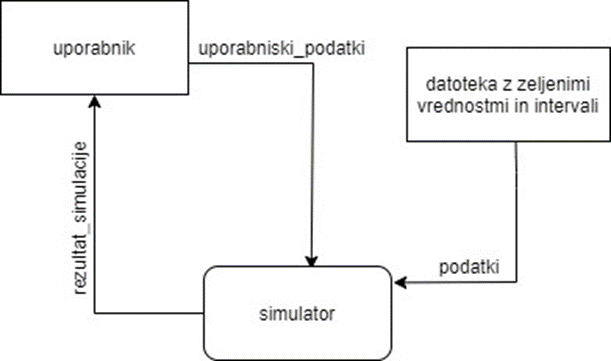
# Načrtovalska dokumentacija

## Identifikacija dokumenta, povezava z drugimi dokumenti

|  |  |
| --- | --- |
| **Dokument Načrtovalska dokumentacija** | |
| verzija | 1.0 |
| datum | 4. 2. 2021 |
| lokacija | datoteka Projekt\_Simulator\_Vinsek\_Volker.pdf |
| **Odgovorna oseba** | |
|  | Primož Volker |
| **Reference na drugo dokumentacijo** | |
|  | Interni standard CVVS 2-2000  Naročnikove zahteve V 2.0  Plan projekta V 1.0  Sistemske specifikacije V 1.0  Uporabniški priročnik V 1.0  Testni primeri V 1.0 |

## Povzetek iz specifikacij

### Kontekstni nivo



### Datoteke, ki jih uporablja uporabnik

Uporabnik za delovanje programa potrebuje tekstovno datoteko, v kateri so zapisani podatki v določenem vrstnem redu. Struktura datoteke mora ostati zmeraj enaka, uporabnik lahko spreminja le vrednosti. Teh datotek je lahko več. Primer imena datoteke: *podatki.txt*.

Za zagon programa uporabnik potrebuje izvršilno datoteko *Simulator.exe*. V tej datoteki se nahaja koda programa, spreminjanje te datoteke ni dovoljeno.

### Zagon programa

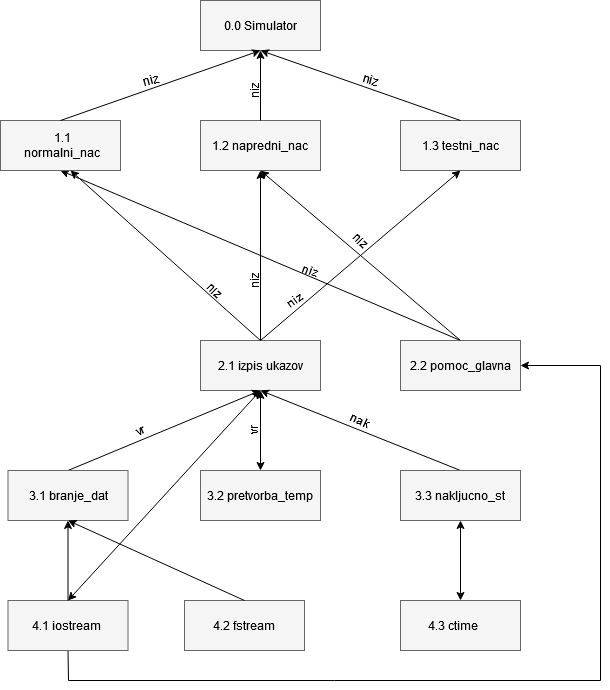
Zagon programa poteka preko ukazne vrstice. Nahajati se moramo v mapi projekta, nato vtipkamo *Simulator.exe* za normalni način. Doda se lahko tudi parameter *-t* za testni način ali *-c* za napredni način. Zagon normalnega načina je mogoč tudi z dvoklikom na ikono ali bližnjico.

### Datoteke, ki jih potrebuje vzdrževalec

Vzdrževalec potrebuje naslednje datoteke:

* celotna projektna dokumentacija (Projekt\_Simulator\_Vinsek\_Volker.pdf)
* mapo projekta za zagon v Visual Studio 2017 (izvorna koda se nahaja v Simulator.cpp)
* primer tekstovne datoteke *podatki.txt*

## Strukturni diagram ter seznam modulov in podatkovnih tokov



Vsi moduli se nahajajo v datoteki Simulator.cpp. Pod pojmom moduli so mišljene posamezne funkcije, knjižnice ali pa deli kode v programu.

