Struktury i funkcje

Informatyka

Materiały

Struktury:

- **Struktura** jest obiektem złożonym z jednej lub wielu zmiennych zgromadzonych pod jedną nazwą („rekordy” w Pascalu).
- Ułatwiają zorganizowanie skomplikowanych danych (traktowane są jako jeden obiekt a nie zestaw oddzielnych obiektów).
- Przykład – atrybuty opisujące pracownika w firmie: imię, nazwisko, stanowisko, data zatrudnienia, pensja, inne.
- Struktura jest więc zestawem pól (nazywane składowymi struktury), gdzie pole może być daną lub strukturą danych.
- Co można robić ze strukturami?
    - przypisywać jedną do drugiej
    - kopiować jedną do drugiej
    - zwracać jako wartość funkcyjną
    - przekazywać do funkcji
- **Definicja struktury:**

  struct *typ_struktury* { *lista_pól* } *lista_identyfikatorów* ;
- Etykieta struktury (typ struktury) jest opcjonalna.
- Deklaracja struktury, po której nie występuje lista zmiennych (identyfikatorów) nie rezerwuje pamięci (opisuje wzorzec struktury).
- **Przykład:**

```
struct point {  // nie rezerwuje pamięci
        int x;
        int y;
};
struct point pt; // definicja zmiennej pt będącej strukturą typu point
point pt; // w C++ (można używać samej nazwy typu bez słowa struct)
```

- odwołania do poszczególnych składowych danej struktury realizuje się poprzez:

nazwa_struktury.nazwa_składowej

np.

```
pt.x = 320;
pt.y = 200;
printf("%d \t %d", pt.x, pt.y); // wypisanie składowych

struct point tab[50]; // definicja tablicy struktur typu point
```

Funkcje

- Pozwalają podzielić dowolny skomplikowany problem na mniejsze jednostki, które łatwiej napisać niż całość.
- Zamykają w sobie pewien fragment rozwiązania, który może być później wykorzystywany do innych problemów (np. funkcja obliczająca maksimum z kilku liczb w tablicy).
- Dzięki podzieleniu zadania na mniejsze jednostki, łatwiej stwierdzić w przypadku błędów co dokładnie nie działa.
- Funkcje w C dzielimy na obliczające wartości oraz nie obliczające wartości.
- Niedozwolone jest:
  - Wywoływanie niezadeklarowanych funkcji.
  - Tworzenie funkcji zagnieżdżonych (funkcji wewnątrz innych funkcji).
- **Definicja funkcji:**

  typ *identyfikator_funkcji* (*lista_argumentów_formalnych*){ blok }
  - ***typ*: liczbowy, wskaźnikowy, referencyjny, strukturowy**
  - ***lok* : ciąg deklaracji, definicji i instrukcji wykonywany :**
    - **do końca bloku**
    - **do instrukcji return**
  - ***argument_formalny:* typ *identyfikator***
- **Przykład:**

```
float sqr( float x ){
  return x * x ;
}
```

- W języku C argumenty przekazywane są przez wartość.
- Nie ma bezpośredniego sposobu na to, aby wywołana funkcja mogła zmienić wartość zmiennej należącej do funkcji wywołującej.
- Jedyny sposób na osiągnięcie celu polega na przekazaniu do funkcji wskaźników do obiektów, których wartości chcemy zmodyfikować.
- Przykład:

```
void swap(int *x, int *y){  // zamień miejscami *x i *y
    int tmp = *x;
    *x = *y;
    *y = tmp;
}
```