Prof. Dr. Agni Dika
Material punues.
Nuk është për shumëzim!

04

Templejtët

Gjuha C++ mundëson definimin e *shablloneve të funkcioneve* (ang. function template) dhe *shablloneve të klasave* (ang. class template). Përmes këtyre dy shablloneve mundësohet krijimi i *funkcioneve të përgjithshme* (ang. generic functions) dhe *klasave të përgjithshme* (ang. generic classes). Në vijim, në vend të fjalës *shabllon* do të shfrytëzohet fjala *templejt*.

Templejte të funkcioneve

Në praktikë shpesh herë nevojitet që llogaritje të caktuara të kryhen mbi vlera të tipeve të ndryshme. Me qëllim të evitimit të krijimit të përsëritur të funkcioneve për veprim mbi të dhëna të tipeve të ndryshme, shfrytëzohen templejte funkcionesh.

Te templejtët, në vend të tipeve të vërteta për variablat dhe funkcionet shfrytëzohen tipe formale, të cilët deklarohen duke e shfrytëzuar fjalën **`class`**, ose fjalën **`typename`**.

Shembull

Shtypja e vlerës së variablës **`a`** përmes funksionit **`print`**, nëse variabla merret e tipit **`int`**, **`double`** dhe **`char`**.

a. Pa templejt

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

void print(int a)
{
      cout << a;
}

void print(double a)
{
      cout << a;
}

void print(char a)
{
      cout << a;
}

int main()
{
      cout << "\nVlerat q% shtypen";
      cout << "\n\nNum%r int .....: ";
```

```
        print(55);
        cout << "\n\nNum%r double ..: ";
        print(34.72);
        cout << "\n\nShkronj% ......: ";
        print('h');
        cout << "\n\n";
return 0;
}
```

Rez.

```
Vlerat që shtypen

Numër int .....: 55

Numër double ..: 34.72

Shkronjë ......: h
```

b. Me templejt

Në vend të 3 funkcioneve të veçanta definohet vetëm një templejt i funkcionit.

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

template <class T> void print(T a)
{
      cout << a;
}

int main()
{
      cout << "\nVlerat q% shtypen";
      cout << "\n\nNum%r int .....: ";
      print(55);
      cout << "\n\nNum%r double ..: ";
      print(34.72);
      cout << "\n\nShkronj% ......: ";
      print('h');
      cout << "\n\n";
return 0;
}
```

Këtu, fjala `class` lidhet me tipet e definuara nga shfrytëzuesi dhe nuk ka të bëjë me klasat. Kurse **a** quhet *argument i templejtit* (ang. template argument).

Definimi i funkcionit mund të bëhet edhe në fund të programit. Gjatë kësaj, në fillim të programit duhet të vendoset prototipi përkatës.

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

template <class T> void print(T a);     // Prototipi

int main()
{
     cout << "\nVlerat q% shtypen";
     cout << "\n\nNum%r int .....: ";
     print(55);
     cout << "\n\nNum%r double ..: ";
     print(34.72);
     cout << "\n\nShkronj% ......: ";
     print('h');
     cout << "\n\n";
return 0;
}

template <class T> void print(T a)
{
     cout << a;
}
```

Titulli mundet të shkruhet edhe i ndarë në dy rreshta:

```
template <class T>
void print(T a);
```

Në vend të identifikatorit **T** të tipit mund të shfrytëzohet çfardo identifikatori tjetër i zgjedhur lirisht. Kështu, p.sh., titulli i funksionit mund të shkruhet edhe duke e shfrytëzuar ndonjë identifikator tjetër:

```
template <class Tipi>
void print(Tipi a)
```

Gjithashtu, në vend të fjalës **class** mund të shfrytëzohet fjala **typename**, kështu:

```
template <typename Tipi>
void print(Tipi a)
```

kjo sepse fjalët **class** dhe **typename** gjatë definimit të templejteve mes veti janë sinonime.

Në vijim të këtij teksti do të përdoret fjala **typename** dhe shkronja **T** për tipin, të cilat zakonisht shfrytëzohen edhe në literaturë. (Më mirë në vend të class është të shfrytëzohet typename, ashtu që të mos na përzihet me klasat)!!!!!

Shembull

Templejti për gjetje të vlerës absolute.

