Programozói dokumentáció

Programozás alapjai 1. - Nagy házi

Mészáros Anna Veronika (I8SQUE), 2023

1. A programról

Az ebben a programban megvalósított játék a négyszín-tételen alapszik: egy tetszőleges mezőkre osztott síkot, ki lehet úgy színezni legfeljebb négy szín felhasználásával, hogy ne legyen két azonos színű szomszédos mező. A játékos feladat a megadott négy színnel minél hamarabb ilyen módon kiszínezni egy folyamatosan mozgó térképet.

Minden mezőt egyetlen ponttal jellemez a program (a mező középpontja / “vertex”). Ennek a pontnak a segítségével tároljuk el a mező színét, sebességét, irányát. A pontból a mezőket a Voronoi diagram szabályai alapján tudjuk meghatározni.

A játék kezdeti állapotban minden mezőt egy semleges színre állít, a játékos csak a játék elindítása után színezi ki a mezőket. Azt, hogy megfelelően van-e kiszínezve a térkép csak akkor ellenőrizzük, ha már egyetlen mező sem semleges színű.

Amikor a játék véget ért, a program elmenti a játékos eredményét egy, a játék nehézségi szintjének megfelelő, fáljba, ahol minden eddigi játékos eredményét tárolja. Ezeket az eredményeket a dicsőséglistán tudja a felhasználó megtekinteni.

A játékos bármikor indíthat új játékot, ilyenkor megadhat egy felhasználónevet és kiválaszthatja az új játék nehézségét.

A program mindezt modulokra bontva valósítja meg, a további fejezetekben ezeknek a moduloknak a leírása található.

1.1. A program fordítása és futtatása

A programnak szüksége van az SDL, SDL2_gfxPrimitives és az SDL_ttf külső könyvtárakra. A program makefile-ának segítségével a program könnyedén lefordítható gcc fordítóval. A létrejövő game állományt a forrásfájlokkal megegyező mappában kell elindítani, mivel a program relatív eléréssel hivatkozik az ott lévő fájlokra. Ezen felül ennek a mappának a program által írhatónak kell lennie, különben a program nem tudja frissíteni a dicsőséglistákat.

2. Main - main.c

A program fő modulja, a program indításakor inicializálja a játék állapotát, illetve a program végén felszabadítja a dinamikus memóriaterületre mutató pointereket. Ezen kívül futtatja az event loopot, és minden érkező eseményt átad az eseménykezelőnek. Meghívja a képernyőkirajzoló függvényt, illetve a mezők mozgatására szolgáló függvényt.

2.1. Függvények

initButtons - void initButtons(BtnList *btns)

- feltölti a kapott gomblistát az alapértemezett gombokkal
- paraméter(ek):
  - egy BtnList pointer, melynek a list mezőjébe egy dinamikusan foglalt memóriaterületre mutató pointert ír a függvény, ezt a hívónak kell felszabadítania

initialize - void initialize(State *state, Objects *objects)

- inicializálja a state és objects struktúrákat
- paraméter(ek):
  - state és object struktúrára mutató pointer, melynek mezői közül a függvény kitölti az objects->btns->list és a objects->vertice->list dinamikusan foglalt memóriaterületre mutató pointer, ezt a hívónak kell felszabadítania

main - int main(void)

- meghívja az inicializáló függvényeket és futtatja az event loopot
- visszatérési érték: hibajelzési kód, 0 ha ok

2.2. Függőségek

- graphics
- map
- controls
- state
- event_handler
- mytime
- debugmalloc
- SDL
- szabványos: time, stdlib

3. Eseménykezelés - event_handler.c

Legfőbb feladata a main-től megkapott esemény feldolgozása. Három fő eseménytípust kezel: adott billentyű lenyomása, kattintás, és folyamatos gépelés (text input). A gombok funkció itt vannak definiálva, a modul pedig a nevük alapján azonosítja őket. Itt válthat a program különböző módok között a felhasználó által generált esemény következményeként, valamint itt vizsgálja a program, hogy véget ért-e a játék.

