Wstęp do programowania w języku C

Grupa TDr

Lista 7.

1. *(10 punktów w trakcie pracowni, później 5 punktów)*
Zadanie polega na napisaniu pięciu funkcji działających na poniższej reprezentacji tablicy. Jednej drukującej tablicę i czterech biorących za jeden z argumentów wskaźnik na funkcję przyjmującą `int` i zwracającą `int` w przypadku `map`, a zwracającą `bool` w przypadku `filter`. Przykład działania podany jest na końcu zadania.

```
typedef struct {
    int* memory;
    int size;
} Int_array;
```

Funkcja o deklaracji `void print_array(Int_array array);` powinna wydrukować w jednej linii elementy tablicy oddzielone przecinkiem ze spacją (przecinek po ostatnim elemencie jest dozwolony), a całość otoczoną nawiasami kwadratowymi (czyli w pythonowym formacie list).

Funkcja `Int_array map_inplace(int f(int), Int_array array);` ma zastąpić każdy element tablicy `array` na wartość funkcji `f` dla tego elementu i zwrócić tę samą tablicę z nowymi wartościami elementów.

Funkcja `Int_array map_alloc(int f(int), Int_array array);` ma zwrócić taką samą tablicę (jak zwracana w poprzednim akapicie), ale w nowo zaalokowanej pamięci, nie zmieniając przy tym tablicy `array`.

Funkcja `Int_array filter_inplace(bool f(int), Int_array array);` ma zostawić w tablicy tylko te liczby, dla których funkcja `f` zwraca wartość prawdziwą, czyli zsunąć je na początek tablicy, a w wynikowej tablicy zaznaczyć ich liczbę.

Funkcja `Int_array filter_alloc(bool f(int), Int_array array);` ma zwrócić taką samą tablicę (jak zwracana w poprzednim akapicie), ale w nowo zaalokowanej pamięci, nie zmieniając przy tym tablicy `array`.

Twoje funkcje nie powinny alokować więcej pamięci niż potrzebują.

Przykład. Uruchomienie funkcji `main` z poniższego kodu:

```
int square(int n) { return n * n; }
bool is_odd(int n) { return n & 1; }
```

```
int main(){
    int array_mem[] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
    Int_array array = {.memory = array_mem, .size = 8};
    print_array(map_alloc(square, array));
    print_array(filter_alloc(is_odd, array));
    print_array(map_inplace(square, array));
    print_array(filter_inplace(is_odd, array));
}
```

powinno wypisać np.:

```
[0, 1, 4, 9, 16, 25, 36, 49, ]
[1, 3, 5, 7, ]
[0, 1, 4, 9, 16, 25, 36, 49, ]
[1, 9, 25, 49, ]
```

2. *(10 punktów)*

Dana jest następująca definicja typu `vec3d` reprezentującego trójwymiarowe wektory.

```
typedef union {
    struct { double x, y, z; };
    double c[3];
} vec3d;
```

Napisz funkcje o podanych deklaracjach i wymaganiach (można korzystać z bibliotek):

1. Funkcja `vec3d_of(double x, double y, double z);` ma zwracać wektor o zadanych współrzędnych.

2. Funkcja `vec3d vec3d_trace(vec3d u);` ma wypisywać w czytelnej postaci współrzędne wektora i znak końca linii oraz zwracać przyjmowany wektor.

3. Funkcja `vec3d vec3d_sum(vec3d u, vec3d v);` ma zwracać sumę wektorów.

4. Funkcja `vec3d vec3d_diff(vec3d u, vec3d v);` ma zwracać różnicę wektorów.

5. Funkcja `double vec3d_dot(vec3d u, vec3d v);` ma zwrócić iloczyn skalarny wektorów. Iloczyn skalarny wektorów $[a, b, c]$ i $[x, y, z]$ to $ax + by + cz$.

6. Funkcja `double vec3d_length(vec3d u);` ma zwrócić długość wektora. Wykorzystaj do tego funkcję `vec3d_dot`.

7. Funkcja `bool nearly_equal(vec3d u, vec3d v, double eps);` ma mówić, czy długość różnicy wektorów `u` i `v` jest mniejsza lub równa `eps`. Wykorzystaj do tego odpowiednie wcześniej zdefiniowane funkcje.

8. Funkcja `vec3d vec3d_cross(vec3d u, vec3d v);` ma zwrócić iloczyn wektorowy wektorów. Iloczyn wektorowy wektorów $[a, b, c]$ i $[x, y, z]$ to $[bz - cy, cx - az, ay - bx]$. Spróbuj użyć pętli.

9. Funkcja `test_vec3d_cross();` ma testować poprzednią funkcję zawierając co najmniej 4 testy następującej postaci.

```
assert(nearly_equal(
  vec3d_cross(vec3d_of(3.,2.,-2.), vec3d_of(1.,0.,5.)),
  vec3d_of(10., -17., -2.),
  1e-12));
```

3. *(10 punktów)*
w serwisie SKOS