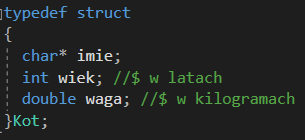
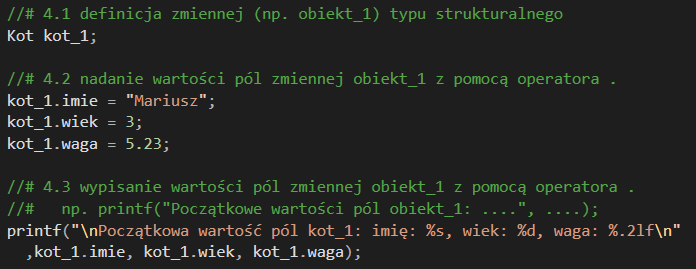
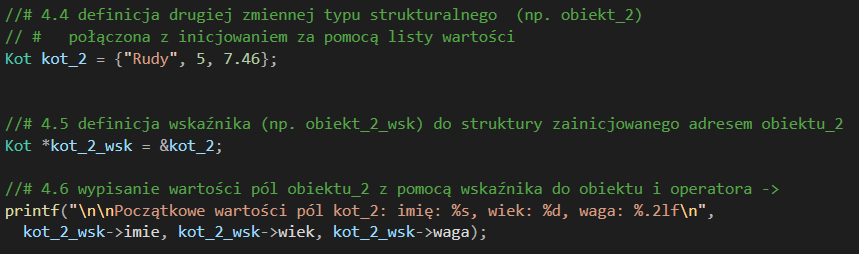
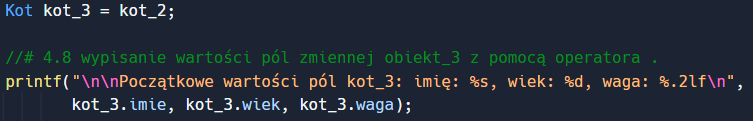
Anna Jasielec

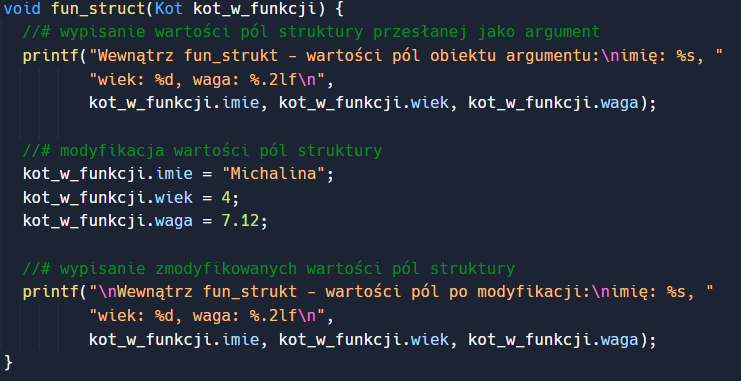
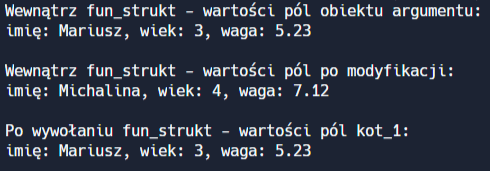
Podstawy programowania, grupa nr 4