3.1. Függvények

startNewGame - void startNewGame(State *state, Objects *objects)

- egyes játékok előtti inicializálás
- paraméterel:
  - alapállapotba állítandó state és objects
  - az objects->vertice->list mezőbe egy dinamikusan foglalt memóriaterületre mutató pointer kerül,ezt a hívónak kell felszabadítania

event_handle - void event_handle(SDL_Event ev, State *state, Objects *objects)

- a modul fő függvénye, kezeli a paraméterként kapott eseményt
- paraméter(ek):
  - az esemény, state, objects

3.2. Függőségek

- controls
- state
- map
- utilities
- mytime
- file_management
- SDL
- szabványos: string, stdbool

4. Grafika - graphics.c

Feladata a játék megjelenítése és az SDL inicializálása. Itt van meghatározva a gombok kinézete a nevük és a típusuk alapján. A módtól függően kirajzolja a popup ablakokat és a gombokat is. A térkép kirajzolásához a Voronoi diagram szabályai követi: a térkép minden pontjához megkeresi a vertexlista hozzá legközelebb eső elemét (illetve vizsgálja, hogy két középpontól közel egyenlő távolságra van-e - azaz két mező határán van), és ennek megfelelően színezi ki.

4.1. Típusdefiníciók és makrók

scWidth, scHeight (makró):

- a megnyitott ablak méretei

SDL_pointers: az SDL könyvtárhoz tartozó objectek pointerei, melyeket az SDL_init ad vissza

- renderer (*SDL_Renderer**)
- window (*SDL_Window**)
- fontSmall (*TTF_Font**): betöltött font, 22-s betűméret
- fontLarge (*TTF_Font**): betöltött font, 40-es betűméret

4.2. Függvények

SDL_init - SDL_pointers SDL_init()

- az SDL könyvtárral kapcsolatos objectek inicializálása
- visszatérési érték: a létrehozott objeckekre mutató pointerek struktúrája

drawScreen - void drawScreen(SDL_pointers sdl, const State *state, const Objects *objects)

- a modul fő függvénye, kirajzolja a teljes képernyőt a state és az objects állapota alapján
- paraméter(ek):
  - kirajzoláshoz szükséges sdl pointerek struktúrája, a state és a kirajzolandó objectek

4.3. Függőségek

- linked_list
- geometry
- map
- controls
- state
- file_management
- mytime
- SDL: SDL, SDL2_gfxPrimitives, SDL_ttf
- szabványos: string

5. State - state.h

A main, a grafika és az eseménykezelő modulok közötti kommunikációt biztosító State és Objects struktúrák definíciói vannak ebben a headerben.

5.1. Típusdefiníciók

State:

- mode (*Mode*): játék jelenlegi módja
- paused (*bool*): játék le van-e állítva
- ended (*bool*): véget ért-e a játék
- username (*char[30+1]*): jelenlegi játékos felhasználóneve
- usrnamebuffer (*char[30+1]*): az új játék módban beállított, de még el nem indított játék játékosának felhasználóneve
- timer (*Timer*): jelenlegi játékhoz tartozó időértékek
- currentColor (*int*): jelenleg kiválasztott szín indexe a palette.fields listában
- blankNum (*int*): semlegesen hagyott mezők száma
- diffSett (*DifficultySetting*): játék nehézségi szintjének beállításai

Objects:

- btns (*BtnList*): játék gombjainak listája
- vertice (*VertList*): mezők középpontjainak listája
- top10 (*ResList*): első tíz legjobb játékos listája a játék jelenlegi nehézségi szintjén
- userPlace (*int*): hányadik helyezést ért el az adott játékos
- palette (*Palette*): játék megjelenítésénél használt színpaletta

5.2. Függőségek

- controls
- file_management

6. Vezérlés - controls.h

Ebben a headerben vannak definiálva a különböző vezérléshez szükséges enumok és struktúrák.

6.1. Típusdefiníciók

Mode (enum)**:** a játék lehetséges módjai

- gameMode: nincs semmilyen felugró ablak
- endWindowMode: játék végén megjelenő ablak
- newGameMode: új játékot beállító ablak
- leaderboardMode: dicsőséglistát megjelenítő mód

BtnName (enum)**:** a gombok lehetséges nevei (számértékeket a program felhasználja!)

