

Typy złożone

Struktury, pola bitowe i unie.

- Typy całkowite:
 - char
 - short
 - int
 - long
- Typy zmiennopozycyjne
 - float
 - double
- Modyfikatory : unsigned, signed
- Typ wskaźnikowy

- Typ wyliczeniowy `enum`
- Tablice
- Struktury `struct`
- Unie `union`
- Pola bitowe
- Wskaźniki na funkcje i wskaźniki na typy złożone

DEKLARACJA

```
enum nazwa_typu
{
    element1,
    element2,
    ...
};
```

Przykład

```
enum kolory { czerwony, niebieski, zielony, czarny };

int main()
{
    enum kolory n;
    n = 0;
    if ( n == czerwony ) printf("To kolor czerwony\n");
}
```

- Typ wyliczeniowy enum to liczba całkowita (prawie to samo co int)
- Operacje arytmetyczne takie same jak na typie całkowitym

```
enum kolory n=zielony;  
n++;  
n=n+100;
```

- Poprawia czytelność kodu
- Może być zastąpiony przez odpowiednie dyrektywy, np.:

```
#define ZIELONY 1  
#define CZERWONY 2  
int kolor = ZIELONY;
```

```
1 #include <stdio.h>
2
3 enum kolory { zielony, niebieski, czerwony, czarny };
4
5 int main()
6 {
7     enum kolory n;
8
9     for(n=zielony; n<=czarny; n++)
10         printf("%d\n", n);
11
12     return 0;
13 }
```

0
1
2
3

 enum1.c

```

1  #include <stdio.h>
2
3  enum kolory { zielony=3, niebieski=7, czerwony, czarny };
4
5  int main()
6  {
7      enum kolory n;
8
9      printf("sizeof = %d\n", sizeof(n));
10
11     for(n=zielony; n<=czarny; n++)
12         printf("%d\n", n);
13
14     return 0;

```

```

sizeof = 4
3
4
5
6
7
8
9

```

enum2.c

- Polecenie typedef pozwala nadać własne nazwy dowolnym typom
- Deklaracja typu wygląda identycznie jak deklaracja zmiennej, należy tylko dodać słowo typedef na początku

```
typedef float liczba;
typedef float punkt[4];
typedef enum { false, true } bool;
typedef float *wskaznik;

int main()
{
    liczba x = 3.14;
    punkt y;
    bool czy_koniec = false;
    wskaznik w = &x;
}
```

- Polecenie typedef pozwala uniknąć powtarzania słowa struct lub enum

```
struct student
{
    char nazwisko[100];
    int indeks;
};

typedef struct student student_s;

void wypisz_student(student_s s)
{
    printf("Nazwisko: %s\n", s.nazwisko);
}
```

Operator -> umożliwia dostęp do pola struktury za pomocą wskaźnika.

wskaznik->pole jest równoważne z (*wskaznik).pole

```
1 #include <stdio.h>
2
3 typedef enum { M, K } plec;
4
5 typedef struct
6 {
7     char nazwisko[100];
8     int indeks;
9     plec p;
10 } student;
11
12 void wypisz_studenta(const student *s)
13 {
14     printf("Nazwisko: %s\n", s->nazwisko);
15     printf("Indeks   : %d\n", (*s).indeks);
16     printf("Plec     : %s\n", s->p == M ? "M" : "K");
17 }
18
19 int main()
20 {
21     student s = {"Zenon Adamski", 123456, M};
22     wypisz_studenta(&s);
23     return 0;
24 }
```

- Unie, podobnie jak struktury, przechowują w polach zmienne dowolnego typu
- W odróżnieniu od struktur, wszystkie zmienne umieszczone są pod tym samym adresem (przechowywana jest pojedyncza wartość).
- Modyfikacja jednego pola zmienia wartość pozostałych.

```
1 union ufloat
2 {
3     float f;
4     unsigned int i;
5     unsigned char c[4];
6 };
```

 union1.c

```
1 #include <stdio.h>
2
3 union liczba
4 {
5     float f;
6     unsigned int i;
7     unsigned char c;
8 };
9
10 int main()
11 {
12     union liczba x;
13
14     x.f = 3.14;
15
16     printf("sizeof = %d\n", sizeof(union liczba));
17
18     printf("adres pola f = %p\n", &x.f);
19     printf("adres pola i = %p\n", &x.i);
20     printf("adres pola c = %p\n", &x.c);
21
22     printf("wartosc pola f = %f\n", x.f);
23     printf("wartosc pola i = %x\n", x.i);
24     printf("wartosc pola c = %x\n", x.c);
25
26 }
```

- Pola bitowe to pola struktury o określonej liczbie bitów
- Pozwalają zoptymalizować zużycie pamięci w reprezentacji zmiennych całkowitych
- Wykorzystywane m. in. przy obsłudze urządzeń zewnętrznych przez porty

```
struct data
{
    unsigned int dzien    : 5 ;
    unsigned int miesiac  : 4 ;
    unsigned int rok      : 11 ;
};
```

```
1  #include<stdio.h>
2
3  typedef struct
4  {
5      unsigned int  dzien    :   5  ;
6      unsigned int  miesiac  :   4  ;
7      unsigned int  rok      :  11  ;
8  } data;
9
10 int main()
11 {
12     data dzis = { 24, 1, 2017 };
13
14     printf("Data: %u-%u-%u\n",
15           dzis.dzien, dzis.miesiac, dzis.rok);
16
17     printf("sizeof(data)=%ld\n", sizeof(data));
18
19     return 0;
```

Pola bitowe pozwalają poprawić czytelność kodu zawierającego operacje na bitach, jednak działają wolniej niż operatory bitowe.

```
typedef struct
{
    int prawo_odczytu: 1;
    int prawo_zapisu : 1;
    int prawo_wykonania : 1;
} atrybuty_pliku;

int main()
{
    atrybuty_pliku plik;

    if ( plik.prawo_odczytu )
    {
        /* ... */
    }
}
```

- deklarowanie własnych typów poleceniem `typedef`
- struktury, unie i pola bitowe
- typ wyliczeniowy `enum`