл свели на већи број пиксела, или мањи ако корисник то жели. Овим би алгоритам за поделу фајла на матрицу имао више података за рад зато што би се растерски фајл потенцијално скалирао да садржи више пиксела. Улазни параметар за скалирање је заправо број нове ширине скалираног улазног растерског фајла у пикселима. Алгоритам за скалирање рачуна оригинални однос ширине и висине улазног растерског фајла и на основу овог односа и жељене ширине скалираног растерског фајла за обраду који се прослеђује као параметар се примењује скалирање. Скалирање се примењује коришћењем функционалности из GDI+ API-a.

Једна битна напомена код скалирања је да бенефит у виду генерисања више информација и детаља од оригиналног растерског фајла није могуће. Скалирање служи како би фајл свели на више пиксела како би алгоритам имао више података за рад и како би се решили претходно поменути проблеми. Генерисање више детаља би потпало под парадигму „upscaling“, који се може служити разним алгоритмима и вештачком интелигенцијом али то није предмет обраде овог рада.

Други изазов је ширина ASCII уметности у виду броја карактера у једном реду. Ово је још један од параметара и променљивих и ту наилазимо на проблематику да се жељени број карактера за ширину ASCII уметности не подудара као потпуни делилац ширине улазног растерског фајла. Ово је веома чест случај и проузрокује да обрада ASCII уметност делује изобличено услед неравномерне поделе растерског фајла због неуједначених димензија ћелија матрице. Како би заобишли овај проблем, уведено је заокруживање жељеног броја карактера у једном реду резултујуће обраде на први цели делилац ширине улазног растерског фајла, било да је већи или мањи од траженог. Овим смо обезбедили неометану поделу и жељени исход је омогућен. Постоји више начина које може бити решење ове проблематике, али овакво решење је конкретно и не оставља простора за даље компликације у обради.

На следећој слици (Слика 3.5) је приказан тестни улазни растерски фајл са свим корекцијама које смо обрадили у овој теми (слика лево) наспрам оригиналног улазног растерског фајла (слика десно). Алгоритму је овом обрадом доста олакшана примена и жељена ASCII уметност је генерисана без проблема са јаснијом дефиницијом.



Слика 3.5 Демонстрација претпроцесирања фаличности боја и димензија улазног растерског фајла приказано у сразмерној величини



### 3.4. Генерисање матрице на основу улазног растерског фајла

Након иницијализације подразумеваних вредности, провере улазних параметара, провере подржаности улазног растерског фајла и опционог претпроцесирања долази следећа обавезна фаза алгоритма а то је генерисање матрице на основу улазног растерског фајла. Као што је већ поменуто, идеја је да се улазни растерски фајл подели на матрицу чије ће ћелије да означавају појединачан карактер резултујуће ASCII уметности. Које димензије ће матрица бити диктира опциони улазни параметар који одређује жељену ширину ASCII уметности у виду броја карактера у једном реду. Ово одређује хоризонтални број ћелија матрице. Вертикални број ћелија матрице се аутоматски одређује тако што се рачуна висина једне ћелије матрице којом се дели укупна висина улазног растерског фајла. Висина ћелије матрице се рачуна као производ њене ширине и константе **1.6**. Ова константа потиче од односа између димензије фонта који се користи за испис ASCII уметности и тестирања исписа. Константа је прилагођена за фонт Terminal 8x12 и не би се могла искористити за друге фонтове. Када се у фонту Terminal 8x12 подели висина са ширином добије се приближно ова константа. Следи слика илустрације како би се претпроцесирани улазни растерски фајл поделио на матрицу.



Слика 3.6 Демонстрација креирања матрице улазног растерског фајла на основу жељеног броја карактера у једној линији

Свака од ћелија матрице са слике 3.6 представља појединачни карактер, односно његову квантитативну вредност релативне осветљености. Број ових ћелија у хоризонтали је произвољан и одређује број карактера резултујуће конверзије. Број ових ћелија у вертикали је резултат поделе висине улазног растерског фајла и висине једне ћелије која се рачуна на поменути начин из претходног параграфа. За потребе чувања података о овој матрици је направљена нова структура података и креиран је низ. Свака ћелија је индексирана и приступом јој се може прочитати релативна осветљеност која се може декодирати у карактер за приказ преко референтне мапе претходно израчунате релативне осветљености карактера за фонт који се користи.

Током поделе матрице, може доћи до једног крајњег случаја у вези броја редова матрице у вертикали. Крајњи случај се манифестује када укупна висина улазног растерског фајла није дељива висином ћелије матрице. У овом случају, овакав крајњи случај је обрађен да се при дељењу висине растерског фајла и висине матрице резултат пореди да ли је већи од **35%** висине сваке друге ћелије. Уколико јесте, алгоритам ће узети у обзир и последњи ред ћелија матрице, односно остатак пиксела и третирати их као равноправне матричне ћелије. Уколико је ова висина мања, онда се овај последњи ред ћелија одбацује јер у висини није довољно велики да би имао значајни утицај на изглед крајње конверзије у ASCII уметност. Константа од 35% је експерименталног карактера и означава минимални проценат дела висине матрице како би се узео у обзир.