```
// Programi template
#include <iostream>
using namespace std;
template <typename T> T absolute(T x)
{
     return (x<0) ? (-x) : (x);
}

int main()
{
     short a=-77,b=55;
     int c=-44;
     long d=-88;
     float e=-15;
     double f=-48;
     long double g=-26;
     cout << "\nVlerat absolute\n";
     cout << "\n  a: " << absolute(a);
     cout << "\n  b: " << absolute(b);
     cout << "\n  c: " << absolute(c);
     cout << "\n  d: " << absolute(d);
     cout << "\n  e: " << absolute(e);
     cout << "\n  f: " << absolute(f);
     cout << endl;
     return 0;
}
```

Kjo mund të realizohet edhe me një funkcion makro, p.sh. si në programin vijues.

```
// Programi makro
#include <iostream>
using namespace std;
#define absolute(x) (x<0) ? (-x) : (x)
int main()
{
     int a=-77,y;
     y=absolute(a);
     cout << "\nVlera absolute y=";
     cout << y << "\n\n";
     return 0;
}
```

Funkcionet me disa parametra

Nëse funkcionet përmbajnë disa parametra definimi i templejteve përkatës nuk dallon nga definimi i funkcioneve të zakonshëm me më shumë parametra.

Shembull

Funksioni për shtypje të sortuar sipas madhësisë të vlerave të dy variablave **x** dhe **y**, nëse ato janë të tipit **int**, **double** dhe **char**.

a. Pa templejt, si funkcione të mbingarkuara.

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

void alfa(int x,int y)
{
     if (x<y)
          cout << x << "  " << y;
     else
          cout << y << "  " << x;
}

void alfa(double x,double y)
{
     if (x<y)
          cout << x << "  " << y;
     else
          cout << y << "  " << x;
}

void alfa(char x,char y)
{
     if (x<y)
          cout << x << "  " << y;
     else
          cout << y << "  " << x;
}

int main()
{
     cout << "\nVlerat e sortuara";
     cout << "\n\nNumra int .....: ";
     alfa(77,5);
     cout << "\n\nNumra double ..: ";
     alfa(12.46,18.6);
     cout << "\n\nShkronja ......: ";
```

```
        alfa('d','a');
        cout << "\n\n";
return 0;
}
```

```
Vlerat e sortuara

Numra int .....: 5  77

Numra double ..: 12.46  18.6

Shkronja ......: a  d
```

b. Me templejt

Këtu merret vetëm një funkcion **alfa**. Gjatë definimit të funkcionit janë shfrytëzuar variablat **x** dhe **y** të tipit fiktiv **T**.

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

template <typename T>
void alfa(T x,T y)
{
     if (x<y)
          cout << x << "  " << y;
     else
          cout << y << "  " << x;
}

int main()
{
     cout << "\nVlerat e sortuara";
     cout << "\n\nNumra int .....: ";
     alfa(77,5);
     cout << "\n\nNumra double ..: ";
     alfa(12.46,18.6);
     cout << "\n\nShkronja ......: ";
     alfa('d','a');
     cout << "\n\n";
return 0;
}
```

Funkcioni është thirrur tre herë me tre parametra të tipeve të ndryshme.

Në rast të përgjithshëm, gjatë thirrjes, parametrat formal të funksionit templejt zëvendësohen ma parametra aktual të të gjitha tipeve të mundshme.

Ngjashëm me templejtet shfrytëzohen edhe makro-funkcionet, ku në fakt parametave nuk u shoqërohen tipet (sepse paraprocesori nuk i njehë tipet).

Shembull

Gjetja e vlerës më të vogël të variablave **x** dhe **y**, të cilat mund të jenë numra të plotë, numra dhjetorë dhe vlera të tipit karakter, duke e shfrytëzuar funksionin makro **alfa**.

```
# include <iostream>
using namespace std;
#define alfa(x,y) (x>y) ? (x) :(y)
int main()
{
     int y;
     double z;
     char v;
     cout << "\nVlerat e sortuara";
     y=alfa(77,5);
     cout << "\n\nNumra int .....: "
          << y;
     z=alfa(12.46,18.6);
     cout << "\n\nNumra double ..: "
          << z;
     v=alfa('d','a');
     cout << "\n\nShkronja ......: "
          << v;
     cout << "\n\n";
return 0;
}
```

Këtu, funkcioni i vetëm **alfa** është shfrytëzuar për gjetje të vlerës minimale të të dhënave të tipeve të ndryshme. Këyu, shtypja direkte pa i vendosur rezultatet në variablat ndihmëse nuk lejohet.

Shembull

Detyra paraprake e zgjidhur duke shfrytëzuar një templejt në të cilin përdoren variabla të tipit fiktiv **T**.

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

template <typename T>
T beta(T x,T y)
{
```

```
        if (x>y)
            return x;
        else
            return y;
}

int main()
{
        cout << "\nVlerat maksimale";
        cout << "\n\nNumri int .....: "
             << beta(77,5);
        cout << "\n\nNumri double ..: "
             << beta(12.46,18.6);
        cout << "\n\nShkronja ......: "
             << beta('d','a');
        cout << "\n\n";
return 0;
}
```

```
Vlerat maksimale

Numri int .....: 77

Numri double ..: 18.6

Shkronja ......: d
```

