ŻYJEMY

- Programowanie obiektowe,
projekt semestralny

Alicji Danilczuk

1.Opis

Program ma za zadanie ułatwiać życie przede wszystkim osobom gotującym. Jedzenie jest ogromną częścią naszego życia. Ogromną i ważną. Nie znam osoby, która choć raz w tygodniu nie spędza kilkunastu minut bezskutecznie zastanawiając się nad tym co ugotować na obiad. Bo ile razy można smażyć mielone? Po którym dniu spędzonym w kuchni na gotowaniu ziemniaków nie możemy myśleć już o ich jedzeniu?

W tym momencie z pomocą przychodzi mój program. Baza potraw nie jest może jeszcze szeroka ale dzięki funkcji dodawania kolejnych dań daje użytkownikowi możliwość nieograniczonego rozwijania swojego menu. Powiększanie bazy danych sprawia nie tylko, że potencjalny kucharz gromadzi w jednym miejscu propozycje potraw, lecz prowadzi nas do kolejnej funkcji programu. Mianowicie do losowania. Na podstawie ustalanej przy uruchomieniu dziennej liczby kilo kalorii aplikacja wygeneruje nam całodzienny jadłospis. Szukanie pomysłu na obiad przestanie być dla nas takim problemem gdy w każdej chwili będziemy mogli wyświetlić sobie spis propozycji kulinarnych.

Liczenie kalorii jest tu sprawą drugorzędną- podane są one raczej orientacyjnie, przeliczane na uśrednione porcje. Mają wskazywać na różnice kaloryczne między daniami, a nie wspomagać dokładne wyliczenia podczas diety. Służą też zdrowemu rozplanowaniu posiłków- na każdą porę jedzenia program wylicza odpowiednią część dobowej ilości kilokalorii.

2.Przewodnik po programie

Zbudowany jest z 6 plików hpp/cpp oraz 5 plików tekstowych:

-Posilki.hpp	-Plik nagłówkowy zawierający deklaracje klas zajmujących się obsługą poszczególnych posiłków, to znaczy śniadania, drugiego śniadania, obiadu, przekąski i kolacji. Wszystkie są pochodnymi klasy posiłek i zawierają zdefiniowane w pliku cpp wirtualne metody z tej klasy.
-Menu.hpp	-Plik nagłówkowy zawierający deklaracje tylko jednej, lecz rozbudowanej klasy - Menu. Jest ona odpowiedzialna za zarządzanie działaniem całego programu. To znaczy za komunikację miedzy użytkownikiem a komputerem oraz edycję pól zachowujących ustalany jadłospis.
-Menu_operacje_na_listach.cpp	-Plik źródłowy zawierający definicje kilku metod klasy Menu. Umieszczone są w nim funkcje zajmujące się edycją listy składającej się na jadłospis.
-Menu_dialog_z_uzytkownikiem.cpp	-Plik źródłowy zawierający definicje pozostałych metod klasy Menu. Funkcje w nim umieszczone są odpowiedzialne za wyświetlanie w konsoli zapytań do użytkownika oraz przełączanie między metodami programu według jego wskazań.
-Posiłki.cpp	-Plik źródłowy zawierający definicje metod klasy posiłek oraz klas jej pochodnych.
-Main.cpp	-Plik źródłowy, który jest jedynie ramami programu. Uruchamiane z niego są metody klasy Menu, które później już same obsługują działanie programu.

Pliki tekstowe o nazwach: Sniadania, Drugie_sniadania, Obiady, Przekaski i Kolacje przechowują dania zapisane w postaci *ilość_kalorii; nazwa_dania;*

3.Struktura plików

Posiłki.hpp

```
class Posilek
{
 public:
   string posilek;
   string nazwa;
   int kalorie;

   friend ostream & operator<< (ostream &wyj, Posilek* s);

   Posilek() =default;
   Posilek(string a,string b,int k): posilek(a), nazwa(b), kalorie(k){};

   virtual Posilek* losuj(int, int) =0;
   virtual Posilek* wybierz(int) =0;
   virtual void dodaj_nowe() =0;
   int ilosc_kalorii(string);
   string nazwa_dania(string);
};
```

Klasa posiłek posiada trzy pola przechowujące, zgodnie z nazwą, informację który posiłek opisuje dany obiekt, nazwę przechowywanego dania i ilość jego kalorii.

Dalej przeciążony został operator wypisywania, tak by w dalszej części programu nie trzeba się było zastanawiać nad sposobem wyświetlania kolejnych pól obiektu.

Klasa ta posiada trzy metody wirtualne.
Losuj(int, int) losuje z pliku z danymi wiersz z proponowanym daniem.
Wybierz(int) wyświetla listę wszystkich dań zapisanych w pliku i pozwala użytkownikowi samemu wybrać to, które dopisze do jadłospisu.
Dodaj_nowe() daje użytkownikowi możliwość dopisania nowych dań do pliku z danymi, spośród których może później losować i wybierać.

Między poszczególnymi klasami pochodnymi różnica w implementacji jest w większości w plikach tekstowych z których czerpią dane.

Klasa posiłek ma jeszcze dwie metody przyspieszające odzyskiwanie z wiersza z bazy danych ilości kalorii oraz nazwy wybranego dania.

Menu.hpp

```
class Menu final
{
public:
   int max_kcal;
   list<Posilek*> lista;

   Menu() = default;

   void ustaw_kalorie();
   int ustaw_kalorie_kobieta(int a);
   int ustaw_kalorie_mezczyzna(int a);

   int okno_wejscia();
   int okno_wyboru_posilku();
   int okno_wyboru_czynnosci();
   void wejscie();
   void wybor_posilku();
   void wybor_czynnosci(Posilek*);

   void uzupelnij_liste();
   void dodaj_do_listy(Posilek*);
   void wyswietl_jadlospis();
   void losuj_jadlospis();
   void usuwanie_listy();
   void wyczysc();
   };
```

Klasa Menu posiada dwa pola.
Pierwsze- max_kcal ustalane podczas iterakcji z użytkownikiem w metodzie *ustaw_kalorie(),* przechowuje dzienną sugerowaną ilość kalorii na podstawie której układać będzie jadłospis.
Drugie pole jest wbudowaną listą w której zapisywane będą wyniki losowań oraz wyborów.
Lista ta tuż po uruchomieniu programu zostaje kolejno uzupełniona pięcioma obiektami wszystkich klas opisujących posiłki.

Metoda *dodaj_do_listy(Posiłek*)* nie tylko dodaje kolejne wybrane dania do listy jadłospisu ale też dba o to, by każdy posiłek pojawił się na niej tylko raz. Dodatkowo zwalnia pamięć przydzieloną elementom, które zastępujemy nowymi.

Natomiast pod koniec działania programu zwalnianiem pamięci zajmuje się funkcja *usuwanie_listy().*

W części operacji na listach znajdują się jeszcze metody wyświetlająca oraz losująca cały jadłospis.

W części dialogu z użytkownikiem funkcje dzielą się na te wyświetlające w konsoli możliwe do wyboru opcje (*okna*) oraz te które zajmują się dopasowaniem akcji do dokonanego wyboru (*wejscie() i wybory).*

Na każdym poziomie podejmowania decyzji co do kolejnych kroków użytkownik ma możliwość wyjścia z programu oraz wycofania się do wyższego poziomu.