## 4. АЛГОРИТМИ И СТРУКТУРЕ ПОДАТАКА

Програм за конверзију растерске графике у ASCII уметност инкорпорује основне облике појединих алгоритама које ћемо објаснити у наредним пасусима. Многи принципи рада програма нису сами по себи засновани на утврђеним теоријским принципима, већ заједно у комбинацији постижу хармонијски рад различитих модула. За потребе рада програма направили смо одређене структуре података о којима ће бити више речи са исечцима кода. Главна идеја програма је да се на лак начин омогући конверзија растерске графике у ASCII уметност, и то користећи објектно-орјентисан приступ који на омогућава програмски језик C++. Овим приступом смо такође омогућили веома лаку рефакторизацију кода и његову употребу у већем систему попут кодека за видео снимке који би у реалном времену могао да прикаже цео видео снимак као ASCII уметност. Следи приказ дефиниције класе **ASCIIFied** из фајла `img2ascii.hpp` кроз коју ћемо детаљно проћи.

```
class ASCIIFied
{
public:
    ASCIIFied(CONST WCHAR* image_path, UINT chars_per_line = MEDIUM, UINT scaled_image_width
    = 0, bool gamma_brightness_correction = false, bool edge_detection = false);
    ~ASCIIFied(VOID);

    UINT      get_output_width(VOID) CONST;
    UINT      get_output_height(VOID) CONST;
    Section*  get_section_map(VOID) CONST;
    UINT      get_section_map_size(VOID) CONST;
    VOID      generate_image(VOID);
    VOID      output_to_console(VOID);

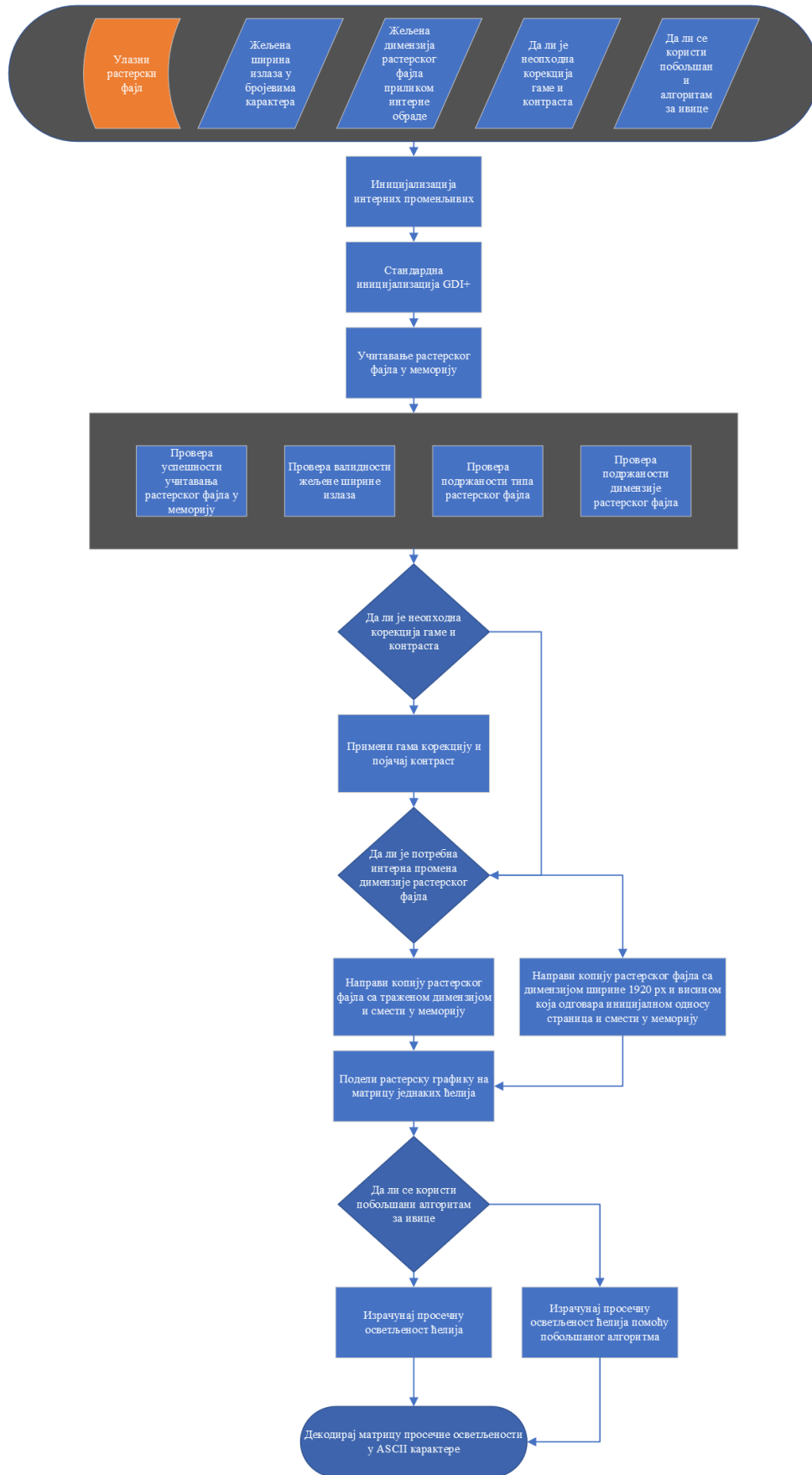
private:
    Bitmap* image;
    CONST WCHAR* image_path;
    UINT chars_per_line;
    UINT scaled_image_width;
    bool gamma_brightness_correction;
    bool edge_detection;
    CHAR* decoded_art;
    Section* section_map;
    ULONG_PTR gdiplus_token;
    UINT section_map_size;
    UINT char_height;

    VOID      resize_image(UINT width);
    INT       find_closest_divider(INT divisor, INT divider);
    VOID      gridify(Bitmap* base);
    VOID      apply_gamma_brightness_contrast_correction(Bitmap* base);
    FLOAT     similarity(CONST FLOAT a[], FLOAT b[]);
    CHAR      brightness_to_char(CONST map<FLOAT, vector<CHAR>>& map, FLOAT brightness);
    CHAR      brightness_map_to_char(const Char map[], FLOAT brightness_map[]);
    bool      is_supported_bitmap_type(VOID);
    bool      is_supported_bitmap_size(VOID);
    VOID      decode_art(VOID);
};
```

Слика 4.0 Декларација класе *ASCIIFied*



Следећа слика представља целокупан дијаграм тока рада програма. Сваки од делова рада програма са слике се сам по себи може издвојити и објаснити. Целокупна логика је енкапсулирана у конструктор класе којој се прослеђују параметри који су објашњени у претходном делу рада. Слика 4.1 ће постати референца по којој ће бити објашњено како програм функционише.



Слика 4.1 Потпун дијаграм тока програма

#### 4.1. Иницијализација и покретање програма

На слици 4.1 као улазни параметри су наведене следеће ставке:

1. Улазни растерски фајл
2. Жељена ширина ASCII уметности изражена у бр. карактера у једном реду
3. Жељена ширина улазног растерског фајла приликом обраде (rescale)
4. Да је ли неопходна корекција гаме или контраста
5. Да ли се користи алгоритам за прецизну детекцију ивица

Како је цео програм енкапсулиран у класу а цела логика је смештена у оквиру конструктора класе, довољно је да корисник просто проследи ове параметре самом конструктору и цела логика ће се одвити у позадини. До ових параметара се може доћи на више начина, а то је остављено као додатни корак имплементатору који користи овај програм. Аргументи могу бити преузети са графичке форме и касније прослеђени конструктору. За потребе рада, програм је имплементиран као конзолни програм коме се аргументи прослеђују приликом самог позива програма кроз терминал. Позив конструктора у овом програму се налази у фајлу main.cpp који је уједно и полазна тачка програма и изгледа овако:

```
ASCIIIFied image(image_path,  
    line_width,  
    scaled_resolution,  
    gamma_contrast_correction,  
    edge_detection);
```

Слика 4.2 Позив конструктора класе ASCIIIFied

Параметри који се прослеђују конструктору на слици 4.2 су декларисани и иницијализовани на следећи начин:

```
INT      line_width = MEDIUM;  
INT      scaled_resolution = SCALED_BITMAP_WIDTH;  
bool     gamma_contrast_correction = false;  
bool     edge_detection = false;  
CONST WCHAR* image_path;
```

Слика 4.3 Декларација и иницијализација параметара конструктора ASCIIIFied

Макрои **MEDIUM** и **SCALED\_BITMAP\_WIDTH** су декларисани у заглављу img2ascii.hpp и служе као подразумеване вредности које се прослеђују у случају да корисник сам не дефинише неке од улазних параметара. Такође, начин на који су дефинисани одговарају већини употребних случајева. Сви дефинисани макрои за потребе програма су следећи:

```
// Macros for predefined character width of ASCII art compatible with 1920px width
#define TINY 96
#define MEDIUM 128
#define LARGE 192

// Macro for scaled bitmap width. Bigger = more detail and SLOWER to process.
#define SCALED_BITMAP_WIDTH 1920

// This defines maximum allowed width and height of input image expressed in pixels
#define MAX_WIDTH 4000
#define MAX_HEIGHT 4000

// This defines maximum allowed character amount in one line for ASCII art
#define MAX_LINE_LENGTH 400

// Section height is at least 1.6 times its width and must be divisible by height of the image.
// Latin alphabet characters are always taller than they are wide.
// 1.6 Works well for font Terminal 8x12px but other values might suit better for different
// fonts
#define FONT_ASPECT_RATIO 1.6f
```