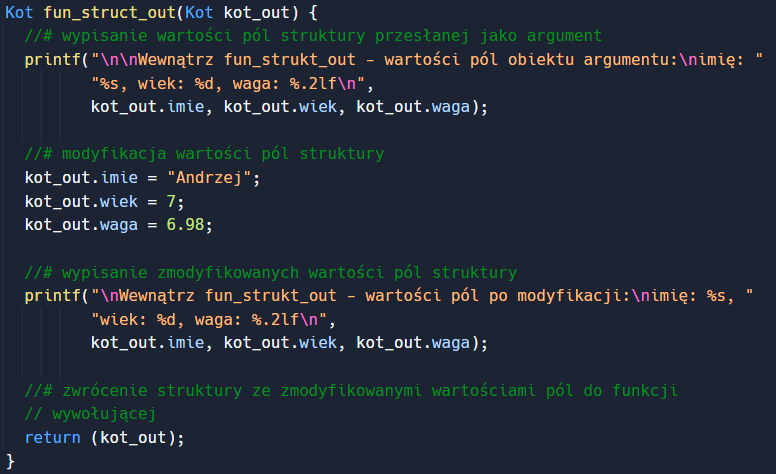
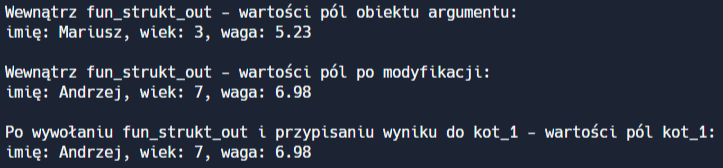
Sprawozdanie z laboratorium nr 10, 11

**Cel laboratorium:** Opanowanie podstaw tworzenia i wykorzystania struktur w C.

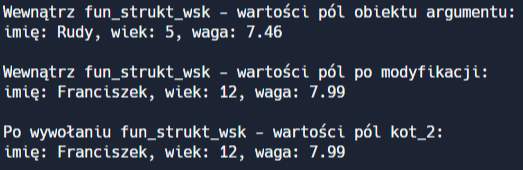
**Przebieg zajęć:**Utworzyłam katalog roboczy lab\_10 i skopiowałam ze strony przedmiotu plik struktury\_szablon.c. Zaprojektowałam własną strukturę, której parametry są zmiennymi różnych typów. Wykorzystałam do tego funkcjonalność typedef.  
  
Następnie zajęłam się pisaniem programu z funkcją main. Zdefiniowałam *kot\_1* będący strukturą zaprojektowanego typu i nadałam jego parametrom wartości posługując się operatorem składowych (**.**). Wypisałam na ekranie wartości parametrów posługując się operatorem składowych (**.**).  
  
  
Zdefiniowałam *kot\_2* będący strukturą, inicjując jednocześnie wartości składowych oraz zdefiniowałam wskaźnik do struktury inicjując adresem kot\_2. Wypisuje na ekranie zawartość struktury posługując się wskaźnikiem i operatorem **–>**.  


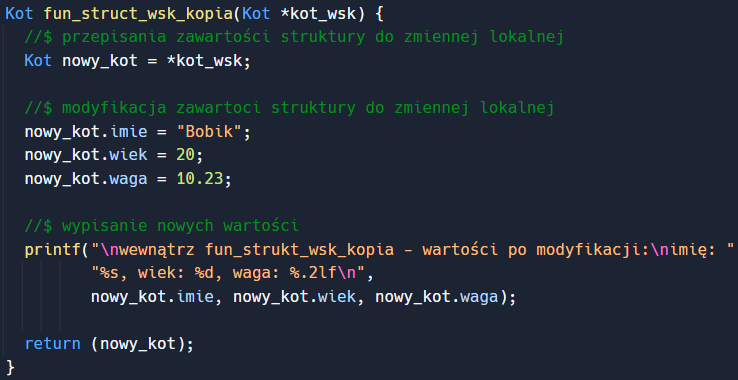
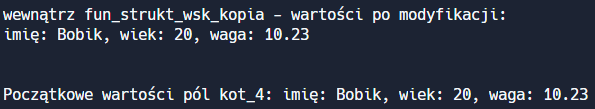
Następnie zdefiniowałam kot\_3 będący strukturą, inicjując jednocześnie wartości składowych poprzez przypisanie (skopiowanie) wartości pól z kot\_2. Wypisuje na ekranie wartości pól obiektu\_3 posługując się operatorem składowych (**.**).  
  

