Podstawy Programowania
Zajęcia laboratoryjne 10
I rok Bioinformatyki Politechniki Poznańskiej

Struktury

1 Struktury

Struktury są elastycznym sposobem reprezentowania danych. Są obiektami złożonymi z jednej lub kilku zmiennych, które mogą być różnego typu (co wyróżnia je od tablic). Innymi słowy, struktura jest zestawem pól (nazywane składowymi struktury), gdzie pole może być daną lub strukturą danych. Strukturę można zdefiniować na kilka różnych sposobów:

1 przykład definicji struktury:

```
1  struct Struktura
2  {
3      int pole_1;
4      int pole_2;
5      char pole_3;
6  }; //konieczny jest tutaj srednik!
```

Deklaracja zmiennej typu strukturowego, czyli zmiennej będącej strukturą typu *Struktura* (dla powyższego przykładu), uzyskuje się poprzez podanie słowa kluczowego *struct*, nazwy struktury (w tym przypadku *Struktura*) i podania nazwy zmiennej (w tym przypadku *zmienna*):

```
1  struct Struktura zmienna;
```

Do utworzonej zmiennej typu strukturowego można bezpośrednio przypisać wartości do pól tej struktury:

```
1  struct Struktura zmienna = {1, 2, 'c'};
```

Odwołania do pól (składowych) struktury wykonuje się poprzez podanie nazwy struktury, a następnie po operatorze jakim jest kropka podajemy nazwę składowej do której się odwołujemy, przykładowo:

```
1  zmienna.pole_1 = 1; //odwolanie poprzez operator kropki "."
2  printf("%d", zmienna.pole_1);
```

W przypadku 1 przykładowej definicji wymagane jest utworzenie zmiennej typu Struktura w main() jeśli chcemy odwołać się do jej pól (struct Structura zmienna;), zwróć uwagę w jaki sposób wykonane jest odwołanie do składowych struktury. Przykład poniżej (kod 1 w materiałach na stronie):

```
1  #include<stdio.h>
2
3  struct trojkat
4  {
5      int a,b,c;
6  };
7
8  int main()
9  {
10     struct trojkat zmienna;
```

```
11      zmienna.a = 1;
12      zmienna.b = 2;
13      zmienna.c = 2;
14
15      printf("result: %d ", zmienna.a + zmienna.b + zmienna.c);
16
17      return 0;
18  }
```

2 przykład definicji struktury:

```
1  struct Struktura
2  {
3      int pole_1;
4      int pole_2;
5      char pole_3;
6  }zmienna; //konieczny jest tutaj srednik!
```

W przypadku powyższej definicji od razu utworzono zmienną typu Struktura: zmienna;. Przykład pokazano poniżej (kod 2 w materiałach na stronie):

```
1  #include<stdio.h>
2
3  struct trojkat
4  {
5      int a,b,c;
6  }zmienna;
7
8
9  int main()
10 {
11     zmienna.a = 1;
12     zmienna.b = 2;
13     zmienna.c = 2;
14
15     printf("result: %d ", zmienna.a + zmienna.b + zmienna.c);
16
17 return 0;
18 }
```

3 przykład definicji struktury:

```
1  typedef struct Struktura
2  {
3      int pole_1;
4      int pole_2;
5      char pole_3;
6  }zmienna; //konieczny jest tutaj srednik!
```

W przypadku powyższej definicji wykorzystano słowo kluczowe *typedef*, które tworzy nazwę zastępczą (alias) dla struktury. Przykład pokazano poniżej (kod 3 w materiałach na stronie):

```
1  #include<stdio.h>
2
3  typedef struct trojkat
4  {
5      int a,b,c;
6  }figura;
7
```

```
8  main()
9  {
10     figura zmienna;
11     zmienna.a = 1;
12     zmienna.b = 2;
13     zmienna.c = 2;
14
15     printf("result: %d ", zmienna.a + zmienna.b + zmienna.c);
16
17     return 0;
18 }
```

Definicja tablicy struktur, możliwe jest utworzenie zmiennej strukturowej, która będzie tablicą, przykładowo:

```
1  struct Struktura tab[10];
```

Dodatkowo, struktury mogą być zagnieżdżone, poniżej przykład składni:

```
1  struct dane_osobowe{
2      ...
3  };
4
5  struct baza_danych{
6      struct dane_osobowe osoba[5];
7      ...
8  };
```

Wskaźniki i funkcje, poniższe kody zostały podzielone zgodnie z trzema różnymi definicjami struktur (opisanymi powyżej).