Слика 4.4 Дефиниције макроа програма

Након успешне обраде конструктора следи део кода који приказује генерисану ASCII уметност у конзоли и пита корисника да ли жели да сачува резултат као слику и смести је на десктоп. Уколико се корисник сложи, слика која представља генерисану ASCII уметност са јединственим именом фајла се генерише.

```
image.output_to_console();

cout << endl << endl << "Do you want to generate an image for this art? (Y/N)" << endl;
string input;
cout << "> ";
cin >> input;

if (tolower(input[0]) == 'y' && input.length() < 2)
{
    image.generate_image();
}
```

Слика 4.5 Приказ и чување ASCII уметности – исечак кода

Функција **output\_to\_console()**; је дизајнирана да генерисану уметност испише у терминалу узимајући у обзир сва додатна подешавања приказа у терминалу. Оваква подешавања су величина и тип фонта као и ширина и висина бафера приказа терминала. То представља системски атрибут command prompt-а и мора се подесити како би приказ био добар. Сва неопходна подешавања су нешто што се претходно мора одрадити како би приказ био онакав какав је програм наменио да прикаже уметност и уколико се разликује то може проузроковати изобличен приказ ASCII уметности. Microsoft је својом детаљном документацијом Windows API-а олакшао употребу и подешавање. Следи дефиниција функције за приказ:

```

VOID ASCIIFied::output_to_console(VOID)
{
    CONSOLE_FONT_INFOEX cfi;

    cfi.cbSize = sizeof(cfi);
    cfi.nFont = 0;
    cfi.dwFontSize.X = 8;
    cfi.dwFontSize.Y = 12;
    cfi.FontFamily = FF_DONTCARE;
    cfi.FontWeight = FW_NORMAL;
    wcscpy_s(cfi.FaceName, L"Terminal");

    HANDLE readHandle = GetStdHandle(STD_OUTPUT_HANDLE);

    // Setting font of the console to the optimal one
    SetCurrentConsoleFontEx(readHandle, FALSE, &cfi);

    CONSOLE_SCREEN_BUFFER_INFOEX info{ 0 };
    info.cbSize = sizeof(info);

    GetConsoleScreenBufferInfoEx(readHandle, &info);
    info.srWindow.Right = 499;
    info.srWindow.Bottom = 499;

    // Setting console character width and height properties so that art isn't cut off by
    // smaller screen buffer for console
    SetConsoleScreenBufferInfoEx(readHandle, &info);

    cout << this->decoded_art << endl << endl;
}

```

Слика 4.6 Дефиниција функције за приказ ASCII уметности

Функција која је одговорна за генерисање битмап слике, представљајући ASCII уметност, има значајну улогу у чувању резултата у облику битмап фајла. Овај фајл може бити касније употребљен за приказивање или друге потребе. Први корак ове функције укључује проверу доступности и спремности ASCII уметности за приказ. Затим, додатна радња обухвата добављање апсолутне путање до десктопа и израчунавање димензија у пикселима битмап слике која ће се генерисати. Неизоставни корак у овом поступку укључује и генерисање јединственог имена фајла. Ово се постиже укључивањем датума и времена генерисања у само име фајла, чиме се осигурава да сваки фајл буде јединствено именован како не би дошло до дуплирања фајлова или других непредвиђених околности. Када се добије ова информација, функција користи и Windows API и GDI+ API за чување генерисане слике. Битмап слика је специфично сачувана у PNG формату растера, изабраном због његових карактеристика које укључују задржавање података без губитка и широку подршку на вебу и различитим рачунарским платформама. Ова стратегија чувања слике обезбеђује оптималну доступност и подршку, како на интернету, тако и на различитим типовима рачунара.

```

VOID ASCIIified::generate_image(VOID)
{
    if (this->decoded_art == nullptr)
    {
        cerr << "Failed to produce image from art because it's accessing nullptr"
        << "reference maybe due to failed art generation or art isn't decoded"
        << " yet.\nTerminating the program." << endl;
        exit(EXIT_FAILURE);
    }

    // In pixels, each letter is 8x12px + 1px for vertical separation + some breathing space
    // (char_height = vertical amount of characters in art, chars_per_line = same but on
    // horizontal axis)
    INT image_width = this->chars_per_line * 8 + 100;
    INT image_height = this->char_height * 13 + 100;

    // Generate bitmap to draw to
    Bitmap art_output(image_width, image_height);
    Graphics g(&art_output);

    // Sets background to black
    Color color(Color::Black);
    g.Clear(color);

    // Creating proper font
    HFONT hfont = CreateFont(-12, -8, 0, 0, FW_NORMAL, FALSE, FALSE, FALSE, OEM_CHARSET,
    OUT_RASTER_PRECIS, CLIP_DEFAULT_PRECIS, PROOF_QUALITY, DEFAULT_PITCH, L"Terminal");

    // Text bounding rectangle
    RECT rc = { 50, 50, image_width, image_height };
    HDC hdc = g.GetHDC();

    // Setting text background and foreground color to console colors along side with font
    SetBkColor(hdc, RGB(0, 0, 0));
    SetTextColor(hdc, RGB(210, 210, 210));

    // Apply font
    SelectObject(hdc, hfont);

    DrawTextA(hdc, this->decoded_art, strlen(this->decoded_art), &rc, DT_LEFT | DT_TOP |
    DT_WORDBREAK);

    g.ReleaseHDC(hdc);
    DeleteObject(hfont);

    WCHAR path[MAX_PATH];

    // Get Desktop path
    if (SHGetFolderPathW(NULL, CSIDL_DESKTOPDIRECTORY, NULL, 0, path) != S_OK)
    {
        cerr << "DESKTOP path not found!\n";
        exit(EXIT_FAILURE);
    }

    SYSTEMTIME time_s;
    GetSystemTime(&time_s);

    wstring day = to_wstring(time_s.wDay);
    wstring month = to_wstring(time_s.wMonth);
    wstring year = to_wstring(time_s.wYear);
    wstring hour = to_wstring(time_s.wHour);
    wstring minute = to_wstring(time_s.wMinute);
    wstring second = to_wstring(time_s.wSecond);

    wstring image_path(path);
    image_path += L"\\ascii_art_" + day + L"-" + month + L"-" + year + L"-" + hour + L"-" +
    minute + L"-" + second + L".png";

    CLSID encId =
    { 0x557cf406, 0x1a04, 0x11d3, { 0x9a, 0x73, 0x00, 0x00, 0xf8, 0x1e, 0xf3, 0x2e } };
    art_output.Save(image_path.c_str(), &encId);

    cout << "Image saved to your desktop." << endl;
}