- paused: játékot leállító/elindító gomb neve
- color1, color2, color3, color4: színváltó gombok nevei
- getNewGame: új játék ablak megnyitására szolgáló gomb neve
- getLeaderboard: dicsőséglista megnyitására szolgáló gomb neve
- back: játékmódba visszatérő gomb neve, több példánya is van
- ok: új játék elindítására szolgáló gomb neve
- easyDiffBtn, mediumDiffBtn, hardDiffBtn, ironmanBtn: nehézséget állító gombok nevei

BtnType (enum)**:** gombok megjelenésének/funkcióinak típusa

- text: szöveges gomb
- color: egyszínű gomb
- icon: egyedi ikonnal rendelkező gomb
- diffRadio: nehézséget állító rádiógomb
- checkBox: jelölőnégyzet

Button: gomb struktúra

- name (*BtnName*), type (*BtnType*): ld. fent
- coord (*Point*): a gomb bal felső sarkának koordinátája
- width, height (*int*): a gomb méretei
- visibility (*Mode*): a gomb láthatósága, azaz melyik módban látható és kattintható a gomb

BtnList: gombokat tartalmazó lista

- list (*Button**): a lista első elemére mutató pointer
- len (*int*): a lista hossza

Timer: saját időzítő struktúra

- timePassed (*Time*): minden eddigi elindítás és leállítás között eltelt idő
- timeSincePaused (*Time*): a legutóbbi leállítás óta eltelt idő
- timeStarted (*int*): a legutóbbi leállítás időpillanata milisekundumban

Difficulty (enum)**:** a játék lehetséges nehézségi módjai (számértékeket a program felhasználja!)

- easyDiff, mediumDiff, hardDiff: a három nehézségi fokozat

DifficultySetting: a játék nehézségi módjának beállítása

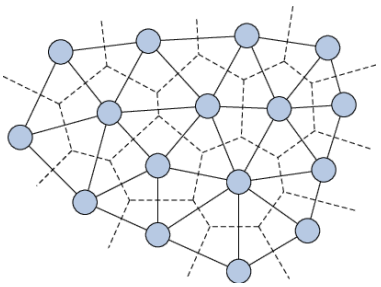
- difficulty (*Difficulty*): jelenlegi játék nehézsége
- selectedDiff (*Difficulty*): az új játék módban beállított, de még el nem indított játék nehézsége
- ironman (*bool*): a jelenlegi játék vasember módban van-e
- selectedIman (*bool*): az új játék módban beállított, de még el nem indított játék vasember módban van-e

6.2. Függőségek

- geometry
- mytime
- szabványos: time, stdbool

7. Térkép - map.c

Ez a modul kezel minden, a térkép mezőivel kapcsolatos dolgot: legenerálja a középpontokat, átszínezi a mezőket és ami a legfontosabb, ellenőrzi, hogy helyesen van kiszínezve-e a térkép.



Mivel a mezőket a Voronoi diagram szabályai szerint generálja a program, így ennek a duálisa, a Delaunay trianguláció (kisebb módosításokkal) megadja a mezők szomszédsági gráfját. A Delaunay trianguláció, azonban azokat a mezőket is szomszédosnak veszi, akik a képernyő határain kívül érintkeznek csak. Így a trianguláción még annyiban kell módosítani, hogy azon háromszögeknek, amelyeknek a körülírt körének középpontja a képernyőn kívül esik a leghosszabb oldalát el kell távolítani a szomszédsági gráfból.

Delaunay trianguláció generálásához használt referencia: link.

7.1. Makrók

mapWidth, mapHeight (makró):

- a térkép méretei

mapOffset (makró):

- a térkép bal felső sarkának pozíciója (x és y irányban is) a megnyitott ablak bal felső sarkától

7.2. Függvények

genVertice - void genVertice(VertList *vertice, int vertNum)

- létrehozza a vertexek tömbjét 4 fix sarokponttal és random genrált elemekkel
- paraméter(ek):
  - VertList pointer, melynek a list mezőjébe egy dinamikusan foglalt memóriaterületre mutató pointer íródik, ezt a hívónak kell felszabadítania
  - a generálandó pontok száma, minimum négyet kell átadni, mivel 4 fix sarokpont mindig van

delaunay - TriLinkedList delaunay(VertList vertice)

- létrehozza a delaunay triangulációt egy láncolt listába
- paraméter(ek):
  - vertexek listája, ami alapján a triangulációt generáljuk
- visszatérési érték: a listára mutató pointer struct - hívónak kell felszabadítania

finalEdges - EdgeLinkedList finalEdges(TriLinkedList triangles)

- elkészíti a a végső szomszédsági gráfot, tehát megnézi, hogy a mezők valójában csak a térképen kívül, vagy valóban érintkeznek-e
- paraméter(ek):
  - trianguláció, ami alapján elkészíti a végleges éleket
- visszatérési érték: a listára mutató pointer struct - hívónak kell felszabadítania

moveVertice - void moveVertice(VertList vertice)