### Moduli

|  |  |  |  |
| --- | --- | --- | --- |
| **id** | **Ime modula** | **Datoteka** | **opomba** |
| 0.0 | Simulator | Simulator.cpp | celoten program |
| 1.1 | normalni\_nac | Simulator.cpp | funkcija |
| 1.2 | napredni\_nac | Simulator.cpp | funkcija |
| 1.3 | testni\_nac | Simulator.cpp | funkcija |
| 2.1 | izpis ukazov | Simulator.cpp | del kode |
| 2.2 | pomoc\_glavna | Simulator.cpp | funkcija |
| 3.1 | branje\_dat | Simulator.cpp | funkcija |
| 3.2 | pretvorba\_temp | Simulator.cpp | del kode |
| 3.3 | nakljucno\_st | Simulator.cpp | del kode |
| 4.1 | iostream | Simulator.cpp | knjižnica |
| 4.2 | fstream | Simulator.cpp | knjižnica |
| 4.3 | ctime | Simulator.cpp | knjižnica |

### Podatkovni tokovi

|  |  |  |
| --- | --- | --- |
| Oznaka | Ime podatkovnega toka | vrsta (vhod/izhod, kontrola,status) |
| niz | znakovni niz (izpis na zaslonu) | izhod |
| vr | številska vrednost | vhod/izhod |
| nak | naključno število z intervala | izhod |

### Modul normalni\_nac

Je funkcija, katera zajema delovanje normalnega načina delovanja Simulatorja. Izvede se takrat, kadar ni podanega nobenega zagonskega parametra. Vključuje modul za izpis podatkov ter glavne pomoči, navigacija poteka preko menija. Potrebne podatke pridobi iz datoteke. Predstavlja glavno funkcionalnost programa.

### Modul napredni\_nac

Kopija funkcije normalni\_nac z dodanimi dodatnimi funkcionalnostmi. Vsebuje več izpisov in izračunov.

### Modul testni\_nac

Funkcija, namenjena predvsem testiranju delovanja programa. Omogoča vnos poljubnih podatkov, nato pa vrne pravilen izpis. Zaradi razlike v delovanju od prejšnjih dveh modulov, ima ta funkcija implementiran svoj izpis pomoči. Poleg glavnega menija vsebuje tudi podmeni.

### Modul izpis ukazov

Ne gre za posamezno funkcijo, temveč za dele kode znotraj prej navedenih funkcij. Te vrstice so namenjene predvsem izpisu na zaslon ob upoštevanju podanih vrednosti. V okviru tega modula so zajeti vsi izračuni (pretvorba temperature, izbira naključnih vrednosti) in branje podatkov iz datoteke.

Na koncu simulacije se izpiše statistika, njene vrednosti so izpisane z eno decimalko ali pa brez nje (če gre za celo število).

### Modul pomoc\_glavna

Funkcija, katera izpiše pomoč za glavni meni. Poklicana je iz funkcij normalni\_nac in napredni\_nac (ob izbiri \* na glavnem meniju).

### Modul branje\_dat

Funkcija branje\_dat je namenjena pridobivanju podatkov iz datoteke. Vsebuje tudi izpis pomoči in vnos imena datoteke, katera se nato prebere. Ta modul je potreben v vseh režimih delovanja.

### Modul pretvorba\_temp

Znotraj vseh funkcij načinov delovanja je del kode, ki pretvarja različne temperaturne enote v stopinje Celzija. Gre za stavka if in if else, katera preverjata, če je željena temperatura večja od 200 oz. večja od 50. Če je vrednost večja od 200, smatramo, da gre za kelvine, zato tej vrednosti odštejemo 273, da dobimo stopinje Celzija. Če pa je vrednost večja od 50, do vključno 200, vrednosti odštejemo 32 in delimo z 1,8 (pretvorba iz fahrenheitov v stopinje Celzija).