```

Слика 4.7 Дефиниција функције за чување ASCII уметности у битмап фајл

## 4.2. Конструктор класе и имплементација програма

```

ASCIIIFied::ASCIIIFied(CONST WCHAR* image_path, UINT chars_per_line, UINT scaled_image_width,
                      bool gamma_brightness_correction, bool edge_detection)
{
    this->image_path = image_path;
    this->chars_per_line = chars_per_line;
    this->scaled_image_width = scaled_image_width;
    this->gamma_brightness_correction = gamma_brightness_correction;
    this->edge_detection = edge_detection;
    this->image = nullptr;
    this->decoded_art = nullptr;
    this->section_map = nullptr;
    this->gdiplus_token = 0;
    this->section_map_size = 0;
    this->char_height = 0;

    // Initialize GDI+
    GdiplusStartupInput gdiplusStartupInput;
    GdiplusStartup(&this->gdiplus_token, &gdiplusStartupInput, NULL);

    this->image = new Bitmap(this->image_path);

    if (this->image->GetLastStatus() != Ok)
    {
        cerr << "Error opening the requested image file at location [ "
              << this->image_path << " ]. Terminating the program." << endl;
        exit(EXIT_FAILURE);
    }

    if (this->chars_per_line > MAX_LINE_LENGTH)
    {
        cerr << "Number of characters per single line provided for ASCII art is bigger"
              << "than allowed, which is " << MAX_LINE_LENGTH << ". Please provide a smaller"
              << "number." << endl;
        exit(EXIT_FAILURE);
    }

    if (!this->is_supported_bitmap_type())
    {
        cerr << "File type provided is different than supported. The supported file"
              << "types are: jpg, png, bmp and tiff." << endl;
        exit(EXIT_FAILURE);
    }

    if (!this->is_supported_bitmap_size())
    {
        cerr << "File width or height exceeds allowed amount. Current allowed dimension"
              << "(width x height) are: " << MAX_WIDTH << " x " << MAX_HEIGHT << endl;
        << "Please choose a smaller image." << endl;
        exit(EXIT_FAILURE);
    }

    // Scale the input image to the desired width in pixels preserving original aspect ratio
    // It will not resize the input image if 0 is provided as scaled image width
    if (this->scaled_image_width != 0)
    {
        this->resize_image(this->scaled_image_width);
    }

    if (this->gamma_brightness_correction)
    {
        this->apply_gamma_brightness_contrast_correction(this->image);
    }

    // Calculates a suited grid for scaled bitmap and populates each section with
    // corresponding data!
    this->gridify(this->image);

    // Decode the art from section map data
    this->decode_art();
}

```

Слика 4.8 Дефиниција конструктора класе ASCIIIFied

На слици 4.8 се налази имплементација конструктора кога смо већ поменули. У овој функцији се налази целокупна логика програма која је задужена за прихватање улазних параметара, конверзију улазног растерског фајла у ASCII уметност и сваку другу врсту обраде која се дешава. Сходно слици 4.1 у конструктору можемо редом уочити иницијализацију интерних променљивих, стандардну иницијализацију GDI+ API-а, учитавање растерског фајла у меморију програма, провере валидности података и растера, скалирање улазног растерског фајла уколико има потребе, примену корекције гаме и контраста уколико има потребе, поделу на матрицу и примену функције за конверзију у ASCII уметност. Целокупна логика се дешава у самом конструктору приликом креирања инстанце, односно објекта класе. Сходно томе је веома лако замислити како би овакав приступ могао да се на један лак начин примени над низом растера код којих би се одрадила конверзија на брз серијски начин.

### 4.3. Напредан алгоритам конверзије

У току овог рада смо поменули напредан алгоритам за конверзију улазног растера у ASCII уметност. Ово је такође и последњи параметар у конструктору класе:

```
ASCIIIFied::ASCIIIFied(CONST WCHAR* image_path,  
    UINT chars_per_line,  
    UINT scaled_image_width,  
    bool gamma_brightness_correction,  
    bool edge_detection)
```

Слика 4.9 Декларација конструктора класе ASCIIIFied

Параметар „edge\_detection“ одређује да ли корисник жели да програм користи напреднији и комплекснији алгоритам који генерише прецизнији резултат по цену већег процесорског времена. Напредни алгоритам конверзије се у многome заснива на основној верзији конверзије, и то по томе што улазни растерски фајл дели на матрицу чије се ћелије касније конвертују у своју репрезентацију ASCII карактера. Оно где се алгоритам разликује од своје основне верзије је то да се свака од ћелија матрице дели на девет неједнаких парцела и то по тачно дефинисаним димензијама. Свака парцела има засебну релативну осветљеност у оквиру ћелије матрице. Када је у питању структура података којом би се овакав податак представио, свака ћелија матрице би заправо била вектор вредности од девет чланова. Ово захтева да се референтна мапа претходно израчунатих вредности релативне осветљености за сваки карактер фонта који се користи калкулише на исти начин и као таква да се користи где би се калкулација избегла при самом покретању програма. Односно, сваки карактер фонта који се користи се мора поделити у девет неједнаких али прецизно одређених парцела, свака од којих би имала квантитативну одредбу релативне осветљености. Алгоритам би у овом случају поредио сваку ћелију матрице, односно вектора са девет чланова, са низом вектора који би представљали референтну мапу. Како би ово најпростије описали, своди се на проналажење најсличнијег вектора из скупа задатом вектору. Сличност два вектора није једноставна тематика и зависи по којим критеријумима се тражи сличност, али



