Języki Programowania, Zadanie 2

(1) Należy stworzyć strukturę **Punkt** reprezentującą punkt w przestrzeni dwuwymiarowej. Struktura ta powinna zawierać dwa pola typu double (składowe x i y) :

- **double x;**
- **double y;**

A program powinien również zawierać następujące funkcje:

- **void zapisz(Punkt *p, double xx, double yy)** - umożliwiająca ustalenie danego punktu (wartości podane jako argumenty należy wpisać do składników struktury; struktura podana przez wskaźnik)
- **void wypisz(Punkt p)** - wypisująca dany punkt na ekran w formacie [x,y]

W programie głównym należy stworzyć pojedynczy obiekt typu Punkt (x = 1, y = 4) i wypisać go na ekran bez użycia funkcji zapisz/wypisz. **(1.5 p)**
Następnie wczytać dwie liczby z klawiatury, odpowiadające x oraz y i stworzyć drugi punkt z podanymi wartościami, również wypisać go na ekran używając funkcji zapisz/wypisz. **(1.5 p)**

(2) Dopisać funkcję składową **zwracającą** odległość punktu na którym wywołana jest metoda od punktu zadanego jako argument funkcji
[typ zwracany] *odleglosc*(double X, double Y) | by użyć pierwiastka należy załączyć bibliotekę math.h:
#include <math.h>. Przy kompilacji trzeba też dodać flagę **-lm** !!!
Przetestować ja w funkcji main() wypisując na ekran odległość pierwszego punktu od punktu [5,6]. (1 p)

UWAGA! Jeśli w treści zadania pojawia się prośba o napisanie funkcji która „**zwraca**" wartość, funkcja powinna tę wartość zwracać (poprzez return bądź parametr, a nie jedynie wypisywać na ekran!)

(3) Dopisać następującą funkcje składowe:
- *dodającą wielkości liczbowe = zwiększające wartości współrzędnych punktu:*
void dodaj(Punkt *p, double xx, double yy) **(0.5 p)**

- *przypisującą wartości zadanego punktu (zmianiający wartości x* i *y punktu na którym funkcja jest wywoływana) :*
void kopiuj(Punkt *p, Punkt p2) **(0.5 p)**

Przetestować obie w funkcji main() wypisując na ekran punkt pierwszy po dodaniu do niego [2,2]
i punkt drugi, po przypisania mu wartości z punktu pierwszego.

Pamiętajcie, żeby regularnie zapisywać i kompilować wasze programy!

Po co wskaźniki?

1. Modyfikację oryginalnych danych

W C/C++ funkcje domyślnie przekazują argumenty przez wartość, czyli tworzą kopię danych. Jeśli funkcja ma zmienić zawartość struktury (lub innego obiektu), trzeba przekazać adres oryginału — czyli wskaźnik.

2. Oszczędność pamięci i czasu

Przekazywanie dużych struktur przez wartość powoduje kopiowanie całej zawartości — co może być kosztowne. Przekazanie wskaźnika (zajmującego tylko kilka bajtów) jest znacznie bardziej wydajne.

3. Dostęp do dynamicznie alokowanych danych

Jeśli obiekt został utworzony dynamicznie (np. przez malloc / new), można się do niego odwołać tylko przez wskaźnik. W takich przypadkach wskaźnik jest jedynym sposobem na przekazanie obiektu do funkcji.

Jak tworzyć struktury w języku C

1. Zdefiniuj strukturę

Użyj słowa kluczowego `**struct**`, aby utworzyć nowy typ danych zawierający kilka pól różnych typów.

```
struct Student {
    int id;
    int age;
    float grade;
};
```

2. Uprość zapis za pomocą typedef

Aby nie powtarzać słowa `struct` przy każdej deklaracji, można użyć `typedef`:

```
typedef struct {
    int id;
    int age;
    float grade;
} Student;
```

3. Utwórz zmienną typu struktura

Po zdefiniowaniu typu można tworzyć zmienne:

```
Student s1;
```

4. Inicjalizuj pola struktury

Przypisanie wartości do pól po kolei:

```
s1.id = 1001;
s1.age = 21;
s1.grade = 4.5;
```

5. Dostęp do pól struktury

Do pól struktury odwołujemy się za pomocą operatora `.`:

```
printf("Student ID: %d, wiek: %d, ocena: %.1f\n", s1.id, s1.age, s1.grade);
```

6. Struktury i wskaźniki

Jeśli masz wskaźnik do struktury, użyj operatora `->`:

```
Student *ptr = &s1;
printf("Ocena przez wskaźnik: %.1f\n", ptr->grade);
```

7. Przekazywanie struktury do funkcji

Struktury można przekazywać do funkcji na dwa sposoby:

a) Przez wartość – funkcja otrzymuje kopię obiektu (oryginał nie zostanie zmieniony):

```
void drukuj(Student s) {
   printf("ID: %d, wiek: %d, ocena: %.1f\n", s.id, s.age, s.grade);
}

drukuj(s1);
```

b) Przez wskaźnik – funkcja otrzymuje adres obiektu (może zmieniać oryginalne dane):

```
void popraw_ocene(Student *s, float nowa_ocena) {
    s->grade = nowa_ocena;
}

popraw_ocene(&s1, 5.0);
printf("Po poprawie: %.1f\n", s1.grade);
```