PRG zadania dodatkowe

Instrukcje warunkowe

1. (CASUAL) Napisz program, który wypisze słownie dzień tygodnia na podstawie liczby podanej przez użytkownika (np. 1 - poniedziałek, 2 - wtorek itp). Użyj instrukcji switch.
2. (CASUAL) Napisz program, który na podstawie podanego numeru miesiąca i roku odpowie, ile dni ma dany miesiąc. Użyj instrukcji warunkowych if, else if, else. Pamiętaj o latach przestępnych.

Pętle

1. (CASUAL / HARDCORE) Napisz program, który sprawdzi czy podana liczba jest liczbą pierwszą. Dodatkowo:
   - program powinien zliczyć, ile razy zostały wykonane pętle użyte w kodzie.
   - porównaj liczbę wykonań z programami innych studentów.
   - Spróbuj zoptymalizować program by dawał dobry wynik przy jak najmniejszej liczbie wykonań pętli.

   Przykład wywołania:

   ```
   podaj liczbę: 16127
   to jest liczba pierwsza; wykonałem 1000 iteracji pętli.
   ```

2. (CASUAL / HARDCORE) Napisz program, który narysuje “choinkę nocą”, o wysokości, podanej w parametrze, zgodnie z przykładem:

   podaj wysokość: 6

   ```
   *********
   ****  ****
   ***    ***
   **      **
   *        *
   *********
   ```

Tablice i struktury

1. (HARDCORE) Napisz prostą grę “kółko i krzyżyk” (możliwa praca w grupach). Planszą może być tablica lub inna struktura danych. Program niech kolejno pyta gracza, gdzie postawić kółko / krzyżyk. I za każdym razem niech sprawdza, czy już ktoś wygrał (albo czy wszystkie pola są zajęte).
2. (HARDCORE) Napisz program z wykorzystaniem argc i argv, do obstawiania totolotka. Program niech losuje tyle liczb, ile poda użytkownik wywołujący program. Dla ułatwienia mogą to być liczby z przedziału 1 do 10, by łatwiej było zgadnąć. Przykład wywołania:

```
home/cxx/ build/totolotek.bin 1 2 3 4 5 6
wylosowane liczby to 1 2 3 7 8 9
trafiłeś 3 z 6
```

3. (CASUAL / HARDCORE) Stwórz strukturę: Samochód. Samochód powinien posiadać pola:
   - marka
   - model
   - rok produkcji
   - przebieg
   - cena wyjściowa
   - cena końcowa

   … oraz metody:
   - konstruktor (marka, model, rok, przebieg)
   - ustaw cenę wyjściową(cena)
   - oblicz cenę końcową (od ceny wyjściowej odejmujemy 10000 za każdy rok wieku samochodu oraz 3 za każdy kilometr przebiegu)

   Stwórz kilka różnych obiektów typu Samochód (o różnych parametrach) i przetestuj na nich obliczanie ceny.

4. (CASUAL / HARDCORE) Napisz program wypisujący w konsoli trójkąt Pascala o wysokości podanej jako parametr. W swoim programie wykorzystaj tablice do przechowywania informacji o liczbach tworzących trójkąt.
5. (CASUAL / HARDCORE) Słownik. Stworzyć strukturę przechowującą słowa typu string (może to być tablica).Słownik powinien posiadać następujące funkcje:

- proste menu tekstowe do wydawania poleceń (np: 1-dodaj słowo; 2-usuń słowo: 0-wyjdź)
- dodawanie słów do słownika
- usuwanie słów ze słownika
- wyszukanie słowa w słowniku
- wypisywanie całej zawartości słownika
- sortowanie alfabetyczne zawartości słownika
- prosta walidacja (jeśli słowo już jest, nie doda, tylko wypisze komunikat; jeśli słowa nie ma, nie usunie, tylko wyświetli komunikat)

6. (HARDCORE) Aplikacja naśladująca listę kontaktów w telefonie. Program ma działać w nieskończonej pętli, dopóki użytkownik nie wpisze z klawiatury, że chce wyjść z programu.

- Należy stworzyć strukturę danych (np. struct Kontakt), która będzie przechowywać informacje wprowadzone przez użytkownika: numer telefonu, nazwisko oraz liczbę połączeń. Książka telefoniczna będzie tablicą obiektów typu Kontakt.
- funkcja wypisz, która wypisze całą książkę telefoniczną wraz z liczbą połączeń
- funkcja szukaj, która sprawdzi czy dany numer telefonu jest w książce (i wypisze go)
- funkcja dodaj, która dodaje numer telefonu do książki (wraz z nazwiskiem) i ustawi liczbę połączeń z tym numerem na zero. Oczywiście najpierw sprawdziwszy, czy taki numer już nie jest dodany
- funkcja usuń, która usunie kontakt wg podanego numeru telefonu.
- funkcja połącz, która zapyta o nazwisko, a następnie zwiększy liczbę połączeń z odpowiednim kontaktem o 1.