емпиријски је у овом случају закључено да Еуклидска удаљеност даје задовољавајуће резултате. Имплементирано Еуклидска удаљеност дефинише корен квадратног збира свих секција ћелије матрице која представља квантитативни износ који се може упоредити са другим износом који је израчунат на идентичан начин. У овом случају, поређење подразумева најмању разлику између две овакве вредности.

```

FLOAT ASCIIified::similarity(CONST FLOAT a[], FLOAT b[])
{
    // Implementation of euclidean distance

    FLOAT powSum = 0;
    FLOAT similarity = 0;

    for (INT i = 0; i < 9; i++)
    {
        powSum += pow((a[i] - b[i]), 2.f);
    }

    similarity = sqrt(powSum);

    return similarity;
}

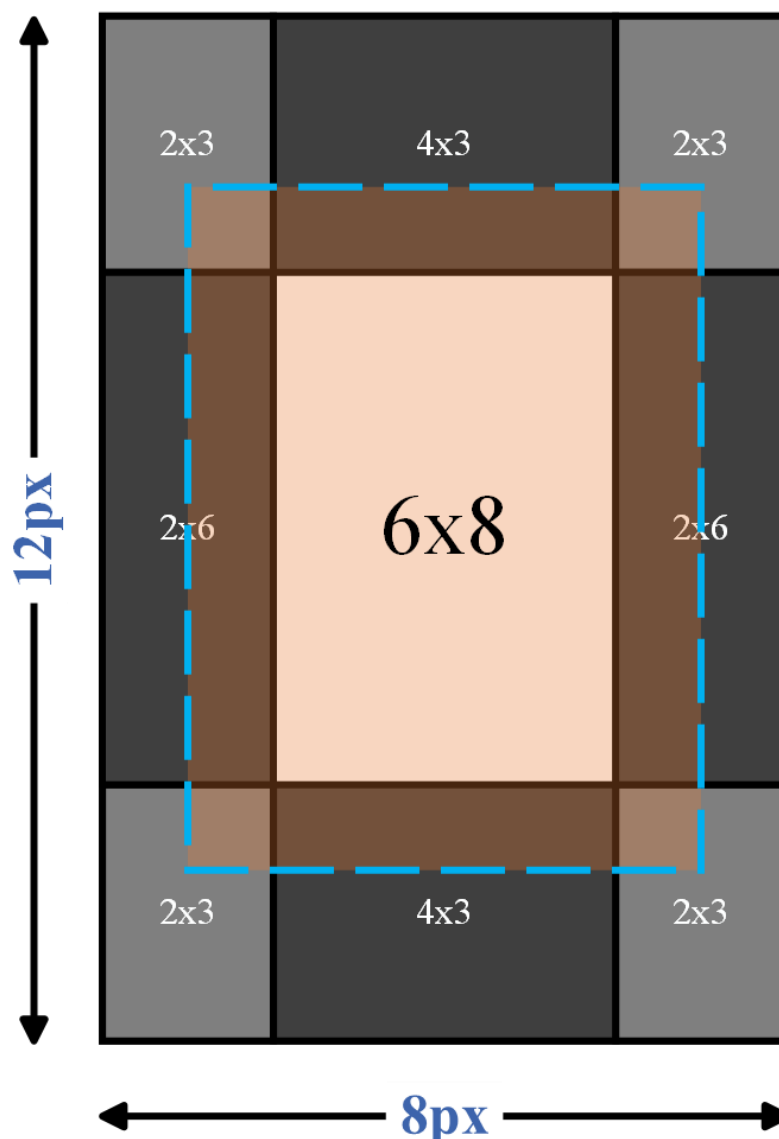
```

Слика 4.10 Имплементација Еуклидске раздаљине

Код овако имплементиране функције је веома лако мапирати векторе ћелије матрице са најсличнијим вектором из скупа вектора сваки од којих представљају карактер из ASCII сета.

#### 4.3.1. Алгоритам генерисања вектора релативне осветљености ћелије

До сада смо поменули како напредни алгоритам конверзије функционише на једном апстрактном нивоу. Када је у питању само генерисање вектора, односно низа, релативне осветљености матрице подељеног улазног растерског фајла и поређења са векторима релативне осветљености сета карактера који сачињавају ASCII уметност, одлучено је за сет од деведесет и пет карактера. Сваки од ових карактера су представљени вектором од девет чланова. Како је избор фонта Terminal 8x12, узимајући ту величину у обзир, неједнаке парцеле су израчунате где централна парцела заузима највећу тежину вредности. Следи подела карактера односно ћелије матрице улазног растерског фајла на поменуте неједнаке парцеле:

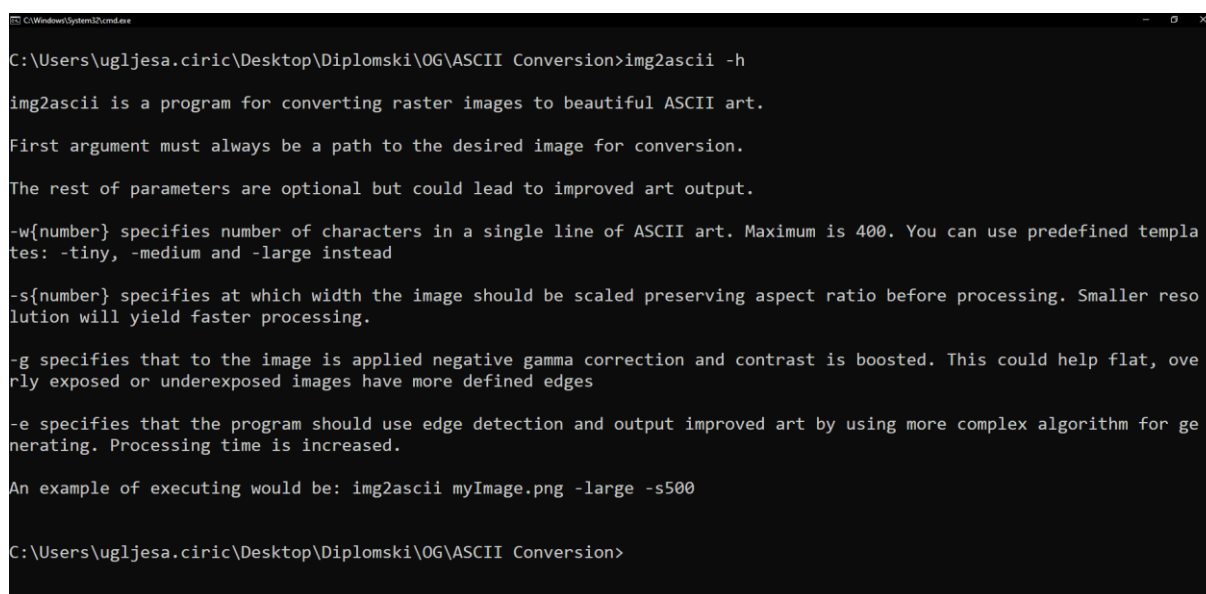


Слика 4.11 Шема поделе карактера/ћелије матрице на парцеле

На слици 4.11 је приказана шема поделе карактера фонта који се користи за приказ ASCII уметности на парцеле, узимајући у обзир тачну димензију фонта. Сваки од димензија парцела се може изразити процентуално сходно целокупној величини карактера и такве процентуалне димензије се користе како би се ћелије матрице ASCII уметности поделиле на исти начин. Разлог томе је зато што није могуће знати димензије ћелије матрице пре покретања програма јер те димензије зависе од више фактора попут броја карактера приказа уметности у једном реду и димензије самог улазног растерског фајла, односно скалираног растерског фајла. Како је неопходно израчунати овакав вектор за сваку ћелију матрице а потом такве векторе поредити са референтном мапом, то може бити процесорски интензивно али је крајњи резултат конверзије уочљиво бољи.

## 5. УПОТРЕБА И РЕЗУЛТАТИ

Сходно имплементацији програма у језику C++ и компајлирању кода у један монолитни извршни бинарни фајл, кроз конзолни приказ кориснику су доступне команде као и кратко упутство коме се може приступити уколико корисник жели да сазна или погледа детаљније како се програм користи и за шта служе параметри који се могу одредити. Програму нису потребни екстерни фајлови од којих би његово извршавање зависило и представља једноставну имплементацију контроле улазних параметара поменуте класе ASCIIFile помоћу корисничког уноса у конзоли приликом покретања програма. На следећој слици се може видети екран како програм иницијално изгледа када се позове само по имену из терминала и како изгледа када се позове са опционим опционим параметром „-h“ који служи за приказ упутства за употребу.



```
C:\Windows\System32\cmd.exe
C:\Users\ugljesa.ciric\Desktop\Diplomski\06\ASCII Conversion>img2ascii -h

img2ascii is a program for converting raster images to beautiful ASCII art.
First argument must always be a path to the desired image for conversion.

The rest of parameters are optional but could lead to improved art output.

-w{number} specifies number of characters in a single line of ASCII art. Maximum is 400. You can use predefined templates: -tiny, -medium and -large instead

-s{number} specifies at which width the image should be scaled preserving aspect ratio before processing. Smaller resolution will yield faster processing.

-g specifies that to the image is applied negative gamma correction and contrast is boosted. This could help flat, overly exposed or underexposed images have more defined edges

-e specifies that the program should use edge detection and output improved art by using more complex algorithm for generating. Processing time is increased.

An example of executing would be: img2ascii myImage.png -large -s500

C:\Users\ugljesa.ciric\Desktop\Diplomski\06\ASCII Conversion>
```

Слика 5.1 Иницијално покретање и упутство програма

### 5.1. Употреба програма

На слици 5.1 је приказано како иницијално изгледа терминал приликом покретања програма без параметара као и испис једноставног упутства употребом опционог параметра „-h“.

