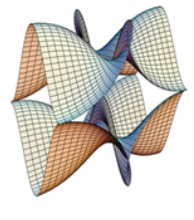

Prirodoslovno-matematički fakultet
Matematički odsjek
Sveučilište u Zagrebu

RAČUNARSKI PRAKTIKUM I

Vježbe 09 - this, static

v2018/2019.

Sastavio: Zvonimir Bujanović

Pokazivač this

Kako funkcija članica “zna” na kojem objektu djeluje?

```
class stack {
    int element[100], size;
    ...
    void push( int x ) { element[size++] = x; }
};

stack S, T;
S.push( 3 ); // radi sa S.element i S.size
T.push( 5 ); // radi sa T.element i T.size
```

Implicitno se funkciji zapravo šalje i pokazivač na objekt koji ju poziva.

```
// U "pozadini" se zapravo dogadja ovo (dakle, kao u SPA!):
void push( stack *S, int x ) { S->element[S->size++] = x; }

stack S, T;
push( &S, 3 ); push( &T, 5 );
```

Pokazivač this

Unutar svake funkcije članice dostupan je pokazivač `this`.
On sadrži adresu objekta koji je pozvao funkciju.

```
class stack
{
    ...
    void push( int x )
    {
        this->element[this->size] = x;
        ++this->size;
    }
};

stack S;
S.push( 3 ); // unutar push: this == &S
T.push( 5 ); // unutar push: this == &T
```

Pokazivač this

Pomoću `this` možemo vratiti referencu na objekt koji je pozvao funkciju. To je korisno za ulančavanje poziva.

```
class stack {
    ...
    stack &push( int x )
    {
        element[size++] = x;
        return *this;
    }
};

int main ()
{
    stack S;
    S.push(5).push(7).push(8); // OK!
    ...
}
```

Zbog čega ovo ne radi ako funkcija vraća samo `stack`?

Statičke varijable

Statičke varijable možemo deklarirati unutar funkcija i unutar klasa.

Statičke varijable unutar funkcija inicijaliziraju se samo prilikom prvog poziva. Njihova vrijednost se "pamti" i nakon izlaska iz funkcije.

```
void f1() {
    static int i = 0;
    cout << ++i;
}
void f2() {
    static int i = 0;
    cout << ++i;
}
...
f1(); // ispise 1
f1(); // ispise 2
f2(); // ispise 1
f2(); // ispise 2
```

Statičke varijable

Statičke varijable se kreiraju prilikom prvog korištenja, a uništavaju na kraju izvršavanja programa.

```
struct Test {
    Test() { cout << "Kreiran Test" << endl; }
    ~Test() { cout << "Unisten Test" << endl; }
};

void f() {
    static Test t;
}

int main() {
    cout << "Poceo program" << endl;
    f();
    cout << "Zavrsio program" << endl;
    return 0;
}
```

Statički članovi klasa

Statičkog člana klase `K` dijele sve varijable tipa `K`.

- Takvi članovi postoje i ako ne postoji niti varijabla tipa `K`.
- Takve članove moramo inicijalizirati izvan svih funkcija (na globalnom nivou). To radimo u `.cpp` datoteci (a ne u `.h`).

```
struct MyStruct {
    static int brojZivihStruktura;
    MyStruct() { ++brojZivihStruktura; }
    ~MyStruct() { --brojZivihStruktura; }
};

int MyStruct::brojZivihStruktura = 0; // inicijalizacija

int main() {
    MyStruct a, b, c;
    cout << MyStruct::brojZivihStruktura; // ispis: 3
    cout << a.brojZivihStruktura;         // ispis: 3
    return 0;
}
```

Zadatak 1

Nadopunite definiciju strukture s prethodnog slide-a tako da osim broja trenutno živih objekata tipa `MyStruct` ona održava i popis (listu) svih tih objekata.

Uoči: popis mora sadržavati pokazivače ili reference. Što bi se dogodilo da se u popisu nalaze samo `MyStruct`-ovi?

Statičke funkcije članice klase

Statičke funkcije članice mogu pristupati samo statičkim članovima klase. Unutar takvih funkcija ne postoji pokazivač **this**.

```
struct MyStruct {
    static int brojZivihStruktura;
    int brojac;
    MyStruct() { brojac = ++brojZivihStruktura; }

    static int broj() {
        return brojac; //  greska, brojac nije static
        return brojZivihStruktura; //  OK, ovo je static
    }
};

int main() {
    MyStruct test;
    int x = MyStruct::broj(); // OK, x = 1
    int y = test.broj();      // OK, y = 1
}
```

Pretpostavljeni parametri funkcija

Za sve ili za nekoliko zadnjih parametara funkcija možemo postaviti *defaultne* vrijednosti.

```
void f( int a, string b = "netko" )
{
    cout << b << " ima " << a << "godina";
}

int main()
{
    f(20, "Mirko"); // ispis: Mirko ima 20 godina
    f(19);          // ispis: netko ima 19 godina

    return 0;
}
```

Pretpostavljeni parametri funkcija

Treba biti oprezan jer može doći do dvosmislenosti!

```
struct MyStruct
{
    MyStruct() { ... }
    MyStruct( int a = 5 ) { ... }
};

MyStruct a;       // compile error! Koji konstruktor pozvati?
MyStruct b( 7 ); // OK! Jasno je koji konstruktor pozvati.
```