Dokumentacja programu zarządzającego bazą studentów, prowadzących i przedmiotów

Spis treści

1 Wstęp

1.1 Opis Problemu

Celem projektu jest jest napisanie programu, który będzie zarządzał bazą danych zawierającą studentów, przedmioty i prowadzących. Funkcje implementujące działanie programu zostaną oddzielone od interfejsu użytkownika, w celu umożliwienia udostępnienia ich jako biblioteka. Program będzie potrafił obsługiwać bazy danych o teoretycznie dowolnej wielkości dzięki wczytywaniu plików we fragmentach.

1.2 Narzędzia

Program został napisany w języku `C` zgodnym ze standardem **ISO C90**. Wykorzystywano kompilator **gcc**, debugger **gdb**. Program jest konfigurowany i budowany przy użyciu **GNU Autotools** i **GNU Make**. Dokumentacja jest formatowana przy użyciu *LaTeX*.

2 Specyfikacja Funkcjonalna

2.1 Funkcjonalność

2.1.1 Funkcjonalność podstawowa

- dodawanie, usuwanie, edycja, wyliczanie oraz wyliczanie powiązań studentów
- dodawanie, usuwanie, edycja, wyliczanie oraz wyliczanie powiązań prowadzących
- dodawanie, usuwanie, edycja, wyliczanie oraz wyliczanie powiązań przedmiotów

2.2 Komunikacja z użytkownikiem

Komunikacja z użytkownikiem odbywa się w trybie wsadowym przez tekstowy interfejs użytkownika.

2.2.1 Argumenty lini poleceń

```
NAZWA
      spp - program zarządzający bazą danych

SKŁADNIA
      spp [ opcje ] ...

OPIS
     spp jest programem przeznaczonym do zarządzania bazą danych
     studentów, lektorów i przedmiotów.

OPCJE
 TRYBY:
      -b  --build <baza>    - tryb budowania bazy danych
      -a  --add             - tryb dodawania
      -r  --rm              - tryb usuwania
      -l  --show-links      - tryb wyliczania powiązań
      -c  --cat             - tryb wyliczania
      -e  --edit            - tryb edytowania pytań
  FLAGI:
      -h --help             - wyświetl krótką pomoc
      -v --verbose          - wyświetlaj więcej informacji
  OPCJE
      -t, --type <student|lecturer|topic> - typ rekordu
      -d, --data <baza>     - baza danych
      -l, --log <file>      - plik logu
      --config <file>       - plik konfiguracyjny
  OPCJE POL REKORDU
      -i, --id <int>        - id rekordu
      -N, --name <nazwa>    - imię lub nazwa
  OPCJE POL REKORDU typu student
      -I, --index <numer indeksu>
      -T, --topic <topic1 id,topic2 id,...>
  OPCJE POL REKORDU typu lecturer (wykładowca)
      -D, --degree <stopień naukowy>
      -R, --room <pokój wykładowcy>
      -Y, --topic-type <typ przedmiotu>
      -T, --topic <topic1 id,topic2 id,...>
  OPCJE POL REKORDU typu topic (przedmiot)
      -A, --date <termin zajęć>
```

```
-H, --hours <liczba godzin w semestrze>
```

2.2.2 Plik konfiguracyjny

Argumenty lini poleceń mogą zostać zapisane w pliku

`~/.spprc`

(lub innym podanym przy opcji –config). Będą one wczytywane przy każdym wywołaniu programu bez konieczności podawania ich z lini komend. Przykładowy plik konfiguracyjny:

```
#------------------------------
--log log.txt                   # komunikaty diagnostyczne do log.txt
--data ~/db                     # lokalizacja bazy danych
#---------------EOF------------
```

2.2.3 Przykłady użycia programu

```
# dodanie przedmiotu
spp --type subject --add --name "Programowanie" \
--subject-type "egz" --hours "30" --date "Piątek 18:15"

spp --type subject --add --name "Wychowanie Fizyczne" \
--subject-type "zal" --hours "30" --date "Poniedziałek 8:15"

# dodanie studenta
spp --type student --add --name "Janek Kowalski" \
--index 123456 --subject "1,2"

# dodanie prowadzącego
spp --type lecturer --add --name "Janusz Kowalski" \
--degree "dr. hab. prof. nzw." --room "GE510" --subject "1"

# wyliczaj studentów
spp --type student --cat

# edytuj studenta
spp --type student --id 5 --edit --subject "2,3,4"

# drukuj przedmioty asocjowane ze studentem
spp --type student --id 5 --show-links

# usuń studenta
spp --type student --id 5 --rm
```