- elmozgatja a pontokat a megadott sebességük alapján és random változtat az irányukon
- paraméter(ek):
  - az elmozgatandó vertexek listája

recolorField - int recolorField(Point click, VertList vertice, int col)

- átállítja annak a mezőnek a színét, ami az adott koordinátájú pontot tartalmazza
- paraméter(ek):
  - a koordináta, amelyhez tartozó mezőt át kell színezni
  - a mezőkhöz tartozó vertexek listája
  - a szín amire át akarjuk állítani a mezőt
- visszatérési érték: -1, 0 vagy 1
  - -1 ha a mezőt semleges színről egy nem semlegesre lett színezve
  - 0 ha a mezőt semlegesről semlegesre, vagy nem semlegesről nem semlegesre lett színezve
  - 1 ha a mezőt nem semlegesről semglegesre lett színezve

correctMap - bool correctMap(const VertList vertice)

- megnézi hogy a térkép kiszínezése helyes-e, azaz nincs két egymás melletti ugyanolyan színű mező
- paraméter(ek):
  - ellenőrizendő vertexek listája
- visszatérési érték: true ha a színezés helyes, false ha nem

7.3. Függőségek

- geometry
- linked_list
- utilities
- debugmalloc
- szabványos: math, stdbool, stdlib

8. Láncolt listák - linked_list.c

Ez a modul kezeli a térkép által használt láncolt listákat. A program kétféle ilyen listát használ, egyet a háromszögek, egyet az élek eltárolására. Ennek a két listának a felépítése közel azonos, csupán a függvények típusai másak.

8.1. Típusdefiníciók

TriChain: a háromszögekből álló láncolt lista egy eleme

- tri (*VertTri*): a tárolt háromszög
- next (*TriChain**): a lista következő elemére mutató pointer, NULL, ha ő az utolsó

TriLinkedList: a háromszögekből álló láncolt lista adatait tárolja el

- first (*TriChain**): az első elemre mutató pointer
- last (*TriChain**): az utolsó elemre mutató pointer
- len (*int*): a láncolt lista hossza

EdgeChain: az élekből álló láncolt lista egy eleme

- e (*VertEdge*): a tárolt háromszög
- next (*EdgeChain**): a lista következő elemére mutató pointer, NULL, ha ő az utolsó

EdgeLinkedList: az élekből álló láncolt lista adatait tárolja el

- first (*EdgeChain**): az első elemre mutató pointer
- last (*EdgeChain**): az utolsó elemre mutató pointer
- len (*int*): a láncolt lista hossza

8.2. Függvények

addToTriLinked - void addToTriLinked(TriLinkedList *list, VertTri new)

- hozzáadja az elemet a láncolt listához, ha szükséges, megváltoztatja a first és last pointereket
- paraméter(ek):
  - a módosítandó lista
  - az új elem adata

rmvfromTriLinked - void rmvfromTriLinked(TriLinkedList *list, TriChain *tormv)

- kivágja az adott elemet a listából, felszabadítva a memóriát, ha szükséges, megváltoztatja a first és last pointereket
- paraméter(ek):
  - a módosítandó lista
  - a törlendő elemre mutató pointer

delTriLinked - void delTriLinked(TriLinkedList *list)

- letörli a láncolt listát, felszabadítva a memóriát, a first és last pointereket NULL-ra állítja
- paraméter(ek):
  - a törlendő lista

addToELinked - void addToELinked(EdgeLinkedList *list, VertEdge new)

- hozzáadja az elemet a láncolt listához, ha szükséges, megváltoztatja a first és last pointereket
- paraméter(ek):
  - a módosítandó lista
  - az új elem adata

rmvfromELinked - void rmvfromELinked(EdgeLinkedList *list, EdgeChain *tormv)

- kivágja az adott elemet a listából, felszabadítva a memóriát, ha szükséges, megváltoztatja a first és last pointereket
- paraméter(ek):
  - a módosítandó lista
  - a törlendő elemre mutató pointer

delELinked - void delELinked(EdgeLinkedList *list)

- letörli a láncolt listát, felszabadítva a memóriát, a first és last pointereket NULL-ra állítja
- paraméter(ek):
  - a törlendő lista

8.3. Függőségek

- geometry
- szabványos: stdlib

9. Fájlkezelés - file_management.c

A fájlkezelő modul segítségével tárolja el a program a dicsőséglistákat, így a játék bezárása után is megmaradnak. A játékban összesen hat féle dicsőséglista van - mindhárom nehézségi szinthez egy vasember módban és egy anélkül. Ezeknek a neve "*leaderboard-[easy/mdum/hard](_i).txt*".

