Podstawy programowania komputerów

Wykład 7: „Struktury i unie”

swalover@ie.tu.koszalin.pl

Struktury

Struktura jest zbiorem elementów różnych typów, traktowanym w C jako jeden obiekt. Każdy element struktury nazywany jest polem.

Deklaracja struktury:

```
struct <nazwa>
    {<typ pola> <nazwa pola>;
     <typ pola> <nazwa pola>;
         ...
    }<lista zmiennych>;
```

Nazwa pola jest zmienną lokalną.

Zmienne strukturalne

swalover@ie.tu.koszalin.pl

Inicjalizacja struktury

Struktury, podobnie jak tablice można inicjować w deklaracji:

```
struct complex {float re,float im};
struct complex l1 = {12.16, 45.72};
```

Można inicjować również tablice struktur:

```
struct complex liczby[ ] =
{{1.0,2.0},{1.2,3.6},{1.4,6.5}};
```

swalover@ie.tu.koszalin.pl

Zmienne typu strukturalnego

Zmienne typu strukturalnego można zadeklarować bezpośrednio:

```
struct{int dd;int mm;int rr;
      }data_ur,data_zatr;
```

i pośrednio:

```
struct data {int dd; int mm; int rr;};
. . .
data data_ur, data_zatr;
```

Zmienne strukturalne mogą być inicjowane wraz z deklaracją:

```
data d = { 12, 10, 2004};
```

swalover@ie.tu.koszalin.pl

Struktury zagnieżdżone

```
struct Osoba {char imie [20];
              char nazwisko [30];
              data data_ur,data_zatr;
              double wynagr;
             } prac1, prac2;
```

Zmienne `prac1, prac2` mają przydzielonych po 70 bajtów pamięci (20 na `imie`, 30 na `nazwisko`, 6 na `data_ur`, 6 na `data_zatr`, 8 na `wynagr`)

swalover@ie.tu.koszalin.pl

Dostęp do pól struktury

Uzyskuje się poprzez konstrukcję:

<nazwa_zmiennej_strukturalnej>.<nazwa_pola>

```
strcpy (prac1.imie,"Jan");
gets (prac2.nazwisko);
prac1.data_ur.dd = 15;
printf("Rok zatrudnienia: %d",
    prac1.data_zatr.rr);
```

swalover@ie.tu.koszalin.pl

Struktury a operacja przypisania

Jeżeli zmienne strukturalne są tego samego typu, dopuszcza się przekazanie „wartości” strukturze przez **przypisanie**:

```
data d1, d2, tmp;
. . .
tmp = d1;
d1 = d2;
d2 = tmp;
```

zamiana dat

swalover@ie.tu.koszalin.pl

Struktura argumentem funkcji

```
void Drukuj (Osoba osoba)
{ printf("Nazwisko:%s\n", osoba.nazwisko);
  printf("Imię: %s\n", osoba.imie);
  printf(„Urodzony %d.%d.%d”,
          osoba.data_ur.dd,
          osoba. data_ur.mm,
          osoba.data_ur.rr);
}    . . .
          Drukuj (prac1);
          Drukuj (prac2);
```

swalover@ie.tu.koszalin.pl

Struktura wartością zwracaną przez funkcję

```
Osoba Czytaj ( )
   {Osoba osoba;
    printf ("\nPodaj nazwisko: ");
    gets (osoba.nazwisko);
    printf ("\nPodaj imię: ");
    gets (osoba.imie);
        . . .
    return osoba; } . . .
       prac2 = Czytaj( );
```

swalover@ie.tu.koszalin.pl

Struktury a wskaźniki

Pola struktury mogą być wskaźnikami:

```
struct adres {char * ulica;
              char * miasto;
              char * kod_poczty; };
```

Dostęp do pól struktury poprzez wskaźnik:

```
Osoba *prac1_ptr;
prac1_ptr = &prac1;
*p1).wynagr=1200;
    // lub p1->wynagr=1200;
```

swalover@ie.tu.koszalin.pl

Unie

Unia jest zbiorem elementów zajmujących ten sam obszar w pamięci (nie równocześnie):

```
union rejestr { struct {
                    unsigned short AL;
                    unsigned short AH;
                          } A;
                 unsigned int AX; };
```

Unie wykorzystuje się tak samo jak struktury

```
union rejestr R;
R.A.AH = 9;
```

swalover@ie.tu.koszalin.pl

Właściwości unii

Każda unia przechowuje tylko jedną wartość.

Dzięki uniom możliwe jest utworzenie tablicy jednostek o jednakowej długości, z których każda może przechowywać dane innego typu.

Przykład szablonu unii:

```
union magazyn {
                int cyfra;
                double duzfl;
                char litera;
              }
```

swalover@ie.tu.koszalin.pl

Definicja i inicjalizacja unii

Tworzymy trzy „magazyny”:

```
union magazyn fit;
    //kompilator przydziela 8 bajtów
union magazyn tab[10];
    //tablica z 80   bajtów
union magazyn *wu;
    //tworzy się wskaźnik na unię
```

Inicjalizacja ⇨ definicja pierwszego składnika.

```
fit.cyfra = 7;
fit.litera = 's';
```

Należy pamiętać aktualny rodzaj danych w unii.

swalover@ie.tu.koszalin.pl

Wykorzystanie unii

Unia niezbędna gdy rodzaj przechowywanych danych zależy od jednego ze składników danych.

```
struct wlasciciel { char nr_dowodu [12]; ...};
struct firma      { char nazwa [40]; ...};

union dane        {struct wlasciciel wlasc_sam;
                   struct firma firma_sam; ...};
struc dane_sam    {char marka [15];
                   union dane dane_wlasc; ...};
```

swalover@ie.tu.koszalin.pl