Laboratorium informatyki

Programowanie w języku C

Ćwiczenie 12

1. Wstęp

Dostępne w języku C typy danych umożliwiają przechowywanie pojedynczej wartości liczbowej, logicznej lub znakowej. Za pomocą typu tablicowego można przechowywać pod jedną nazwą zmiennej określoną liczbę wartości pojedynczego typu. Możliwe jest także zdefiniowanie złożonego typu danych umożliwiającego przechowywanie złożonych struktur danych różnych typów.

2. Struktura

Deklarując strukturę można zdefiniować własny typ danych oparty o dostępne lub zdefiniowane typy danych. Struktura umożliwia pod jedną nazwą zgrupować dowolną liczbę zmiennych różnych typów. Elementami struktury mogą być zmienne, tablice, struktury i unie. Na podstawie zdefiniowanego typu strukturalnego można w kodzie programu zadeklarować zmienne i tablice strukturalne.

Deklarację struktury można umieścić w kodzie funkcji, ale zmienną strukturalną na jej podstawie będzie można utworzyć tylko w danej funkcji. Korzystniejsze jest definiowanie jej poza kodem funkcji po deklaracjach include, w takim przypadku zmienną strukturalną można utworzyć w dowolnym miejscu kodu programu.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct punkt
{
    int x,y;
    char nazwa[30];
}A = {10,10,"Alfa"}, D = {5,5,"Theta"};
int main()
{
    struct punkt B = {20,20,"Beta"};
    struct punkt C;
    C.x = 30;
    C.y = 30;
    strcpy(C.nazwa,"Gamma");
    printf("%i, %i, %s\n", A.x,A.y,A.nazwa);
    printf("%i, %i, %s\n", B.x,B.y,B.nazwa);
    printf("%i, %i, %s\n", C.x,C.y,C.nazwa);
    printf("%i, %i, %s\n", D.x,D.y,D.nazwa);
}
```

```
C:\Windows\system32\cmd.exe
10, 10, Alfa
20, 20, Beta
30, 30, Gamma
5, 5, Theta

Press any key to continue . . .
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct punkt
{
    int x,y;
    char nazwa[30];
};
int main()
{
    srand(time(0));
    struct punkt P[4];
    for (int i=0;i<4;i++)
    {
        P[i].x = rand()%21;
        P[i].y = rand()%21;
        char liczba[10] = {'\0'};
        char napis[30] = "Punkt_";
        itoa(i,liczba,10);
        strcat(napis,liczba);
        strcpy(P[i].nazwa,napis);
        printf("%i, %i, %s\n", P[i].x,P[i].y,P[i].nazwa);
    }
}
```

```
C:\Windows\system32\cmd.exe
4, 12, Punkt_0
17, 13, Punkt_1
11, 19, Punkt_2
16, 6, Punkt_3

Press any key to continue . . .
```

3. Unia

Unia jest także złożonym typem danych możliwym do zdefiniowania w programie. Podobnie jak struktura budowana jest w oparciu o dostępne typy danych. W odróżnieniu od struktury w unii można korzystać w danej chwili tylko z jednego z zdefiniowanych w niej elementów. Przy tworzeniu zmiennej typu unia w pamięci rezerwowany jest obszar umożliwiający przechowanie elementu o największym rozmiarze.

Unia nie zapewnia automatycznej kontroli aktywnego typu, podczas pisanie kodu konieczne jest samodzielne pilnowanie aktywnego elementu.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
union test
{
    int x;
    float y;
    char z[10];
};
void show(union test B)
{
    printf("B.x = %i\n",B.x);
    printf("B.y = %f\n",B.y);
    printf("B.z = %s\n",B.z);
}
int main()
{
    union test A;
    A.x = 20;
    show(A);
    A.y = 123.98;
    show(A);
    strcpy(A.z,"Test unia");
    show(A);
}
```

```
C:\Windows\system32\cmd.exe
B.x = 20
B.y = 0.000000
B.z = ¶
B.x = 1123546563
B.y = 123.980003
B.z = ├Ś B
B.x = 1953719636
B.y = 771352121069081500000000000000000.000000
B.z = Test unia

Press any key to continue . . .
```

4. Zadania

Wykorzystując poznane typy złożone napisać program zbierający dane o punktach nawigacyjnych wycieczki. Na podstawie wprowadzonych danych program wyświetla podsumowanie wycieczki wyliczając odległość pomiędzy kolejnymi punktami trasy oraz licząc długość przebytej drogi w każdym z punktów nawigacyjnych od pierwszego punktu wycieczki.

Lp.	Nazwa Punktu	Współrzędne	Odległość	Trasa
1	Alfa	0,0	-	-
2	Beta	10,100	100	100
3	Gamma	23,300	200	300
4	Delta	300,200	200	500
5	Theta	100,100	200	700