2.3 Format Danych

2.3.1 Baza danych

Dane będa przechowywane w prostej bazie danych. Przykładowy rekordy:

- student

```
id|name|indeks|subjects
-----------------------------------------
5|Janek Kowalski|123456|Programowanie, Wychowanie fizyczne
```

- prowadzący

```
id|name|degree|room|subjects
-----------------------------------------
5|Janusz Kowalski|dr. hab. prof. nzw.|GE510| Programowanie, Bazy danych
```

- przedmiot

```
id|name|subject type|hours|date
-----------------------------------------
5|Programowanie|egz|30|Piątek 18:15
```

2.3.2 Dane wyjściowe

- program standardowo wyświetla jeden rekord na linię
- wszystkie komunikaty poprzedzone są znakiem "!"i wypisywane na stderr lub do loga
  * komunikaty ostrzegawcze "! W "
  * komunikaty fatalne "! E "
  * możliwe jest zwiększenie ilości wypisywanych komunikatów diagnostycznych przez dodanie flagi –verbatim

2.4 Działanie w przypadku nieprawidłowych danych

Program stara się testować argumenty. W przypadku podania przez użytkownika nieprawidłowych danych program przerywa działanie oraz wypisuje kod błędu.

3 Specyfikacja Implementacyjna

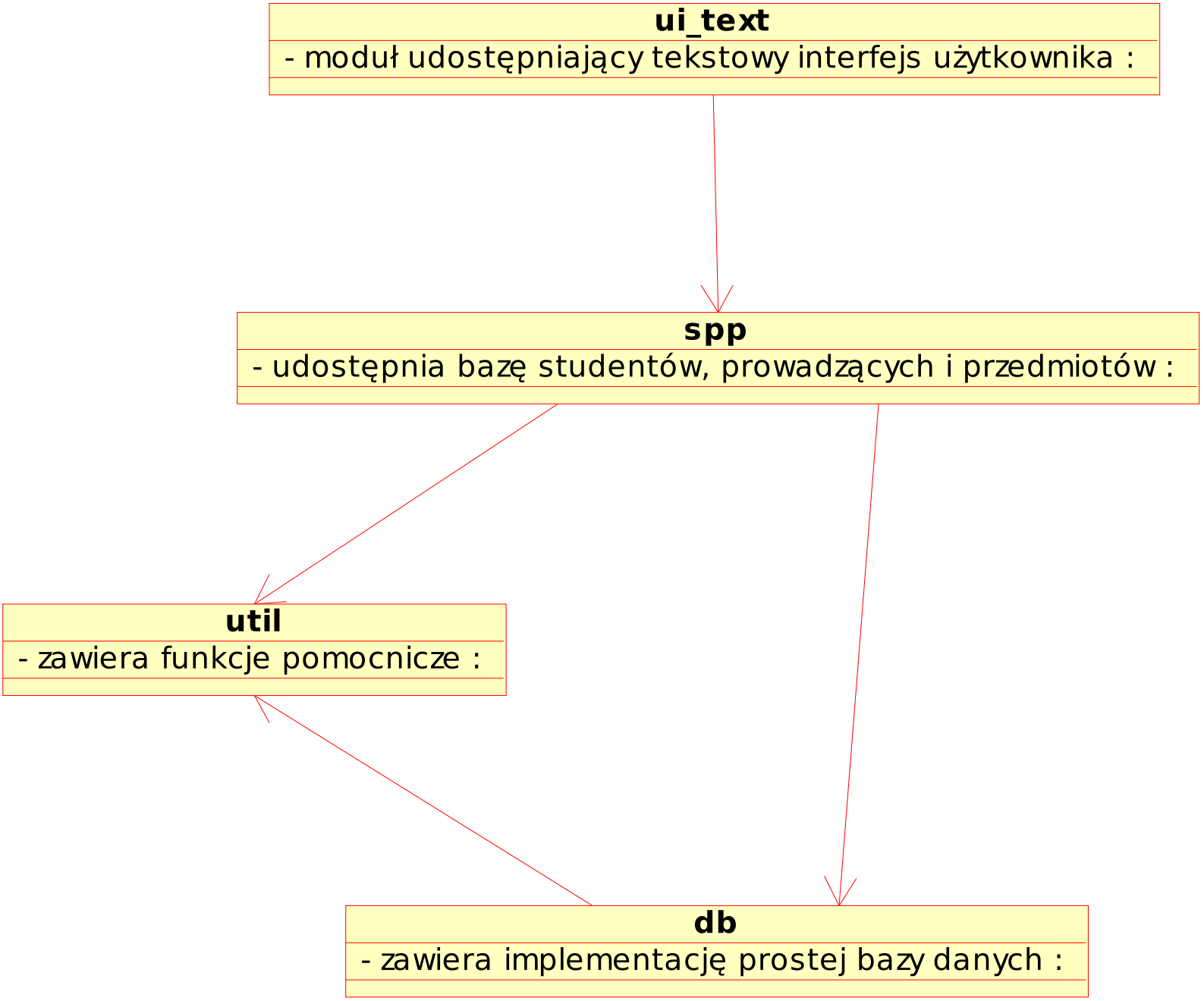
3.1 Opis Ogólny

Krótki opis modułów.

Moduł	**Pliki**	**Zależy**	**Opis**
Baza danych			
db	db_io.[ch] dbtypes.h	util	prosta baza danych
spp	spp.c spp.h	db util	eksportuje funkcjonalność bazy danych
util	util.c util.h	-	funkcje pomocnicze
define	define.h	-	lokalne stałe
Interfejs Użytkownika			
ui_text	ui_text.c ui_text.h	spp	tekstowy interfejs użytkownika

Program korzysta z następujących systemowych plików nagłówkowych:

- stdlib.h
- stdio.h
- string.h
- time.h
- stdarg.h
- getopt.h