Parametrat e funkcionit mund të merren edhe si parametra referent, të tipit fiktiv **T**.

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

template <typename T>
T beta(T &x,T &y)
{
        if (x>y)
            return x;
        else
            return y;
}

int main()
{
        int a=10,b=30;
        cout << "\nVlerat maksimale";
        cout << "\n\nNumri int .....: "
             << beta(a,b);
        double c=12.46,d=18.6;
        cout << "\n\nNumri double ..: "
```

```
            << beta(c,d);
      char e='d',f='a';
      cout << "\n\nShkronja ......: "
            << beta(e,f);
      cout << "\n\n";
return 0;
}
```

Këtu, gjatë thirrjes së funkcionit nuk janë shënuar direkt vlerat sepse patjetër duhet të shfrytëzohen variablat referente, të variablave **x** dhe **y**, në të kundërtën kompjuteri do të na lajmëroj gabim.

Deklarimi i variablave të tipit fiktiv brenda funkcionit

Variablat fiktive përveç që deklarohen si parametra të funkcioneve mund të deklarohen edhe si variabla lokale brenda funkcionit.

Shembull

Funkcioni që e përdorëm më sipër, por te i cili si variabël e përkohshme shfrytëzohet një variabël e tipit fiktiv **T**.

```
template <typename T>
T beta(T x,T y)
{
      T z;
      if (x>y)
            z=x;
      else
            z=y;
      return z;
}
```

Funkcione me parametra të përzier

Në templejt, përveç parametrave fiktiv mund të shfrytëzohen edhe parametra të tipeve standarde.

Shembull

Llogaritja e vlerës së funkcionit vijues duke e shfrytëzuar templejtin **gama**, ashtu që variabla **x** të deklarohet si variabël e tipit fiktiv **T**, kurse variablën **n** si variabël e tipit **int**.

$$y = 2x + 4\sum_{i=1}^{n+1}(i + 3x)$$

Në program, templejti të shfrytëzohet dy herë, ashtu që variabla **x** të merret e tipi **int** dhe e tipit **double**.

```
#include <iostream>
using namespace std;
template <typename T>
double gama(T x,int n);

int main()
{
     int a=2;
     cout << "\nRezultati p%r x int: y="
          << gama(a,3);
     double b=2.75;
     cout << "\nRezultati p%r x double: y="
          << gama(b,3)
          << "\n\n";
return 0;
}

template <typename T>
double gama(T x,int n)
{
     T s=0;
     int i;
     for (i=1;i<=(n+1);i++)
          s=s+(i+3*x);
     return 2*x+4*s;
}
```

Këtu **n** dhe rezultati **y** janë marrë variabla të zakonshme. Vetëm **x** është menduar të jetë variabël e përgjithshme.

```
Rezultati për x int: y=140
Rezultati për x double: y=177.5
```

Parametra të përgjithshëm të ndryshëm

Templejti mund të ketë disa parametra fiktiv të tipeve të ndryshme.

Shembull

Shembulli paraprak por gjatë definimit të templejtit shfrytëzohen dy tipe fiktive të ndryshme të parametrave.

