TREŚĆ PROJEKTU

Lista cykliczna ze stosami.
Napisać program zarządzający strukturami dynamicznymi.
Niech istnieje pewna obrotowa tacka na ciasteczka różnego typu. Po jednej stronie tacki niech siedzi dostawca ciasteczek, a naprzeciw niego zjadacz ciasteczek. Dostawca może dokładać na tackę po ciasteczku, ale tylko na stosik ciasteczek tego samego typu. Jeśli stosiku brak, tworzymy nowy, przy czym musi on znaleźć miejsce pomiędzy stosikami na których znajdują się ciastka o nazwie alfabetycznie mniejszej i większej, niż nazwa ciasteczka dokładanego. Zjadacz może w dowolnej chwili obrócić do siebie stolik na pozycje dowolnego ciasteczka i zjeść ile mu się podoba. Program ma pozwalać w dowolnym momencie w pełni podglądać i modyfikować struktury danych. (Projekt 4) Zaimplementować zapis i odczyt dynamicznej struktury z dysku (binarny). Podanie przy uruchamianiu programu nazwy pliku z zapisem ma wczytać na starcie dany plik. Wymyślić dodatkowe dwa parametru uruchomienia i je obsłużyć. Nie wolno w programie stosować zmiennych globalnych. Program należy podzielić na pliki. Niepodane dane należy sobie wymyślić – wypełnić projektowane struktury jakimiś sensownymi danymi.

WYKORZYSTANE STRUKTURY

```
typedef struct CakesStackElement
{
	time_t created;
	struct CakesStackElement* next;
} CakesStackElement;

typedef struct CakesStack
{
	char name[MAX_NAME];
	int count;
	struct CakesStackElement* topOfStack;
	struct CakesStack* next;
} CakesStack;
```

Pierwsza struktura zawiera dane dotyczące elementu na stosie. Zawiera parametr dotyczący czasu dodania ciastka na stosik. Stos jest realizowany za pomocą listy. Druga struktura dotyczący listy cyklicznej ze stosami(nazwa stosu, ilość elementów na stosie).

PROGRAM

Nagłówki funkcji i struktury znajdują się w pliku header.h a ich definicje w header.c. W pliku main.c odbywa się testowanie programu, którego dokonuje użytkownik. Program posiada menu dzięki któremu użytkownik może wybierać jaką operację chce wykonać np. tworzenie nowego stosiku z ciastkami, jedzenie ciastek, zapisanie do pliku, wypisanie zawartości tacki . Maksymalna liczba ciastek na stosie wynosi 100 a minimalna 5.

FUNKCJE

void removeFromStack(CakesStack* stack); -usuwa ciastko ze stosu

void refill(CakesStack* iter); -uzupełnia stosik losową ilością ciastek danego rodzaju

void create(char* name); -tworzy stos

void eat(char* name, int count); -funkcja realizująca zjadacza, usuwa podaną ilość ciastek z danego stosu

void print(int val); -drukuje zawartość stosików z ciastkami

void clear(); -czyści pamięć

void save(char* name); -zapisuje aktualny stan programu do pliku o podanej nazwie

void read(char* name); -wczytuje stan programu z pliku