- ctype.h


ui_text
- moduł udostępniający tekstowy interfejs użytkownika :
spp
- udostępnia bazę studentów, prowadzących i przedmiotów :
util
- zawiera funkcje pomocnicze :
db
- zawiera implementację prostej bazy danych :

3.2 Baza danych

Baza danych jest przechowywana w trzech plikach tekstowych:

- *.stud - rekordy studentów
- *.lecturer - rekordy prowadzących
- *.topic - rekordy przedmiotów

Maksymalna liczba rekordów jednorazowo wczytywanych przez bazę danych jest określona poprzez MAX_RECORDS (domyślnie 10). Po wykonaniu operacji następuje wczytanie kolejnych rekordów z bazy danych. Dzięki takiej implementacji, wielkość wymaganej pamięci operacyjnej nie zwiększa się wraz z wiekością bazy danych. Rekordy są indeksowane od 1 (dla każdego pliku oddzielnie). Rekordy studentów i prowadzących są powiązane z rekordami przedmiotów poprzez indeksy odseparowane przecinkami, które są wczytywane do pola **topic**. Usunięcie przedmiotu powoduje również usunięcie powiązań znajdujących się w rekordach studentów i prowadzących. W pliku db_types.h znajdują się definicje struktur danych:

```
typedef struct _st_rekord_lecturer
{
    int id;
    char name[MAX_NAME];
    char degree[MAX_DEGREE];
    char room[MAX_ROOM];
    char topic[MAX_TOPIC];
} st_rekord_lecturer;

typedef struct _st_rekord_topic
{
    int id;
    char name[MAX_NAME];
    char topic_type[MAX_TOPIC_TYPE];
    int hours;
    char date[MAX_DATE];
} st_rekord_topic;

typedef struct _st_rekord_student
{
```

```
    int id;
    char name[MAX_NAME];
    int index;
    char topic[MAX_TOPIC];
} st_rekord_student;
```

Baza danych udostępnia swoją funkcjonalność przez moduł spp.