### Modul nakljucno\_st

Modul nakljucno\_st generira celo naključno število znotraj podanega intervala. Po branju iz datoteke se določijo minimalne ter maksimalne dovoljene vrednosti, nato pa se izbere naključna vrednost v tem razponu. Te vrednosti so shranjene v poljih, katerih dolžina je enaka številu simulacij.

### Modul iostream

Osnovna knjižnica za izpis in vnos v konzolo. Uporabljena je tudi pri branju iz datoteke (v kombinaciji z fstream).

### Modul fstream

Knjižnica za delo z datotekami. Uporabljena je v modulu branje\_dat, kjer omogoča odpiranje, branje ter zapiranje datoteke.

### Modul ctime

Knjižnica za delo z datumi in časi. V projektu je uporabljena v kombinaciji z razredom srand, da je generacija naključnih števil bolj učinkovita. V nasprotnem primeru lahko pride do ponavljanja naključnih števil.

## Najpomembnejši parametri in opisi podatkovnih struktur

### Struktura vhodne datoteke

Namen te datoteke je pridobitev željenih ambientalnih lastnosti in intervalov. Je tekstovna datoteka s šestimi vrsticami. Spreminjajo se lahko le vrednosti na koncu vrstic, ter vrednosti v intervalih. Paziti je potrebno na to, da so željene lastnosti podane znotraj intervalov. Primer datoteke:

TEMPERATURA: 22

VLAZNOST: 45

OSVETLJENOST: 500

INTERVAL TEMPERATURE: [10,40]

STOPNJA VLAZNOSTI: [30,60]

INTERVAL OSVETLJENOSTI: [10,10000]

Enote so določene v sistemskih specifikacijah.

### Parametri pri zagonu programa

Program prepozna dva vhodna parametra. Parameter *-t* zažene program v testnem režimu, parameter *-c* pa napredni režim načina delovanja. Ob zagonu programa lahko podamo samo en parameter, nikoli oba, saj se program tako ne bo zagnal. Prav tako pa se ne zažene (samo izpiše opozorilo), če namesto teh dveh parametrov vnesemo karkoli drugega. Ob nepravilnem podanem parametru program izpiše opozorilo o nepravilnem vnosu, nato pa se zaustavi.

Po zagonu se v prvi vrstici za imenom ter verziji programa izpiše tudi način delovanja. To velja za testni in napredni način, pri navadnem tega izpisa ni.

## Natančna identifikacija uporabljenih orodij in knjižnic

* Za pisanje in prevajanje izvorne kode smo uporabili Microsoft Visual Studio 2017. Kodo bo najverjetneje mogoče zagnati tudi z najnovejšo različico tega orodja in ostalimi C++ prevajalniki.
* Za izdelavo dokumentacije je bilo uporabljeno orodje Microsoft Word iz paketa Microsoft Office, s tem orodjem smo tudi pretvorili dokumentacijo v format PDF.
* Za izdelavo grafov smo uporabili spletno orodje draw.io, dosegljivo na spletnem naslovu <https://app.diagrams.net/>
* Za izdelavo vhodne tekstovne datoteke smo uporabili beležnico sistema Windows 10, za to datoteko lahko uporabimo tudi druge urejevalnike besedila.

## Postopek potreben za ustvarjanje izvršilne kode

V Visual Studiu odpremo projekt Simulator.sln, nato pa v zgornji vrstici izberemo meni Build. Po odprtju menija izberemo Build Solution. Počakamo nekaj sekund, nato pa si v spodnjem zavihku Output lahko ogledamo »BuildLog«. Po uspešni izgradnji izvršilne kode, program Simulator.exe najdemo v mapi projekta v podmapi Release ali pa Debug, odvisno od nastavljene konfiguracije.

Primer izpisa v oknu Output:

1>------ Build started: Project: Simulator, Configuration: Release Win32 ------

1>Simulator.cpp

1>Generating code

1>All 209 functions were compiled because no usable IPDB/IOBJ from previous compilation was found.

1>Finished generating code

1>Simulator.vcxproj -> C:\Users\primozvolker\Documents\Visual Studio 2017\Projects\Simulator\Release\Simulator.exe

1>Done building project "Simulator.vcxproj".

========== Build: 1 succeeded, 0 failed, 0 skipped ==========

## Pojmovnik