Упутство дефинише употребу опционог параметра „-w“ који одређује број карактера у једном реду исписа резултујуће конверзије улазног растерског фајла као и ограничење у виду маскималног броја који се може дефинисати. Овај број се може контролисати кроз изворни код и предмет је провере приликом покретања програма како смо објаснили у претходним тачкама. Поред конкретног броја, корисник може искористити један од три предефинисана шаблона tiny, medium и large. Ови макрои представљају замену за ширине деведесет шест, сто двадесет осам и сто деведесет два редом. Ови бројеви су циљано дефинисани тако из разлога што су потпуно дељиви са бројем хиљаду деветсто двадесет. То је ширина „full HD“ резолуције која је такође узета да буде предефинисана подразумевана ширина за скалирање улазног растерског фајла,

уколико корисник сам не дефинише своју жељену ширину скалирања. Програм има мање потешкоћа приликом рачунања уколико су бројеви дељиви на овај начин. Уколико корисник сам не проследи овај параметар, подразумевано се користи макро „medium“, односно ширина од сто двадесет осам карактера.

Опциони параметар „-s“ који се такође дефинише у упуству означава жељену ширину скалирања улазног растерског фајла у пикселима. Некада улазни растерски фајл има премало или превише пиксела и неопходно га је скалирати како би програм имао довољно података за обраду. Уколико се овај параметар не дефинише, програм је подешен да користи ширину од хиљаду деветсто двадесет пиксела, и сходно овој ширини пропорционално скалира висину улазног растерског фајла како не би дошло до изобличавања. Треба имати на уму да ако се стави превелика ширина, програму ће требати више процесорског времена за обраду јер има више пиксела за калкулацију, док премали број резултује са недовољним бројем пиксела односно информације за обраду и може доћи до изобличене конверзије у најбољем случају а у најгорем до непредвиђеног понашања програма. Програм је подешен да уколико корисник проследи нулу као одредбу за параметар „-s“, онда програм неће скалирати улазни растерски фајл.

Опциони параметар „-g“ дефинише да ли се током процесирања треба применити негативна гама корекција и да ли се контраст треба појачати приликом препроцесирања улазног растерског фајла. Оваква предобрада помаже лоше осветљеним сликама и сликама са премалим диверзитетом боја да дођу до изражаја и тиме побољша резултат конверзије због јасније дефинисаних линија и ивица овако обрађеног улазног растерског фајла. Подразумевана вредност је да се ова корекција не користи, али једноставним навољењем овог аргумента корисник може да укључи овај вид претпроцесирања.

Опциони параметар „-e“ дефинише употребу напредног алгоритма приликом конверзије. Као што је већ објашњено, програм располаже са два алгоритма која могу да обраде улазни растер и конвертују га у ASCII уметност. Један алгоритам је једноставнији и ограниченији у виду диверзитета карактера који се користе за приказ резултујуће конверзије док навођењем параметра корисник може укључити комплекснији алгоритам који ће конвертовати улазни растерски фајл користећи напредније технике и већи диверзитет ASCII карактера за приказ резултујуће конверзије. Подразумевано, уколико корисник не наведе параметар „-e“, користиће се основна верзија алгоритма.

Сваки од ових параметара су опциони и произвољни. Нису неопходни да се наведу како би програм радио због подразумеваних вредности које су дефинисане у изворном програму. Најједноставнији облик команде која позива и конвертује улазни растерски фајл би изгледала овако: „img2ascii NikolaTesla.png“. Оваквим позивом, програм би покренуо конверзију улазног растерског фајла под именом „NikolaTesla.png“ са пропорционалним скалирањем на ширину од хиљаду деветсто двадесет пиксела, ширином приказа резултујуће конверзије ASCII уметности од сто двадесет осам карактера, без примене предобраде и применом основне верзије алгоритма конверзије.

## 5.2. Резултати конверзије

Узимајући у обзир програм као једну готову целину која се у овом тренутку може употребити са претходним објашњењима како се програм употребљава из перспективе обичног корисника, у овом делу рада ћемо приказати резултате конверзије са различитим подешавањима које програм нуди. За резултат, корисници ће моћи да визуелно погледају разлику између два претходно описана алгорита за конверзију растерске графике у ASCII уметност, као и утицај других параметара на визуелни резултат конверзије. Следећа слика представља улазни параметар програма који није манипулисан ни на један други начин, осим оних објашњених у овом раду приликом обраде и конверзије фајла.



Слика 5.2 - Улазни растерски фајл за конверзију у демонстрацији

На слици 5.2 је представљен улазни растерски фајл за демонстрацију конверзије и као такав је прослеђен програму. На следећој слици ће бити представљен резултат конверзије између основног алгорита без додатних побољшања програма приликом претпроцесирања и напредног алгорита са свим претпроцесирањем која ће бити укључена, односно побољшање фаличности боја и гама корекције. Компарација ће иницијално бити направљена на резултат обраде ASCII уметности која у ширини има дефинисан број карактера од двадесет, тридесет и педесет. Програм је покренут из директоријума у коме се налази бинарни фајл програма, као и улазни растерски PNG фајл који је приказан на слици 5.2.

Овако је програм покренут:

```
img2ascii.exe NikolaTesla.png -w20
```

```
img2ascii.exe NikolaTesla.png -w30
```

```
img2ascii.exe NikolaTesla.png -w50
```