9.1. Típusdefiníciók

PlayerResult: egy játékos által elért eredményt tároló struktúra

- name (*char[30+1]*): a játékos felhasználóneve
- t (*Time*): a játékos által elért eredmény

ResList: játékosok eredményeit eltároló lista

- len (*int*): a lista hossza
- results (*PlayerResult**): a lista első elemére mutató pointer

9.2. Függvények

getTop10 - void getTop10(ResList *list, DifficultySetting diff)

- kiolvasssa a nehézségnek megfelelő fájlból az első tíz legjobb játékos eredményeit
- paraméter(ek):
  - eredmények listája, melynek list mezőjébe egy dinamikusan foglalt memóriaterületre mutató pointer kerül, ezt a hívónak kell felszabadítania
  - a jelenlegi nehézségi szint, ami alapján ki tudja választani a megfelelő fájlt

addToLeaderBoard - int addToLeaderBoard(PlayerResult newres, DifficultySetting diff)

- hozzáad egy új eredményt a megfelelő fájlban lévő dicsőséglistához
- paraméter(ek):
  - az új eredmény, melyet a megfelelő helyre ír be a megfelelő fájlba
  - a jelenlegi nehézségi szint, ami alapján ki tudja választani a megfelelő fájlt
- visszatérési érték: a paraméterként megkapott eredmény helyezése az adott nehézségi kategóriában

9.3. Függőségek

- mytime
- controls
- debugmalloc
- szabványos: stdio

10. Időkezelés - mytime.c

Ebben a modulban található a program egyedi időstruktúrája és az azzal kapcsolatos függvények.

10.1. Típusdefiníciók

Time: a program egyedi időstruktúrája

- min, sec, csec (*int*): percek, másodpercek, századmásodpercek száma

10.2. Függvények

timeAdd - Time timeAdd (Time t1, Time t2)

- összead két időértéket
- paraméter(ek):
  - összeadandó idő struktúrák
- visszatérési érték: időstruktúrák összege

timeConvert - Time timeConvert(int t1)

- milliszekundumot átkonvertálja az egyéni időstruktúrára
- paraméter(ek):
  - konvertálandó idő miliszekundumban
- visszatérési érték: konvertált időstruktúra

compTime - int compTime(Time t1, Time t2)

- összehasonlít két időértéket
- paraméter(ek):
  - összahasonlítandó idő struktúrák
- visszatérési érték: -1, 0, 1
  - -1 ha t1 < t2
  - 0 ha egyenlőek
  - 1 ha t1 > t2

10.3. Függőségek

- a modulnak nincsenek függőségei

11. Geometria - geometry.c

Ebben a modulban vannak definiálva a geometriai struktúrák mint a pont, háromszög vagy él. Itt vannak még az ezen struktúrákkal kapcsolatos, és egyéb geometriai függvények.

11.1. Típusdefiníciók

Point: pont struktúra

- x, y (*double*): a pont koordinátái

Vertex: egy mező középpontját adó vertex struktúra

- coord (*Point*): a vertex koordinátái
- col (*int*): a mező színének sorszáma a paletta fields listájában
- speed (*double*): a vertex mozgásának sebessége frame-enként
- dir (*double*): a vertex mozgásának iránya radiánban

VertList: vertexek listája

- list (*Vertex**): a lista első elemére mutató pointer
- len (*int*): a lista hossza

Palette: grafikus megjelenésben látható színek

- bckgr (*SDL_Color*): a háttér színe
- btn (*SDL_Color*): a gombok háttérszíne
- dark (*SDL_Color*): általános felhasználású sötét szín
- grey (*SDL_Color*): általános felhasználású szürke szín
- pauseArrow (*SDL_Color*): a leállító gomb nyilának színe
- fields (*SDL_Color[5]*): a semleges szín, illetve a négy alapszín, amivel a térképet ki kell színezni

VertTri: egy Vertex pointerekből háromszög struktúrája

- a, b, c (*Vertex**): a háromszög három csúcs-vertexére mutató pointerek, a,b,c x majd y koordinátájuk alapján csökkenő sorrendbe vannak rendezve
- center (*Point*): a háromszög körülírt körének középpontjának koordinátája

VertEdge: egy Vertex pointerekből háromszög struktúrája

- a, b, c (*Vertex**): az él két csúcs-vertexére mutató pointerek, a,b x majd y koordinátája alapján csökkenő sorrendbe vannak rendezve
- invalid (*bool*): az él két olyan pontot köt-e össze, amelyek valójában nem is szomszédosak

