Потсетување за поимот обременување на функции

Обременета (overloaded-оптоварена, преклопена) функција се добива кога во иста програма се дефинираат неколку функции со исто име, но различни листи на аргументи и се декларирани во исто подрачје на важење.
Предноста на преоптоварувањето на функциите е тоа што не треба да се користат различни имиња за истата акција,а со тоа се зголемува прегледноста на програмскиот код.

Кога во иста област на важење има повеќе преклопени функции, за функциите со исто име освен првата, преведувачот ја толкува декларацијата на следниов начин:

1. Ако листите на аргументи се разликуваат по бројот или по типот, функциите се преклопени.
Пр.

```
void print (const, double);
viod print (int);
```

2. Ако типот на функциите и листите на аргументи се идентични, втората декларација ја смета за повторна декларација на функцијата.

3. Ако листите на аргументи се идентични, а типот на функциите се различни, втората декларација ја смета за погрешна декларација на функцијата и се јавува грешка.
Пр.

```
int maks(int m, int &n);
unsigned int maks(int, int &);
```

2. Ако листите на аргументи се разликуваат само по имињата на аргументите, втората декларација се смета за повторна декларација на функцијата.
Пр.

```
int maks(int m, int n);
void maks(int , int n=10);
```

Пр.

```
#include <iostream>
using namespace std;

void print(int i)
{
  cout << " ovde e int " << i << endl;
}
void print(double  f)
 {
  cout << " ovde e float " << f << endl;
}
void print(char const *c)
{
  cout << " ovde e char* " << c << endl;
}

int main() {
  print(10);
  print(10.10);
```

```
  print("deset");
  return 0;
}
```

„Разрешување на преклопени функции“ е постапка со која повикот на функцијата се насочува кон една од преклопените функции, односно од сите функции со исто име се избира најсоодветната.

Пр.

```
#include <iostream>

using namespace std;
void f();

void f(int);

void f(double, double=12.34);

void f(int, int);

int main()
{
  f();
  f(56);
    f(56, 89);
    f(56.78);
  cout<<endl;

   return 0;
}
void f()
{
   cout<<" f() se izbira funkcija bez argumenti"<<endl;
}
void f(int a)
{
   cout<<" f(56) se izbira funkcija so eden argument"<<endl;
}
void f(int a, int b)
{
   cout<<" f(56, 89) se izbira funkcija so dva argumenti"<<a<<" i "<<b<<endl;
}
void f(double a, double b)
{
   cout<<" f(56.78) se izbira funkcija so dva double argumenti, vtoriot e podrazbirliv"<<a<<" i
"<<b<<endl;
}
```

1. ФУНКЦИИ КАНДИДАТИ

Се идентификуваат ф-циите кандидати според функцискиот повик и листата на аргументи во повикот. Овде ф-ции кандидати се f(), f(int), f(float, float=12.34), f(int, int). Повикот содржи листа со еден аргумент од типот float.

2. СООДВЕТНИ ФУНКЦИИ НА ПОВИКОТ:
   Тие се оние функции:

- Чија листа на аргументи (според бројот и типот) одговара на листата на аргументи на повикот
- Или е можна конверзија на типовите на соодветните аргументи
- Или има повеќе аргументи но секој додатен аргумент има подразбирлива вредност.

Во примеров можни се две функции: f(int) со која е можна конверзија на аргумент од тип float во тип int и f(float, float=12.34), која има повеќе аргументи но вториот има подразбирлива вредност.

3. НАЈСООДВЕТНА ФУНКЦИЈА НА ПОВИКОТ

Е функцијата за која потребните конверзии се најдобри во однос на конверзиите на другите функции. Во примерот, за втората функција не е потребна конверзија на аргументот, затоа за најсоодветна се избира f(float, float=12.34).