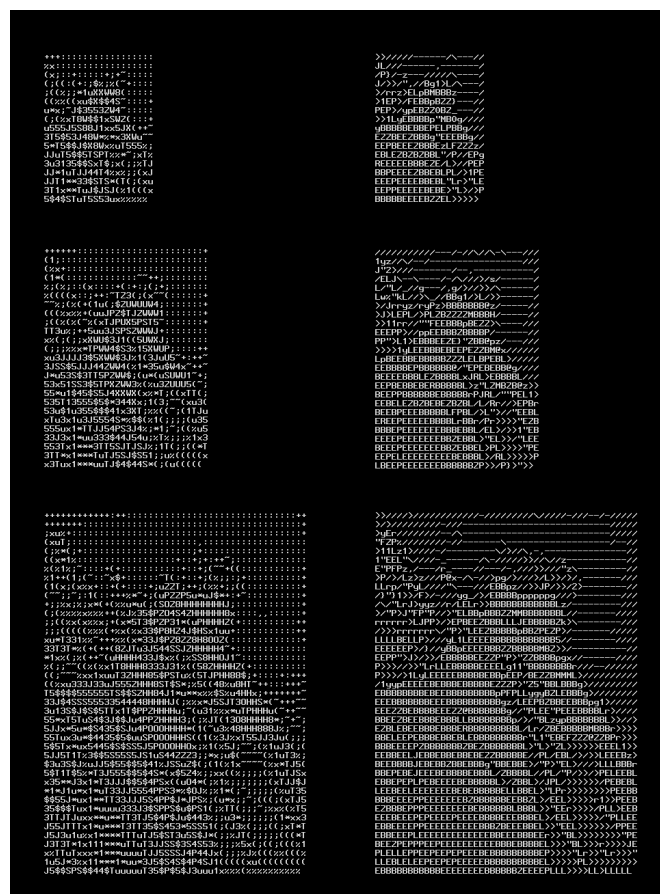
```
img2ascii.exe NikolaTesla.png -w20 -e -g
```

```
img2ascii.exe NikolaTesla.png -w30 -e -g
```

```
img2ascii.exe NikolaTesla.png -w50 -e -g
```

Слика 5.3 - Позивање програма за обраду

Резултати овакве обраде се налазе на слици 5.4. Лева колона представља прве три команде извршене на примеру 5.3 док десна колона приказује последње три команде из примера 5.3. Разлика је у алгоритму обраде и укљученим претпроцесирањем фајла приликом конверзије. Оба алгоритма подразумевају иницијално скалирање улазног растерског фајла на ширину од хиљаду деветсто двадесет пиксела.



Слика 5.4 Поређење обраде два алгоритма – 20, 30, 50 карактера

На слици 5.4 су приказани резултати обраде улазног растерског фајла користећи основни и напредни алгоритам са претпроцесирањем. Ширина ASCII уметности обраде је редом двадесет, тридесет и педесет карактера. Чак и на овако малим величинама се јасно види разлика и визуелна дефинисаност напредног алгоритма. На следећој слици ће бити приказан резултат обраде у већој дефиницији:



Слика 5.5 - Поређење обраде два алгоритма - medium preset

На слици 5.5 је приказан резултат обраде улазног растерског фајла са искоришћеним шаблоном „medium“, који је објашњен у поглављу 5.1 овог рада. У овој постави је визуелно евидентна разлика коју пружа напредан алгоритам обраде али уметнички који је бољи је тешко закључити и то зависи од посматрача. Обрада је покренута следећим кодом:

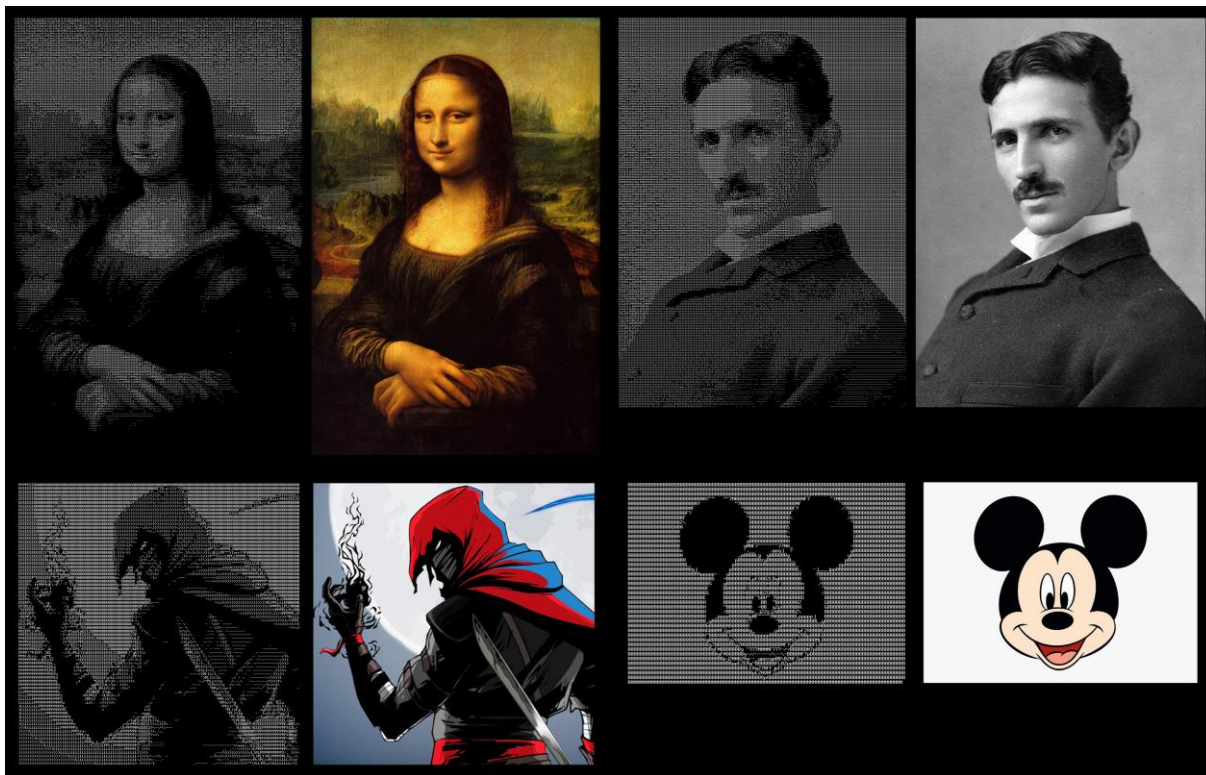
```
img2ascii.exe NikolaTesla.png -medium
```

```
img2ascii.exe NikolaTesla.png -medium -e -g
```

Слика 5.6 - Позивање програма за обраду са шаблоном medium



Програм се такође добро показао и у обради растерске графике која је дигитално направљена, односно она која није усликана фотоапаратом. У наставку су примери такве конверзије која користи напредни алгоритам конверзије са омогућеним претпроцесирањем.



Слика 5.7 - Примери конверзије са напредним алгоритмом и претпроцесирањем

Слика 5.7 представља добар пример који показује да програм није ограничен само на конверзију фотографија које су усликане фотоапаратом, већ слика које су насликане, као и дигитално креиране слике које варирају од комплексности до разних палета боја које се обрађују.



## 6. ЗАКЉУЧАК

img2ascii је програм који трансформише растерску графику у ASCII уметност користећи објектно-оријентисани приступ и програмски језик C++ који то омогућава. Програм прима улазне параметре као што су произвољна растерска датотека, жељени број знакова у једном реду у приказу ASCII уметности, жељену ширину улазне растерске датотеке приликом обраде при скалирању, и информацију да ли је потребна корекција гаме или контраста. Алгоритам програма дели улазну растер датотеку у матрицу чије ћелије се касније конвертују у своју репрезентацију ASCII знакова. Програм користи основни алгоритам за ову сврху, али такође нуди и напредни алгоритам конверзије који дели сваку ћелију матрице на девет неједнаких парцела са прецизно дефинисаним димензијама ради постизања прецизнијих резултата, уз дуже време обраде. Програм такође укључује функције за приказ и чување генерисане ASCII слике у виду растерске датотеке.

