ZADANIE 1

Pytanie **1**

Zakończone

Ocena: 29,00 z 30,00

Oflaguj pytanie

Struktura przechowująca dane macierzy 2D na stercie zdefiniowana jest następująco:

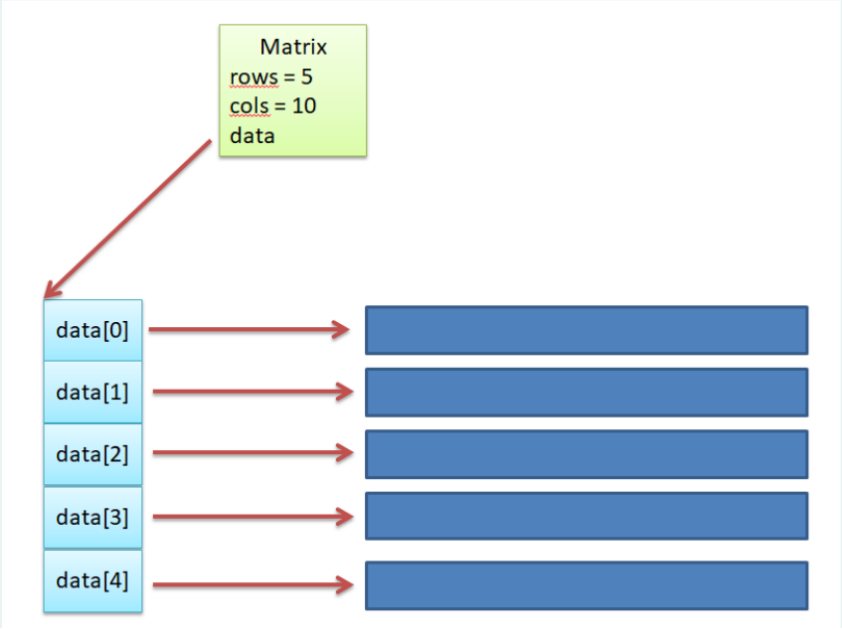
```
typedef struct _matrix{
   int rows;
   int cols;
   double**data;
}Matrix;
```

Wskaźnik data wskazuje na początek tablicy wskaźników typu double*. Każdy z nich wskazuje dane wiersza macierzy. Czyli data[i] to i-ty wiersz macierzy.

Załóż, ze zaimplementowana jest funkcja

int create_matrix(Matrix*pmatrix, int rows, int cols)

Która wypełnia wskazaną strukturę danymi, równocześnie alokując pamięć dla alokując pamięć dla wszystkich tablic na stercie. Elementy macierzy mają wartość 0. Funkcja zwraca 1, jeżeli operacja zakończyła się sukcesem, 0 w przypadku błędnych parametrów (rows,cols<1).



Napisz następujące funkcje:

a) int get(Matrix*pm,int row,int col,double*pvalue)

Oraz

int set(Matrix*pm,int row,int col,double value)

Funkcje pozwalają na dostęp do elementu macierzy. Zwracają 0, jeżeli podano niewłaściwe wartości wiersza lub kolumny (albo data jest wskaźnikiem zerowym),natomiast 1, kiedy dostęp został zrealizowany.

b) void create_identity_matrix(Matrix*pm,int size)

Alokuje pamięć dla macierzy jednostkowej o rozmiarze size x size, umieszcza tam dane (jedynki na przekątnej) i wpisuje informacje do struktury wskazywanej przez pm

c) double dot_prod_vect(const double*v1,const double*v2,int size)

Funkcja oblicza iloczyn skalarny dwóch wektorów (czyli sumę iloczynów elementów o tych samych indeksach).

d) void mul_vect(double *result, const Matrix*pm, const double*vect)

Funkcja oblicza iloczyn macierzy pm i wektora vect oraz umieszcza wynik w wektorze result. Nie podano rozmiarów wektorów, ponieważ wynikają one z rozmiarów macierzy. Użyj wewnątrz napisanej wcześniej funkcji dot_prod_vect()

e) void max_element_in_rows(double*maxdata,const Matrix*pm)

Funkcja wypełnia tablicę maxdata największymi elementami znalezionymi w wierszach. Czyli maxdata[i] to największy element w wierszu o indeksie i. Rozmiar maxdata nie jest przekazywany, wynika z rozmiaru macierzy

ZADANIE 2

Pytanie **1**
Zakończone
Ocena: 24,00 z 35,00
Oflaguj pytanie

Zadanie odnosi się do listy, która ma być implementacją kolejki FIFO. Każdy element kolejki zawiera w strukturze *Data* dane o jednym ciągu pomiarów: wynikach zapisanych na stercie w tablicy *results* i ich liczbie *len*.

```
typedef struct  {
   double *results;
   int len;
} Data;
```

Na kolejce mogą być wykonywane tylko operacje:
tworzenia i inicjowania listy,
dodania elementu na koniec kolejki,
usunięcia elementu z początku kolejki,
odczytu danych z początkowego elementu kolejki,
obliczenia danej wymagające dostępu do każdego elementu kolejki,
likwidacji kolejki (niezależnie od jej stanu).