1 przykład z deklaracją zmiennej typu strukturowego w głównej funkcji - main(), (kod 4 w materiałach na stronie):

```
1  #include<stdio.h>
2
3  struct trojkat
4  {
5      int a,b,c;
6  };
7
8  int obwod(struct trojkat zmienna);
9
10 int main()
11 {
12     struct trojkat zmienna;
13     zmienna.a = 1;
14     zmienna.b = 2;
15     zmienna.c = 2;
16
17     printf("result: %d", obwod(zmienna));
18
19     return 0;
20 }
21
22 int obwod(struct trojkat zmienna)
23 {
24     return zmienna.a + zmienna.b + zmienna.c;
25 }
```

2 przykład z deklaracją zmiennej typu strukturowego od razu przy definicji struktury, (kod 5 w materiałach na stronie):

```c
#include<stdio.h>

struct trojkat
{
    int a,b,c;
}zmienna;

int obwod(struct trojkat zmienna);

int main()
{
    zmienna.a = 1;
    zmienna.b = 2;
    zmienna.c = 2;

    printf("result: %d", obwod(zmienna));

    return 0;
}

int obwod(struct trojkat zmienna)
{
    return zmienna.a + zmienna.b + zmienna.c;
}
```

Jak można zauważyć nie ma znaczenia jaki wariant definicji struktury wybierzemy. W obu przypadkach do funkcji jako argument jest przekazywania zmienna typu strukturowego. Innym wariantem jest przekazanie do funkcji wskaźnika na strukturę.

Wskaźniki do struktur, jeśli chcemy przekazać strukturę do funkcji i ją tam zmodyfikować, musimy użyć wskaźnika (różnicą jest operowanie na oryginalnych wartościach, a operowanie na lokalnych kopiach).

3 przykład z przekazaniem wskaźnika na strukturę przy pomocy aliasu (kod 6 w materiałach na stronie):

```c
#include<stdio.h>

typedef struct trojkat
{
    int a,b,c;
}figura;

int obwod(figura *zmienna);

int main()
{
    figura zmienna;
    zmienna.a = 1;
    zmienna.b = 2;
    zmienna.c = 2;

    printf("result: %d", obwod(&zmienna));

    return 0;
}

int obwod(figura *zmienna)
{
    return (*zmienna).a + (*zmienna).b + (*zmienna).c;
}
```

Przykład prostej struktury, (kod 7 w materiałach na stronie):

```c
#include<stdio.h>

struct dane_osobowe{
    char imie[10];
    int wiek;

}osoba;

int main()
{
    printf("\nimie: ");
    scanf("%s", osoba.imie);
    printf("\nwiek: ");
    scanf("%d", &osoba.wiek);

    printf("\npodane imie to: %s", osoba.imie);
    printf("\npodany wiek to: %d", osoba.wiek);

    return 0;
}
```

Przykład struktury z alokacją pamięci: zwróć uwagę w jaki sposób możemy odwołać się do pola struktury wskazywanej przez wskaźnik:

```c
(*osoba).wiek //odwolanie do konkretnego pola struktury wskazywanej przez wskaznik
osoba->wiek //inny zapis odwolania do pola struktury wskazywanej przez wskaznik
```

Przykład struktury z alokacją pamięci (kod 8 w materiałach na stronie):

```c
#include<stdio.h>
#include<stdlib.h>

typedef struct {
    char *imie;
    int wiek;
}dane;

int main()
{
    dane *osoba;
    osoba = (dane*)malloc(sizeof(dane));

    osoba -> imie = "Janek";
    osoba -> wiek = 20;
    printf("%s ma %d lat\n", osoba->imie, osoba->wiek);

    free(osoba);
    printf("\nzwolnienie pamieci:\n%s ma %d lat\n", (*osoba).imie, (*osoba).wiek);

    return 0;
}
```

2 Zadania

Zad 1. Zdefiniuj strukturę przechowującą współrzędne punktu, czyli zmienną x i y. Przypisz polom struktury wartości i je wyświetl.

Zad 2. Zdefiniuj strukturę prostokąt przechowującą długości boków a i b. Napisz funkcję, która otrzymuje jako argument zmienną typu struct prostokąt i zwraca jako wartość pole prostokąta przekazanego w argumencie.

Zad 3. Napisz funkcję, która otrzymuje jako argumenty zmienną trojkat_1 (typu struct trojkat) oraz zmienną trojkat_2 (wskaźnik na zmienną typu struct trojkat). Funkcja ta przepisuje do zmiennej wskazanej przez trojkat_2 zawartość zmiennej trojkat_1.

3 Zadanie domowe

Poniższe zadania dotyczą rozszerzenia do programu przeliczania temperatur:

- Zamień 4 globalne tablice (*float t1[20], float t2[20], char z1[20], char z2[20]*) na strukturę *przeliczenie*, która zawiera 4 zmienne (*float t1, float t2, char z1, char z2*) i tablicę o rozmiarze 20, której elementami będą struktury *przeliczenie*. Opisana zmiana umożliwi wykorzystanie tylko jednej tablicy struktur zamiast 4 tablic globalnych.
- Po utworzeniu struktury odpowiednio zmodyfikuj cały kod, czyli dostosuj funkcje, które tego wymagają (niektóre z nich nie będą wymagały zmian).