3.2.1 Moduł spp

Moduł **spp** składa się z 2 plików: ***spp.c spp.h***
spp daje dostęp do bazy danych przez prosty interfejs
spp jest zależny od modułów *db , util.*
Potrzebuje definicje stałych zawarte w pliku *define.h.*

spp składa się z następujących elementów:

```
/* Struktura sppArgs zawiera wszystkie~
 * parametry potrzebne modułowi spp */
struct sppArgs
{
    int mode; /* tryb działania */

    char type; /* typ rekordu na którym operujemy */
    char *data; /* nazwa bazy danych */
    char *log; /* nazwa pliku logu */
    char *out; /* plik wyjścia standardowego */
    char *build; /* nazwa tworzonej bazy danych */

    int id; /* pole rekordu: numer identyfikacyjny rekordu */
    char *name; /* pole rekordu: nazwa */
    char *degree; /* pole rekordu: stopień naukowy */
    char *room; /* pole rekordu: pokój */
    char *date; /* pole rekordu: termin zajęć */
    char *topic; /* pole rekordu: przedmioty */
    char *topic_type; /* pole rekordu: typ przedmiotu */
    int hours; /* pole rekordu: liczba godzin */
    int index; /* pole rekordu: numer indeksu */
};

/* Funkcja przygotowująca moduł spp do działania,
```

```
 * jako argument pobiera struckturę sppArgs.
 * Kody błędów powyżyej. */
int
spp_setup (struct sppArgs *);

/* Funkcja kończąca działanie modułu spp.
 * Zamyka pliki i uwalnia pamięć
 * Kody błędów powyżyej. */
void
spp_finish ();

/* Wywołanie funkcji spp_run powoduje wykonanie
 * działań zaprogramowanych w strukturze sppArgs
 * Kody błędów powyżyej. */
int spp_run ();
```

3.2.2 Moduł db

Moduł **db** składa się z 3 plików: ***db_io.c db_io.h dbtypes.h***
db zawiera implementacje prostej bazy danych.
db jest zależny od modułu *util*.
Potrzebuje definicje stałych zawarte w pliku *define.h*.

Funkcjonalność udostępniana przez moduł db jest zadeklarowana w pliku db_io.h

```
/* Funkcje wczytujące z plików (parametr FILE * in) zbiory rekordów.
 * Wynik zapisywany jest w tablicach struktur.
 *  Parametr max to maksymalna ilosc rekordów wczytana jednorazowo.
 * Funkcje zwracają liczbę wczytanych rekordów.
 */
int db_read_records_students (FILE * in, st_rekord_student records[],

int db_read_records_lecturers(FILE * in, st_rekord_lecturer records[],

int db_read_records_topics (FILE * in, st_rekord_topic records[], int

/* Funkcje zapisujące w plikach (FILE * out) rekordy przechowywane
 * w tablicach struktur podawanych jako parametr funkcji.
 * int n - liczba rekordow w tabeli
```

```
 */

void db_write_records_students (FILE * out, st_rekord_student records[

void db_write_records_lecturers(FILE * out, st_rekord_lecturer records

void db_write_records_topics (FILE * out, st_rekord_topic records[], i


/* Generuje niewykorzystany ID dla rekordu */
int dbGetNewID (FILE * in);

/* Usuwa rekord o podanym ID, wynik zapisuje w out */
int dbRemoveID(FILE * in, FILE * out, int rmid);

int db_remove_topic_id_from_stud(FILE * in, FILE * out, int rmid);

int db_remove_topic_id_from_lecturer(FILE * in, FILE * out, int rmid);
```

3.2.3 Moduł util

Moduł **util** składa się z 2 plików: ***util.c util.h***
util zawiera funkcje pomocnicze **util** nie zależy od innych modułów.
Potrzebuje definicje stałych zawarte w pliku *define.h*.

util składa się z następujących elementów:

```
/* Funkcja kopiująca pliki.
 * Zwraca 0 lub wartość != 0 dla niepowodzenia */
int
util_copy_file(FILE *in, FILE *out);

/* Funkcja usuwająca znak nowej lini.
 * Zwraca wskaźnik na wynikowy łańcuch znaków */
char *
util_strip_nl(char *s);
```

3.3 Interfejs użytkownika

Program działa w trybie tekstowym, a komunikacja z użytkownikiem odbywa się w trybie wsadowym. Nakładka tekstowa wykorzystuje bibliotekę getopt do czytania argumentów lini poleceń.