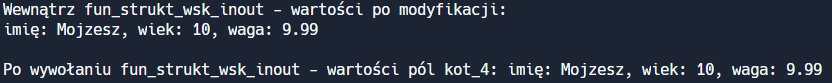
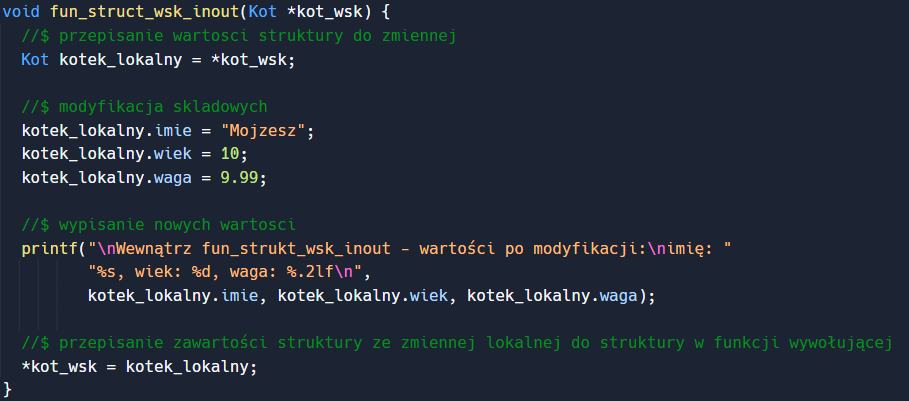

Zmodyfikowałam program pisząc kolejne funkcje, wywoływanych następnie przez funkcje main. Napisałam funkcję, która przyjmuje jako argument wejściowy obiekt będący strukturą, modyfikuje składowe struktury i wypisuje na ekranie nowe wartości. W funkcji main dodałam wywołanie tej funkcji z kot\_1 jako argumentem. Ponownie wypisałam wartości składowych kot\_1 po powrocie z funkcji.  
  


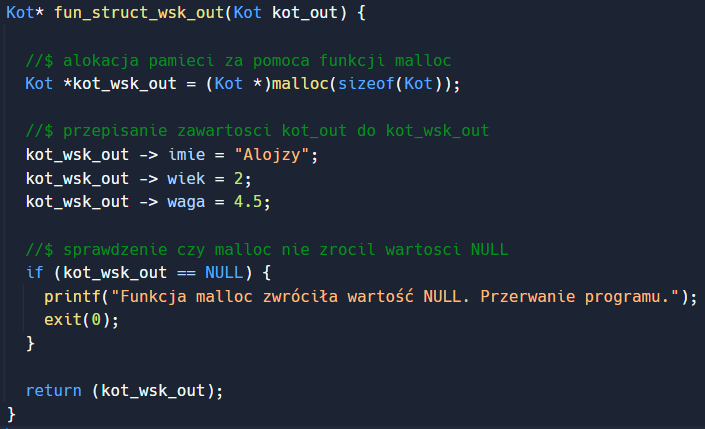
Kolejna funkcja przyjmuje obiekt będący strukturą jako argument wejściowy, modyfikuje składowe struktury, wypisuje na ekranie nowe wartości i zwraca obiekt ze zmodyfikowanymi wartościami. Uzupełniłam funkcje main o wywołanie tej funkcji z kot\_1 jako argumentem. Przypisałam zwracaną strukturę kot\_1 i wypisałam wartości pól kot\_1 po skopiowaniu.  
  


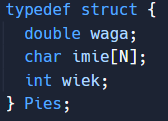
Następna funkcja przyjmuje jako argument wejściowy wskaźnik do obiektu będącego strukturą, modyfikuje składowe struktury i wypisuje na ekranie nowe wartości. W funkcji main wywołałam funkcję z kot\_2 jako argumentem i wypisałam wartości składowych po powrocie z funkcji.  