```
#include <iostream>
using namespace std;
template <typename T,typename Z>
double gama(T x,Z n);

int main()
{
     int a=2;
     cout << "\nRezultati p%r x int: y="
          << gama(a,3);
     double b=2.75;
     cout << "\nRezultati p%r x double: y="
          << gama(b,3)
          << "\n\n";
return 0;
}

template <typename T,typename Z>
double gama(T x,Z n)
{
     T s=0;
     int i;
     for (i=1;i<=(n+1);i++)
          s=s+(i+3*x);
     return 2*x+4*s;
}
```

Këtu, janë deklaruar dy tipe fiktive, **T** dhe **Z**.

Parametrat mund të jenë edhe variabla referente të tipeve fiktive të ndryshme.

Shembull

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

template <typename R,typename S>
void delta(R &a,S &b);

int main()
{
     cout << "\nRasti i par%: ";
     int x=7, y=12;
     delta(x,y);
     cout << "\n\nRasti i dyt%: ";
```

```
        double z=9.44;
        delta(x,z);
        cout << "\n\nRasti i tret%: ";
        char v='h';
        delta(z,v);
        cout << "\n\n";
return 0;
}

template <typename R,typename S>
void delta(R &a,S &b)
{
        cout << "\nVlera parë: "
             << a;
        cout << "\nVlera dytë: "
             << b;
}
```

Rez.

Edhe në këtë rast, në programin kryesor janë deklaruar variablat referente, sepse programi nuk funkcionon nëse gjatë thirrjes së funkcionit merren vlera konkrete të parametrave aktualë.

Fushat në templejt

Ngjashëm si te funkcionet e zakonshëm, edhe te templejtët si parametra fiktiv mund të merren fusha.

Shembull

Shumat e anëtarëve të vektorëve **A(n)** dhe **B(n)**, duke i marrë të tipeve **int** dhe **float**. Për definim të llogaritjes së shumës të shfrytëzohet templejti **shuma**.

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

```
template <typename T>
T shuma(T A[],int n)
{
     T s=0;
     int i;
     for (i=0;i<n;i++)
          s=s+A[i];
     return s;
}

int main()
{
     const int n=5;
     int A[n]={2,5,3,4,7};
     cout << "\nShumat e llogaritura";
     cout << "\n\nP%r vektorin int: "
          << shuma(A,n);
     const int m=7;
     double B[m]={3.5,2.7,4.9,7.2,6.3,8.4,1.5};
     cout << "\n\nP%r vektorin double: "
          << shuma(B,m)
          << "\n\n";
return 0;
}
```

```
Shumat e llogaritura

Për vektorin int: 21

Për vektorin double: 34.5
```

Njëllojë funksionon edhe versioni kur vektori shënohet në titullin e templejtit si pointer.

Shembull

```
#include <iostream>
using namespace std;
template <typename T>
T shuma(T *p,int n)
{
     T s=0;
     int i;
     for (i=0;i<n;i++)
          s=s+*(p+i);
     return s;
```

```
}

int main()
{
     const int n=5;
     int A[n]={2,5,3,4,7};
     cout << "\nShumat e llogaritura";
     cout << "\n\nP%r vektorin int: "
          << shuma(&A[0],n);
     const int m=7;
     double B[m]={3.5,2.7,4.9,7.2,6.3,8.4,1.5};

     cout << "\n\nP%r vektorin double: "
          << shuma(&B[0],m)
          << "\n\n";
return 0;
}
```

Mbingarkesa e funkcioneve templejt

Mund të shfrytëzohen funkcione templete me emra të njëjtë por që kryejnë operacione të ndryshme. Që të dallohen mund të shfrytëzohet numër i ndryshëm i parametrave.

Fillimisht të merret ndonjë shembull më i thjeshtë!

```
#include <iostream>
using namespace std;
template <typename T>
T shuma(T *R,int n)
{
     T s=0;
     int i;
     for (i=0;i<n;i++)
          s=s+R[i];
     return s;
}

template <typename T>
T shuma(T *R,int n,int k)
{
     T s=0;
     int i;
     for (i=k;i<n;i++)
          s=s+R[i];
     return s;
```

```
}
int main()
{
     const int n=5;
     int A[n]={2,5,3,4,7};
     cout << "\nShuma e an%tar%ve te A: "
          << shuma(A,n);
     const int m=7;
     double B[m]={3.5,2.7,4.9,7.2,6.3,8.4,1.5};
     cout << "\nShuma e an%tar%ve te B: "
          << shuma(B,m);  // Mundet edhe shuma(&B[0],m)
     cout << "\nShuma e an%tar%ve me te B: "
          << shuma(A,n,3)
          << "\n\n";
return 0;
}
```

Funkcione pa rezultat

Si templejt mund të definohen edhe funkcione të cilat si rezultat nuk japin ndonjë vlerë dalëse, por rezultatet që fitohen përmes tyre shtypen duke shfrytëzuar komanda brenda funkcioneve, gjë që u pa edhe në pjesët paraprake.

