Napredno programiranje i programski jezici

02 C++

Fakultet tehničkih nauka, Novi Sad
23-24/Z
Dunja Vrbaški

FUNKCIJE

```
tipPovratneVrednosti nazivFunkcije ( param1, param2, ...) { teloFunkcije }
```

```
void fun(int x)
{
    x++;
}
void funPok(int *x)
{
    (*x)++;
}
void funRef(int &x)
{
    x++;
}
int main()
{
    int x = 3;
    fun(x);     cout << x;
    funPok(&x); cout << x;
    funRef(x);  cout << x;
    return 0;
}
```

```
void inc(int x)
{
    x++;
}
void print(int x)
{
    cout << "x = " << x << endl;
}

int main()
{
    int x = 3;
    inc(x);
    print(x);
}
```

```
void inc(int &x)
{
    x++;
}
void print(int &x)
{
    cout << "x = " << x << endl;
}

int main()
{
    int x = 3;
    inc(x);
    print(x);
}
```

Šta će biti ispisano?

```
void inc(int &x)
{
    x++;
}
void print(int &x)
{
    cout << "x = " << x << endl;
}

int main()
{
    int x = 3;
    inc(x);
    print(x);
}
```

Algoritam, zadatak, druge funkcije je takav da ne menja prosleđeni podatak.

Nekad želimo da budemo sigurni da će tako biti.

```
void inc(int &x)
{
    x++;
}
void print(int const &x)
{
    cout << "x = " << x << endl;
}

int main()
{
    int x = 3;
    inc(x);
    print(x);
}
```

Zaštitni mehanizam

Kad prosleđujemo po referenci - podatak nam je dostupan za izmenu.

Kad su podaci veliki

- želimo da prosledimo po referenci, da bismo izbegli nepotrebno kopiranje ALI
- želimo i da zaštitimo podatak od menjanja

```
void inc(int const &x)
{
    x++;
}

void print(int const &x)
{
    cout << "x = " << x << endl;
}

int main()
{
    int x = 3;
    inc(x);
    print(x);
}
```

```
error: increment of read-only reference 'x'
```

```
void inc(int &x)
{
    x++;
}

void print(int const &x)
{
    cout << "x = " << x << endl;
    inc(x);
}

int main()
{
    int x = 3;
    inc(x);
    print(x);
}
```

error: ...

```
const int globProm = 5;

int main()
{
    const int x = 3;

    ...
}
```

- koristi se i na drugim mestima
- *const correctness*
- posebno značajno u većim sistemima
- “sigurnosni” mehanizam

- const se koristi i u drugim jezicima.
- obratiti pažnju na razlike

```
#define MAX 100

const int globProm = 5;

int main()
{
    const int x = 3;
    int duzinaNiza = MAX;
    ...
}
```

#define - pretprocesorska direktiva
const - ključna reč

oblast važenja, vidljivost
MAX se može redefinisati (#undef)

```
void fun(int &);

void fun(int &x)
{
    x++;
}
```

referenca kao parametar funkcije

```
int main()
{
    int x = 3;
    int &y = x;
    y = 5;
}
```

referenca

```
int main()
{
    int x = 3;
    int &y;
}
```

```
error: 'y' declared as reference but not
initialized
```

```
int main()
{
    int x = 3;
    int &y = x;
    ...
}
```

```
int main()
{
    int x = 3;
    int &y = x;
    cout << x << y << endl;

    y++;
    cout << x << y << endl;
 }
```

Šta će biti ispisano?

```
int main()
{
    int x = 3;
    int &y = x;
    cout << x << y << endl;

    y++;
    cout << x << y << endl;
 }
```

Sve promene su nad istim podatkom

```
33
44
```

```
int main()
{
    int x = 3;
    int &y = x;
    cout << x << y << endl;

    y++;
    cout << x << y << endl;

    int z = 5;
    y = z;
    cout << x << y << endl;
}
```

Šta će biti ispisano?

```
int main()
{
    int x = 3;
    int &y = x;
    cout << x << y << endl;

    y++;
    cout << x << y << endl;

    int z = 5;
    y = z;
    cout << x << y << endl;
}
```

Neće y postati referenca na z, ona ostaje čvrsto vezana za podatak.
(nije rešenje 4, 5, 5)

x će preko reference y dobiti vrednost z.

```
33
44
55
```

```
int main()
{
    int x = 3;
    int &y = x;
    ...

    int  *pok1 = &y;
    int& *pok2 = y;

    int* &pok3 = pok1;

    int& niz[5];
}
```

- nisu podaci
- moramo pridružiti podatak
- alijas (koncept)
- ne možemo imati pokazivač na referencu (pokazivač čiji je tip vrednosti referenca)
- možemo imati referencu na pokazivač (drugo ime)
- ne možemo imati niz deklarisan kao niz referenci

Povezana tema: lvalue i rvalue podaci.

OOP