

pair i tuple

Objektno programiranje - 3. vježbe (1. dio)

dr. sc. Sebastijan Horvat

Prirodoslovno-matematički fakultet,
Sveučilište u Zagrebu

20. ožujka 2024. godine



- treba zaglavlje **utility**:

```
#include <utility>
```



- predložak (poput spremnika) iz kojeg generiramo određeni tip
- za držanje dva podatka - za svaki navodimo tip

Primjer 1.

```
pair<string, string> a;  
pair<string, size_t> b;  
pair<string, vector<int>> c;  
pair<string, string> d{"abc", "def"};  
cin >> d.first >> d.second;
```

- u posljednjem umjesto *defaultnih* vrijednosti (dva prazna stringa), inicijalizirali smo svaki član para
- direktan pristup elementima para: **first** i **second**

Funkcija za stvaranje para

make_pair(v1, v2)

- vraća par inicijaliziran s v1 i v2 - tip određen iz tipova od v1 i v2

Primjer 2. Funkcija koja vraća par:

```
pair<string, int> fja(vector<string> &v) {  
    if (!v.empty())  
        return {v.back(), v.back().size()};  
    else  
        return pair<string, int>();  
}
```

Pitanje. Što će se ispisati:

```
vector<string> vek{"ab", "cd", "efg"};  
auto p = fja(vek);  
cout << p.first << " " << p.second << endl;
```

Malo priče o & s prethodnog slajda

Primjer 3.

```
void zamjena(int &a, int &b) {
    int temp = a;
    a = b;
    b = temp;
}

int main(void) {
    int a = 10, b = 20;
    zamjena(a, b);
    cout << a << endl << b << endl;
    return 0;
}
```

Pitanje. Koja je razlika između prosljeđivanja po vrijednosti, po referenci i po pokazivaču?

Primjer 4. Što se ispiše?



```
pair<string,int> a = {"ab", 2},  
    b("ab",1), c = a, d(b);  
cout << (a < b) << endl  
    << (a != d) << endl  
    << (a == c) << endl;
```

Odgovor:

0
1
1

- potrebno zaglavlje `tuple`
- par imao točno dva člana - predložak `tuple` može imati proizvoljan (ali fiksni!) broj članova
- korisno ako želimo „nabrzinu“ zapakirati podatke u jedan objekt
- treba navesti tip i ime svakog člana
- ako ih želimo inicijalizirati, imati na umu da je konstruktor za `tuple` eksplicitan



Primjer 5.

```
tuple<size_t, size_t, size_t> triNule;  
tuple<string, vector<double>, int, list<int>>  
    a("bc", {3.14, 2.7}, 42, {0,1,2});  
tuple<size_t, size_t, size_t> triB{1,2,3};  
auto b = make_tuple("abc", 3, 20.00);
```

- tip od `b` je `tuple<const char*, int, double>`



tuple - pristupanje elementima

- koristimo predložak za funkciju `get`
- obavezni eksplicitni argument predloška je pozicija člana kojem pristupamo (brojimo od 0) - za `tuple` objekt vraća referencu na traženi član



Primjer 6.

```
tuple<string, int, double> t("ab",2,3.4);
get<1>(t) *= 5;
cout << get<0>(t) << endl
      << get<1>(t) << endl
      << get<2>(t) << endl;
```

- za usporedbu, moramo imati isti broj članova
- odgovarajući članovi moraju se moći usporediti



Primjer 7.

```
tuple<string, string> t1("1", "2");  
tuple<size_t, size_t> t2(1, 2);  
bool b = (t1 == t2);    X  
tuple<size_t, size_t, size_t> t3(1, 2, 3);  
b = (t2 < t3);    X  
tuple<size_t, size_t> t4(0, 0);  
b = (t4 < t2);    ✓
```


Zadatak 1.

Česta upotreba `tuple` - kad funkcije trebaju vratiti više vrijednosti.

Učitati riječi (do EOF) u vektor. Pomoću funkcije pronaći sve riječi iz unosa koje imaju barem dva slova 'a' i barem dva slova 'b', te ih sve vratiti u posebnom vektoru (zajedno s brojem pojava tih slova u njima). Nakon toga u glavnom programu ispisati sve te riječi i broj pojava 'a' i 'b' u njima.

Primjer.

Ulaz:	zid abba abeceda abrakadabra javor baraba
Izlaz:	abba, a: 2, b: 2 abrakadabra, a: 5, b: 2 baraba, a: 3, b: 2

Napomena. Funkcija `std::count(begin, end, val)` iz zaglavlja `algorithm` prima iteratore `[begin, end)` i vraća broj elemenata iz tog raspona koji su jednaki `val`.



Zadatak 2.

Napišite program koji od korisnika učitava uređene trojke cijelih brojeva (do EOF). Potrebno je spremiti učitane trojke u vektor te ga sortirati ulazno prema najvećoj vrijednosti u pojedinoj trojci. Primjerice, $(2, 3, 4)$ je u tom smislu manji od $(1, 6, 2)$ jer vrijedi $\max\{2, 3, 4\} < \max\{1, 6, 2\}$. Ispišite tako sortirane točke.

Primjer - ulaz:

```
1 2 3
2 4 1
6 2 5
1 6 9
5 5 5
3 8 1
```

Primjer - izlaz:

```
(1,2,3)
(2,4,1)
(5,5,5)
(6,2,5)
(3,8,1)
(1,6,9)
```

Uputa. U zaglavlju `algorithm` imamo funkcije `std::max` i `std::sort`.