Пр. наоѓање најголем од три цели броја, три реални броја, три знаци и три зборови со користење на функцииза преклопување

```
#include <iostream>
#include <string>
using namespace std;
int najgolem (int, int, int);
double najgolem(double, double, double);
char najgolem(char, char, char);
string najgolem(string, string, string);
int main()
{
   int a,b,c;
   double d,e,f;
   char g,h,i;
   string j,k,l;

   cout << "vnesi tri celi broja" << endl;
   cin>>a>>b>>c;
   cout<<"najgolem e "<<najgolem(a,b,c)<<endl;

    cout << "vnesi tri realni broja" << endl;
   cin>>d>>e>>f;
   cout<<"najgolem e "<<najgolem(d,e,f)<<endl;

    cout << "vnesi tri znaci" << endl;
   cin>>g>>h>>i;
   cout<<"najgolem e "<<najgolem(g,h,i)<<endl;
```

```
    cout << "vnesi tri zbora" << endl;
    cin>>j>>k>>l;
    cout<<"najgolem e "<<najgolem(j,k,l)<<endl;
    return 0;
}

int najgolem (int a1, int a2, int a3)
{
    int n=a1;
    if(a2>n) n=a2;
    if(a3>n) n=a3;
    return n;
}

double najgolem(double a1, double a2, double a3)
{
    float n=a1;
    if(a2>n) n=a2;
    if(a3>n) n=a3;
    return n;
}
char najgolem(char a1, char a2, char a3)
{
    char n=a1;
    if(a2>n) n=a2;
    if(a3>n) n=a3;
    return n;
}

string najgolem(string a1, string a2, string a3)
{
    string n=a1;
    if(a2>n) n=a2;
    if(a3>n) n=a3;
    return n;
}
```

Пр. подредување на три броја по големина

```
#include <iostream>
using namespace std;

void zameni (float &x, float &y);

int main()
{
  float a, b, c;

  cout << "vnesi tri broja " << endl;
```

```
  cin >> a >> b>>c;
  if(a>b)
          zameni (a,b);
if(a>c)
            zameni (a,c);
    if(b>c)
          zameni (b,c);
  cout << "Подредени броевите се : ";
  cout << a << " " << b <<" " <<c<< endl;
  return 0;
}
void zameni(int &x, int &y) // x i y se parametri
{
int t;
t=x; // vrednosta na x se stava vo privremena promenliva t
x=y; // vrednosta na y se stava vo promenlivata x
y=t; // vrednosta na t se stava vo promenlivata y
}
```

Пр. Напиши функција со која се проверува дали еден број се дели со друг број. Во главната програма да се одредат сите делители на дадениот природен број.

```
#include <iostream>
using namespace std;
int deliv(int m,int n)
{
   int t=0,b;
   t=m%n;
   if (t==0) b=0;
   else b=1;
   return b;
}
int main()
{
   int k,i,p;
   cout << "Vnesi go k" << endl;
   cin>>k;
   for(i=1;i<=k;i++)
   {
      p=deliv(k,i);
      if(p==0) cout<<i<<"   ";
   }
   return 0;
}
```

Пр. *Дадениот природен број да се разложи на прости множители со користење функција со која се проверува дали еден број се дели со друг број.

```
#include <iostream>
```

```
using namespace std;
int deliv(int m,int n)
{
   int t=0,b;
   t=m%n;
   if (t==0) b=0;
   else b=1;
   return b;
}
int main()
{
   int k,i=2,p;
   cout << "Vnesi go k" << endl;
   cin>>k;
  while(k!=1)
   {
      p=deliv(k,i);
      if(p==0)
      {
         cout<<i<<"   ";
         k=k/i;
   }
   else i++;
   }
   return 0;
}
```

пр. *Да се отпечатат сите прости броеви од 1 до n со користење на функцијата prost.

```
#include <iostream>
using namespace std;
bool prost(int n)
{
   int b=0,j;
   for(j=2;j<=n/2;j++)
   {
      if(n%j==0) b++;
   }
   if(b==0) return true;
   else return false;

}
int main()
{
   int n,i;
   bool t;
   cout << "vnesi go n" << endl;
   cin>>n;
   for(i=2;i<=n;i++)
   {
```

```
        t=prost(i);
        if (t) cout<<i<<" e prost  "<<endl;
    }
    return 0;
}
```

Пр. Напиши функција (од типот void) izmeni која како аргумени има референци на 3 цели броеви. Во процедурата, првиот аргумент се зголемува за 1, вториот аргумент се множи по 2, и третиот се намалува за 1. Потоа во главната програма да се декларираат 3 целобројни променливи x, y и z и да се внесат нивните вредности преку тастатура. Да се повика процедурата со нив како аргументи. На крајот во главната програма да се отпечатат нивните нови вредности.

Пример: x=3, y=2, z=4;
Излез: 4, 4, 3; (3+1=4, 2*2=4, 4-1=3)

```
#include <iostream>
using namespace std;
void izmeni(int &x, int &y, int &z)
{
   x++;
   y*=2;
   z--;

}
int main()
{
   int x,y,z;
   cout << "vnesi tri celi broja" << endl;
   cin>>x>>y>>z;
   izmeni(x,y,z);
   cout<<x<<"  "<<y<<"   "<<z;
   return 0;
}
```