Логика извршавања конверзије је имплементирана кроз објектно-оријентисани модел приступа решењу, где се већина логике дешава у главној класи, односно конструктору класе. Апликација користи GDI+ API и Windows API за управљање графичким елементима у оперативном систему Windows и приступу фајловима. Windows API је саставни део Windows оперативног система и доступан је у свим његовим верзијама.

Општа идеја је да се улазна растерска датотека подели у матрицу где би свака ћелија имала једнаке димензије, и то у зависности од параметра који одређује број карактера ASCII уметности за приказ у једном реду. Избор фонта има значајан утицај на крајњи резултат, а за приказ се користи Terminal фонт. Алгоритма за генерисање ASCII уметности из растерских слика узима у обзир потенцијалне грешке у уносу корисника и спроводи провере како би се обезбедила неометана обрада са смањеним шансама за непредвиђене ситуације. Предобрада је такође имплементирана како би се решиле евентуалне фалиčnosti боја и неповољне димензије улазне растер датотеке која подразумева примену корекције контраста и гаме улазног растерског фајла као и његовог скалирања у одговарајућу величину која је један од параметара програма.

Алгоритам генерише матрицу на основу улазне растер датотеке, са бројем ћелија у хоризонтално одређеним опционим улазним параметром, и бројем ћелија у вертикали израчунатим на основу висине једне ћелије и укупне висине улазног растера. Добијена матрица представља појединачне карактере ASCII уметности, при чему свака ћелија садржи квантитативну вредност релативне осветљености. Алгоритам омогућава корисницима контролу над изгледом и величином резултујућег рендеровања.

У закључку, img2ascii је моћан алат за претварање растерске графике у ASCII уметност. Користи објектно-оријентисани приступ омогућен програмским језиком C++ и пружа корисницима контролу над изгледом и величином резултујућег рендеровања.

## 7. ИНДЕКС ПОЈМОВА

### А:

ASCII уметност 2, 4, 7, 8, 9, 10, 11, 12, 13, 14, 15, 18, 19, 20, 23, 24, 25, 27, 28, 30, 32

Алгоритам генерисања вектора релативне осветљености 24

API 5, 11, 19, 20, 23, 32

### Б:

Битмап слика 4, 9, 20

### Г:

Графика 2, 28

GDI+ API (Graphics Device Interface) 4, 5, 11, 20, 23, 26, 32

### Д:

Димензије 7, 8, 9, 10, 11, 13, 25, 26, 32

### Е:

Еуклидска удаљеност 24

### И:

Имплементација 22, 23, 24, 26, 32

Иницијализација 9, 13, 18, 23

### К:

Конверзија 4, 15, 23, 26, 27, 28, 29, 31, 32

Конструктор 4, 16, 18, 19, 23, 26

Корекција 9, 10, 12, 18, 26, 27, 28, 32

### Л:

Логика 4, 16, 18, 23, 26, 32

### О:

Објектно-оријентисани 2, 4, 15, 26, 32

### П:

Парцела 23, 24, 25, 26, 32

Предобрада 26, 32

Приказ 4, 5, 8, 9, 12, 14, 15, 19, 20, 25, 26, 27, 28, 29, 30, 32

### Р:

Растер 2, 4, 5, 7, 8, 9, 10, 11, 12, 13, 14, 15, 18, 20, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32

### С:

Сет 9, 10, 24

Скалирање 9, 11, 23, 25, 26, 27, 29, 32

Структура података 4, 5, 14, 23

### Т:

Terminal фонт 8, 9, 13, 24, 26

Ток 4, 6, 8, 9

### Ф:

Фајл 4, 7, 9, 10, 11, 12, 13, 14, 15, 18, 19, 20, 23, 24, 25, 26, 27, 28, 29, 32, 32

Фонт 4, 7, 8, 9, 13, 14, 19, 23, 24, 25

### Ц:

C++ 4, 5, 6, 8, 15, 32

W: Windows API 5, 6, 19, 20, 32

## 8. ЛИТЕРАТУРА

- [1] Microsoft Official Documentation - GDI+ [GDI+ - Win32 apps | Microsoft Learn](#)
- [2] Microsoft Wiki -Windows API [Windows API | Microsoft Wiki | Fandom](#)
- [3] educative - How to compute Euclidean distance [How to compute Euclidean distance in C# \(educative.io\)](#)
- [4] Michal Franc - LockBits vs Get Pixel Set Pixel – Performance [LockBits vs Get Pixel Set Pixel - Performance | Michal Franc \(mfranc.com\)](#)
- [5] Stack Overflow - Image to ASCII art conversion [c++ - Image to ASCII art conversion - Stack Overflow](#)
- [6] Stack Overflow - Formula to determine perceived brightness of RGB color [image - Formula to determine perceived brightness of RGB color - Stack Overflow](#)
- [7] Wikipedia - ASCII Art [ASCII art - Wikipedia](#)
- [8] Wikipedia - Relative Luminance [Relative luminance - Wikipedia](#)
- [9] Wikipedia - Row- and column-major order [Row- and column-major order - Wikipedia](#)
- [10] YouTube, daivuk (channel) - Image to ASCII experimentations [Image to ASCII experimentations - YouTube](#)

## 9. ИЗЈАВА О АКАДЕМСКОЈ ЧЕСТИТОСТИ

### ИЗЈАВА О АКАДЕМСКОЈ ЧЕСТИТОСТИ

Студент (име, име  
једног родитеља и презиме):

Угљеша Дејан Ћирић

Број индекса:

РТ-8/18

Под пуном моралном, материјалном, дисциплинском и кривичном одговорношћу изјављујем да је завршни рад, под насловом:

Програм за конверзију растерске графике у ASCII уметност

---

1. резултат сопственог истраживачког рада;
2. да овај рад, ни у целини, нити у деловима, нисам пријављиво/ла на другим високошколским установама;
3. да нисам повредио/ла ауторска права, нити злоупотребио/ла интелектуалну својину других лица;
4. да сам рад и мишљења других аутора које сам користио/ла у овом раду назначио/ла или цитирао/ла у складу са Упутством;
5. да су сви радови и мишљења других аутора наведени у списку литературе/референци који је саставни део овог рада, пописани у складу са Упутством;
6. да сам свестан/свесна да је плагијат коришћење туђих радова у било ком облику (као цитата, прарафаза, слика, табела, дијаграма, дизајна, планова, фотографија, филма, музике, формула, вебсајтова, компјутерских програма и сл.) без навођења аутора или представљање туђих ауторских дела као мојих, кажњиво по закону (Закон о ауторском и сродним правима), као и других закона и одговарајућих аката Високе школе електротехнике и рачунарства струковних студија у Београду;
7. да је електронска верзија овог рада идентична штампаном примерку овог рада и да пристајем на његово објављивање под условима прописаним актима Високе школе електротехнике и рачунарства струковних студија у Београду;
8. да сам свестан/свесна последица уколико се докаже да је овај рад плагијат.

У Београду, 11.03.2024. године

Својеручни потпис студента

---