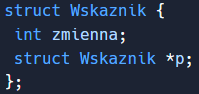
Następnie napisałam funkcję, która przyjmuje jako argument wejściowy wskaźnik do obiektu będącego strukturą, przepisuje zawartości struktury do zmiennej lokalnej, modyfikuje składowe tej zmiennej, wypisuje na ekranie nowe wartości i zwraca strukturę będącą zmienną lokalną. W funkcji main dodałam wywołanie funkcji z adresem kot\_3 jako argumentem, przypisałam zwracaną przez funkcję strukturę do nowej zmiennej kot\_4 zaprojektowanego typu.  
  


Kolejna funkcja przyjmuje wskaźnik do obiektu będącego strukturą jako argument, przepisuje zawartość struktury do zmiennej lokalnej posługując się operatorami przypisania i wyłuskania, modyfikuje składowe struktury będącej zmienną lokalną i przepisuje zawartość struktury ze zmiennej lokalnej do struktury w funkcji main. Funkcję main uzupełniłam o wywołanie funkcji z adresem kot\_4 jako argumentem.  


Następna funkcja przyjmuje jako argument strukturę, dokonuje alokacji pamięci dla nowej struktury, przepisuje zawartość tej struktury do nowej struktury w obszarze dynamicznym i zwraca wskaźnik do zaalokowanej struktury. Sprawdzenie czy nie została zwrócona wartość NULL, zwolnienie pamięci w funkcji main.  
   
  
  


Zaprojektowałam nowy typ strukturalny zawierającej pola, pośród których znajduje się tablica znaków.  


Zbadałam za pomocą operatora sizeof rozmiar pojedynczej zmiennej zaprojektowanego typu dla N równych 1,2,3,4,5 itd.  
  
Wyrównanie jest zawsze do największego obsługiwanego na danej architekturze typu prostego (char-1, int-4, double-8).

Zaprojektowałam nowy typ strukturalny, którego jednym z pól jest wskaźnik do zmiennych tego właśnie typu.  
****  
Polem nie może być zmienna tego samego typu, ponieważ wtedy struktura by sie zapętlała, natomiast możemy zdefiniować wskaźnik do zmiennej typu, wtedy wskazuje on na jej miejsce.

**Wnioski:**

* Słowo kluczowe *typedef* umożliwia przypisanie własnej nazwy do istniejących typów.
* Parametrom struktury można nadać wartość za pomocą operatora **.** (zmienna.parametr = …), za pomocą listy wartości (zmienna = {… , … , …}) lub poprzez skopiowanie (zmienna1 = zmienna2).
* Posługując się wskaźnikiem do struktury zainicjowanego adresem obiektu, zamiast operatora **.** użyjemy **->** (lub *(\*wskaźnik).parametr* = …).
* Można zmienić wartości parametrów struktury tworząc funkcję, która jako argument przyjmuje obiekt będący strukturą i zwraca obiekt ze zmodyfikowanym wartościami, zwrócony obiekt należy skopiować do obiektu użytego jako argument. Szybszym sposobem jest stworzenie funkcji, która przyjmuje wskaźnik do obiektu będącego strukturą i za jego pomocą modyfikuje wartości parametrów obiektu.
* Umiejętne wykorzystanie wskaźników ułatwia operacje na obiektach będących strukturami.
* Funkcji *malloc* służy do alokowania pamięci, należy pamiętać o sprawdzeniu, czy nie została zwrócona wartość NULL oraz o zwolnieniu pamięci.
* Można sprawdzić rozmiar struktury za pomocą *sizeof(Struktura),* wyrównanie jest zawsze do największego obsługiwanego na danej architekturze typu prostego (char-1, int-4, double-8).
* Polem struktury nie może być zmienna tego samego typu, ponieważ wtedy struktura by sie zapętlała, natomiast możemy zdefiniować wskaźnik do zmiennej tego typu.