3.3.1 Moduł ui_text

Moduł **ui_text** składa się z 2 plików: ***ui_text.c*** ***ui_text.h***
ui_text zawiera funkcje nakładki tekstowej
ui_text jest zależny od modułu *spp*.
Potrzebuje definicje stałych zawarte w pliku *define.h*.

ui_text składa się z następujących elementów:

```
/* Funkcja kontrolująca działanie nakładki tekstowej. */
int
ui_run (int argc, char **argv);

/* Wczytaj argumenty lini poleceń */
int
ui_setup_arguments (int argc, char **argv);

/* Wypisz wszystkie dostępne opcje */
void
ui_show_options (FILE *);

/* Pokaż opis programu */
void
ui_show_description (FILE *);

/* Pokaż krótka pomoc */
void
ui_show_help (FILE *);
```

3.3.2 Plik define.h

W pliku **define.h** znajdują się definicje stałych potrzebnych w programie oraz definicja typu *bool*.

3.4 Kody błędów

Tablica kodów błędów:

-1 - złe/niewystarczające argumenty

0 - nie wykryto błędów

1 - nie otrzymano struktury sppArgs

2 - wymagana nazwa pliku jest nieznana

3 - nie udało się otworzyć pliku bazy danych

4 - nie udało się otworzyć pliku logu

5 - nie udało się otworzyć pliku wyjścia

6 - nie udało się otworzyć pliku do zakodowania

7 - nie udało się otworzyć pliku do odkodowania

10 - błąd podczas czytania pliku

11 - błąd podczas zapisywania do pliku

111 - zbyt długie dane wejściowe

4 Testy

4.1 Kompilacja

System operacyjny: Linux 2.6.24-21 Kompilator: gcc 4.2.4

4.2 Działanie programu

Działanie programu testowano manualnie. Przykładowe testy (plik tests.sh):

```
echo utworzenie bazy danych
spp --build /tmp/db
echo utworzenie pliku konfiguracyjnego ustawiającego domyślną bazę danych
echo '--data /tmp/db' > ~/.spprc

echo utworzenie przedmiotów i prowadzących
for s in `seq 1 4`;
do
spp --add --type topic --name "przedmiot $s"  --hours 30;
spp --add --type lecturer --name "prowadzący $s" --degree "mgr inż." \
        --room "`expr 500 + $s`";
done

echo edycja prowadzących
spp --edit --type lecturer --id 1 --topic 1,2
spp --edit --type lecturer --id 2 --topic 3,4
spp --edit --type lecturer --id 3 --topic 1
spp --edit --type lecturer --id 4 --topic 3

echo edycja przedmiotów
spp --edit --type topic --id 1 --date "czwartek 10:00" --topic-type egz
spp --edit --type topic --id 2 --date "środa 12:15" --topic-type zal --hours 15
spp --edit --type topic --id 3 --date "poniedziałek 8:15" --topic-type egz
spp --edit --type topic --id 4 --date "wtorek 8:15" --topic-type zal --hours 20

echo wypisanie przedmiotów
spp --cat --type topic

echo utworzenie studentów
for s in `seq 1 30`;
do
spp --add --type student --name "student $s" --index "`expr 300000 + $s`";
done
```

```
echo usuwanie studentów
spp --rm --type student --id 6
spp --rm --type student --id 16
spp --rm --type student --id 26

echo edycja studentów
spp --edit --type student --id 1 --name "Jan Kowalski" --topic 1,3,4
spp --edit --type student --id 10 --name "Ola Kowal" --topic 1
spp --edit --type student --id 30 --name "Michał Kowalczuk" --topic 1,2,3,4


spp --cat --type student

echo wypisanie rekordu studenta o id 10
spp --cat --type student --id 10

echo wypisanie powiązań dla studenta
spp --show-links --type student --id 10

echo wypisanie powiązań dla prowadzącego
spp --show-links --type student --id 1

echo wypisanie studentów i prowadzących powiązanych z przedmiotem
spp --show-links --type topic --id 3

echo wypisanie przedmiotów studenta
spp --show-links --type student  --id 10

echo wypisanie przedmiotów prowadzącego
spp --show-links --type lecturer  --id 1
```

4.3 Zgodność z założeniami

Program działa tak jak założono podczas analizy przedprojektowej.