Przykład wywołania:

```
Co chcesz zrobić? 1 - szukaj numeru, 2 - dodaj numer, 3 - usuń
numer, 4 - połącz z kontaktem, 5 - wypisz numery, 0 - wyjdź
> 1
Podaj numer do wyszukania:
> 444
Znaleziono: 444, Jan Kowalski, połączenia: 4

Co chcesz zrobić? 1 - szukaj numeru, 2 - dodaj numer, 3 - usuń
numer, 4 - połącz z kontaktem, 5 - wypisz numery, 0 - wyjdź
> 4
Podaj nazwisko do połączenia:
> Jan Kowalski
Znaleziono: 444, Jan Kowalski, połączenia: 4

Łączę: liczba połączeń: 5

Co chcesz zrobić?...
```

7. (HARDCORE) Aplikacja szczepionkowa. Program mający na celu sprawdzanie, czy dana osoba ma ważne szczepienie albo czy może skorzystać z dawki przypominającej. W programie powinny znaleźć się następujące elementy

- struktura danych Pacjent: login, hasło, nazwisko, tablica dat szczepień
- funkcja main() z nieskończoną pętlą i wybieraniem poleceń liczbami
- funkcja rejestruj(), w której możemy podać dane nowego pacjenta
- funkcja zaloguj(), która sprawdza podany login i hasło. Jeśli są poprawne, pokazuje dane pacjenta i pozwala na uruchomienie funkcji związanych ze szczepieniami
- funkcja wyloguj()
- funkcja wypisz() - wypisuje daty zapisanych szczepień aktualnie zalogowanego pacjenta

- funkcja czyMoge() - sprawdza, czy aktualnie zalogowany pacjent może się zaszczepić. Zależnie od tego, ile elementów ma wpisane w tablicy szczepień, funkcja zwraca true: (0 dawek: zawsze; 1 dawka: po upływie miesiąca od daty; 2 lub więcej dawek: po upływie pół roku od daty ostatniej dawki)
- funkcja zaszczep() - po sprawdzeniu, czy aktualnie zalogowany pacjent może się zaszczepić, dopisuje nową datę do tablicy dat szczepień
- funkcja czyWaznyCertyfikat() - sprawdza, czy aktualnie zalogowany pacjent ma ważny certyfikat szczepień (min. 2 dawki, ostatnia mniej niż 12 miesięcy wcześniej niż dziś)

Podpowiedź: aby sprawnie testować daty (zwłaszcza kilkumiesięczne różnice między nimi), możemy:

- Podawać datę (np. w funkcji zaszczep()) i pozwalać na wpisanie dowolnej daty (oczywiście sprawdzając ją przy pomocy funkcji czyMoge())
- Przechowywać w programie daty jako struktury {dzień, miesiąc, rok}, co ułatwi porównywanie.

```
Co chcesz zrobić? 1 - zaloguj, 2 - zarejestruj , 0 - wyjdź z
programu.
> 1
Podaj login:
> jan
Podaj hasło:
> abcde
Logowanie poprawne. Witaj jan (Jan Kowalski).

Co chcesz zrobić? 1 - pokaż daty moich szczepień, 2 - sprawdź czy
mogę się zaszczepić, 3 - zaszczep, 4 - sprawdź ważność
certyfikatu, 5 - wyloguj, 0 - wyjdź z programu.
> 3
Podaj datę szczepienia (YYYY-mm-dd):
> 2021-10-10
Data poprawna, zapisuję szczepienie.

Co chcesz zrobić...
```

8. (CASUAL / HARDCORE) Trójkąt. Stwórz klasę Trójkąt, zawierającą pola a, b, c, odpowiadające długościom boków trójkąta. Klasa powinna także zawierać metodę sprawdź, która wyświetla informacje, jaki to jest rodzaj trójkąta (ostrokątny, prostokątny, rozwartokątny) na podstawie długości jego boków. W funkcji main swojego programu stwórz kilka obiektów klasy Trójkąt o różnych bokach. I sprawdź czy prawidłowo działa funkcja “sprawdź”.
9. (CASUAL / HARDCORE) Banknoty i monety. Napisz program, który pobiera od użytkownika kwotę w złotych (int), a następnie wyświetla informację, jakimi monetami i banknotami najlepiej tę kwotę uiścić (by użyć ich jak najmniej). W programie użyj tablicy z dostępnymi nominałami (1, 2, 5, 10, 20, 50, 100, 200, 500 zł).

```
Podaj kwotę w złotych:
> 159

Twoje nominały to:
100 - 1 szt
50 - 1 szt
5 - 1 szt
2 - 2 szt
```

10. (CASUAL / HARDCORE) Macierze. Z użyciem pętli oraz tablic, napisz programy tworzące macierze o rozmiarze podanym przez użytkownika, według przedstawionych wzorów.

a)	b)	c)	d)
[1, 2, 3, 4]	[0, 0, 0, 1]	[2, 4, 6, 8]	[4, 3, 2, 1]
[2, 3, 4, 5]	[0, 0, 1, 2]	[4, 4, 6, 8]	[3, 2, 1, 0]
[3, 4, 5, 6]	[0, 1, 2, 3]	[6, 6, 6, 8]	[2, 1, 0 ,0]
[4, 5, 6, 7]	[1, 2, 3, 4]	[8, 8, 8, 8]	[1, 0, 0, 0]

(przykłady dla rozmiaru 4).

```
Podaj wariant macierzy (a,b,c,d)
> a

Podaj rozmiar macierzy (1-10)
>5

1 2 3 4 5
2 3 4 5 6
3 4 5 6 7
4 5 6 7 8
5 6 7 8 9
```