Należy:
(5 pkt.) Uzupełnić deklaracje dwóch "oszczędnych" struktur w taki sposób, aby nie zawierały pól niekoniecznych, a złożoność obliczeniowa operacji na elementach listy była O(1). Łącząc te deklaracje z definicją nowych typów należałoby uzupełnić wyrażenia:
- deklarację typu elementu listy (kolejki)
`typedef struct tagQueueElement { Data data; .... } QueueElement;`
- deklarację typu listy
`typedef struct tagQueue { .... } Queue;`
W przypadku, gdyby struktura *tagQueue* miała zawierać tylko jedno pole, można strukturę zastąpić prostą zmienną - zmienną typu *Queue*, czyli
`typedef ..... Queue;`

(5 pkt.) Napisać funkcję `void free_queue(Queue *pqueue);` która likwiduje kolejkę tzn. zwalnia pamięć przydzieloną wszystkim elementom na kolejki. Nie zwalnia pamięci przydzielonej danym (tj. Wynikom pomiarów).

(5 pkt.) Napisać funkcję `void push_copy(Queue *pqueue, const Data*pdata);` która wstawia do kolejki element z danymi przekazanymi przez argument *pdata*.

(5 pkt.) Napisać funkcję `int peek(const Queue *pqueue, Data *pdata);` która odczytuje dane z elementu najdłużej oczekującego w kolejce i przepisuje je pod adres wskazany parametrem *pdata*. Funkcja zwraca 0, jeżeli kolejka jest pusta, a 1 w przeciwnym przypadku.

(7 pkt.) Napisać funkcję `int pop(Queue *pqueue, Data*pdata);` która
- usuwa z kolejki element najdłużej czekający,
- zwalnia pamięć elementu,
- przekazuje jego dane do zmiennej wskazywanej przez *pdata*,
Funkcja nie zwalnia pamięci, w której są zapisane wyniki pomiarów wskazywane w zwalnianym elemencie.
Funkcja zwraca -1 gdy kolejka była już pusta, 0 gdy jest pusta po usunięciu elementu, 1 w pozostałych przypadkach.

(5 pkt.) Funkcję `int get_total_count(const Queue *pqueue)`, która zwraca liczbę wszystkich pomiarów, do których mają dostęp elementy czekające w kolejce.

ZADANIE 3

Pytanie **1**
Zakończone
Punkty: 35,00
Oflaguj pytanie

Informacje z dowodu rejestracyjnego pojazdu są zapisane w strukturze *Vehicle*. Założenia:

Każda struktura typu `struct Vehicle` może zawierać informacje o pojazdach jednego z trzech typów pojazdów - osobowego / autobusu / ciężarowego.

Tego samego typu dane w takich strukturach dla wszystkich typów pojazdów to:

- właściciel (nazwa firmy lub imię i nazwisko - zapisane w jednym łańcuchu na stercie),
- data najbliższego przeglądu - zapisana w 3-elementowej tablicy typu *int* zapisanej w jednym polu struktury (w kolejności: d,m,y),
- typ pojazdu (dana typu wyliczeniowego z stałymi: car, *bus, truck*),
- typ napędu (dana typu wyliczeniowego z stałymi: *electric, hybrid, combustion*).

Dane specyficzne dla typu pojazdu to:
- dla osobowego: maks. liczba osób (*int*) oraz moc silnika (*float*),
- dla autobusu: liczba miejsc siedzących (*int*) i liczba miejsc stojących (*int*),
- dla ciężarowego: pojemność silnika (*float*) i dopuszczalny nacisk na oś (*float*).

Należy napisać:

(5 pkt.) Deklarację struktury *Vehicle* oraz wszystkich jej składników typu struktura, unia lub wyliczeniowy;

(8 pkt.) Funkcję `void new_owner(struct Vehicle *pvehicle);` która wczytuje z klawiatury nazwę nowego właściciela, usuwa informację o poprzednim właścicielu oraz odpowiednio modyfikuje zawartość pól struktury wskazywanej przez *pvehicle*.

(12 pkt.) Funkcję szukającą spóźnialskich
`int delayed (struct Vehicle *pvehicle, int size, int *base_date, char ***pdelayed_owners);`
która znajduje w tablicy *pvehicle* (o długości *size*) wszystkie struktury pojazdów, których data przeglądu jest wcześniejsza niż przekazana przez parametr *base_date* (w formacie takim w jakim jest zapisana w strukturze vehicle).
Funkcja tworzy tablicę *delayed_owners*, do której przepisuje adresy łańcuchów zawierających nazwy właścicieli pojazdów z "przekroczoną" datą. Adres tej tablicy przekazuje korzystając z parametru *pdelayed_owners*.
Funkcja zwraca liczbę opóźnień. W przypadku zerowej liczby opóźnień, funkcja zamiast adresu tablicy powinna przekazać adres zerowy.

(6 pkt.) Funkcję `void print_pojazd(struct Vehicle *pvehicle);` która wyprowadza wszystkie dane o pojeździe. Typ pojazdu oraz typ napędu może być wypisany jako kod liczbowy (wartość liczbowa zmiennej typu wyliczeniowego).