11.2. Függvények

gt - bool gt(Point a, Point b)

- két pont komparátor: először x majd y alapján vizsgálja
- paraméter(ek):
  - összehasonlítandó pontok
- visszatérési érték: true ha a>b, különben false

eqPoint - bool eqPoint(Point p1, Point p2)

- pontok ekvivalenciáját vizsgálja (x és y koordinátájuk is megegyezik-e)
- paraméter(ek):
  - összehasonlítandó pontok
- visszatérési érték: a pontok egyenlőek-e

eqEdge - bool eqEdge(VertEdge e1, VertEdge e2)

- élek ekvivalenciáját vizsgálja (végpontjaik megegyeznek-e)
- paraméter(ek):
  - összehasonlítandó élek
- visszatérési érték: az élek egyenlőek-e

eqTriangle - bool eqTriangle(VertTri t1, VertTri t2)

- háromszögek ekvivalenciáját vizsgálja (csúcsaik megegyeznek-e)
- paraméter(ek):
  - összehasonlítandó háromszögek
- visszatérési érték: a háromszögek egyenlőek-e

newTri - VertTri newTri(Vertex *a, Vertex *b, Vertex *c)

- VertTri konstruktor három Vertex pointerből: x majd y koordinátájuk alapján sorba rendezi őket
- paraméter(ek):
  - háromszöget alkotó vertexekre mutató pointerek
- visszatérési érték: az új háromszög

newEdge - VertEdge newEdge(Vertex *a, Vertex *b)

- VertEdge konstruktor két Vertex pointerből: x majd y koordinátájuk alapján sorba rendezi őket, alapértelmezetten invalid mező hamis
- paraméter(ek):
  - élet alkotó vertexekre mutató pointerek
- visszatérési érték: az új él

point_in_tri - int point_in_tri(Point a, VertTri tri)

- adott vertex a háromszög potnjai között van-e és ha igen, melyik helyen
- paraméter(ek):
  - a vizsgálandó pont és háromszög
- visszatérési érték: 0, 1, 2, 3
  - 0 ha a pont nem eleme a háromszögnek
  - 1 ha a pont a háromszög a mezőjével egyenlő
  - 2 ha a pont a háromszög b mezőjével egyenlő
  - 3 ha a pont a háromszög c mezőjével egyenlő

point_in_circumscribed - bool point_in_circumscribed(Point p, VertTri tri)

- adott pont benne van-e egy háromszög körülírt körében
- paraméter(ek):
  - a vizsgálandó pont és háromszög
- visszatérési érték: true ha benne van a körülírt körében, különben false

dist2 - double dist2(Point a, Point b)

- két pont közötti távolság négyzete
- paraméter(ek):
  - a vizsgálandó pontok
- visszatérési érték: a két pont közötti távolság négyzete

dotProduct - int dotProduct(Point a, Point b)

- két síkvektor skalárszorzata
- paraméter(ek):
  - a vizsgálandó kétdimenziós vektorok
- visszatérési érték: a két vektor skalárszorzata

11.3. Függőségek

- SDL
- szabványos: stdbool

12. Segédfüggvények - utilities.c

Általános felhasználású függvények, melyekre több modulnak is szüksége van.

12.1. Függvények

randint - int randint(int a, int b)

- visszaad egy egész értéket [a,b) intervallumból
- paraméter(ek):
  - az intervallum két végpontja
- visszatérési érték: random egész szám

randDouble - double randDouble(double a, double b, int precision)

- visszaad egy valós értéket [a,b) intervallumból adott pontossággal
- paraméter(ek):
  - az intervallum két végpontja
  - tört pontossága tizedesjegyben
- visszatérési érték: random valós szám

min2 - int min2(int a, int b)

- két egész közül visszaadja a kisebbet

max2 - int max2(int a, int b)

- két egész közül visszaadja a nagyobbat

safeCat - void safeCat(char *str1, const char *str2, int maxlen)

- ha az összefűzött sztring hossza nem haladja meg a paraméterként kapott maxlent, hozzáfűzi az elsőhöz a másodikat, különben nem csinál semmit

delOneChar - void delOneChar(char *str)

- a többites karakterek törlésére szolgál, mindig pontosan a legutolsó karaktert törli egy sztringből

12.2. Függőségek

- szabványos: stdlib, math, string, stdbool