Shemull

Shembulli i dhënë në pjesën paraprake.

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

template <typename T>
void min(T &a,T &b);

int main()
{
     cout << "\nPrej numrave t% plot%: ";
     int x=7, y=4;
     min(x,y);
     cout << "\nPrej numrave dhjetor%: ";
     double g=6.3, h=9.44;
     min(g,h);
     cout << "\nPrej karaktereve: ";
     char z='u', v='c';
     min(z,v);
     cout << "\n\n";
return 0;
}
```

```
template <typename T>
void min(T &a,T &b)
{
     T y;
     if (a<b) y=a; else y=b;
     cout << y;
}
```

Kujdes, edhe ky shembull nuk funksionoi nëse vlerat shënohen si parametra aktual, sepse duhet patjetër të kemi të deklaruara variabla referente.

Rez.

```
Prej numrave të plotë: 4
Prej numrave dhjetorë: 6.3
Prej karaktereve: c
```

Shembull

Krahasimi i tre vlerave të tipeve të ndryshme.

a. Zgjidhja pa shfrytëzuar templejt. Duhet të shfrytëzohen 3 funkcione të veçanta, nga një për secilin tip të të dhënave që krahasohen.

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

void max(int x,int y,int z);
void max(double x,double y,double z);
void max(char x,char y,char z);

int main()
{
     cout << "Rezultatet e krahasimeve";
     cout << "\n\nVlera më e madhe int .....:   ";
     max(3,8,5);
     cout << "\nVlera më e madhe double ..: ";
     max(4.7,6.3,9.4);
     cout << "\nVlera më e madhe char ....:   ";
     char x='a',y='b',z='c';
     max(x,y,z);
     cout << "\n\n";
return 0;
}

void max(int x,int y,int z)
{
     if ((x<y) && (x<z))
```

```
                cout << x;
        else
                if (y<z)
                        cout << y;
                else
                        cout << z;
}
void max(double x,double y,double z)
{
        if ((x<y) && (x<z))
                cout << x;
        else
                if (y<z)
                        cout << y;
                else
                        cout << z;
}

void max(char x,char y,char z)
{
        if ((x<y) && (x<z))
                cout << x;
        else
                if (y<z)
                        cout << y;
                else
                        cout << z;
}
```

Këtu, 3 herë shkruhet funksioni i njëjtë për shak se krahasohen tipe të ndryshme vlerash.

b. Duke shfrytëzuar një templejt

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

template <typename T>
void max(T x,T y,T z);

int main()
{
        cout << "Rezultatet e krahasimeve";
        cout << "\n\nVlera më e madhe int .....:   ";
        max(3,8,5);
        cout << "\nVlera më e madhe double ..: ";
        max(4.7,6.3,9.4);
        cout << "\nVlera më e madhe char ....:   ";
```

```
        char x='a',y='b',z='c';
        max(x,y,z);
        cout << "\n\n";
    return 0;
    }

    template <typename T>
    void max(T x,T y,T z)
    {
        if ((x<y) && (x<z))
                cout << x;
        else
            if (y<z)
                cout << y;
            else
                cout << z;
    }
```

Përzierja me funkcione jotemplejt

Njëkohësisht mund të shfrytëzohen edhe funkcione që nuk ka të bëjë me funkcione templejt.

Shembull

Dy funkcione, prej të cilëve funkcioni i dytë është definuar si templejt.

```
# include <iostream>
using namespace std;
template <typename T>
void alfa(T k)
{
    cout << "Shtypja brenda templejtit: "
         << k
         << endl;
}

void alfa(int b)
{
    cout << "Shtypja jasht% templejtit: "
         << b
         << endl;
}

int main()
{
```

```
     alfa(15.4);
     alfa(77);      // Thirret funkcioni i cili nuk është templejt
     alfa('z');
return 0;
     }
```

Disa templejtë njëkohësisht

Shembull

Programi brenda të cilit përfshihen dy funkcione templejt.

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

template <typename T>
void print(T a)
{
     cout << a;
}

template <typename R>
void shtyp(R x)
{
     cout << x;
}

int main()
{
     cout << "\nVlerat q% shtypen:";
     cout << "\n\nNum%r int .....: ";
     print(55);
     cout << "\n\nNum%r double ..: ";
     print(34.72);
     cout << "\n\nShkronj% ......: ";
     print('h');
     cout << "\n\nMe programin shtyp ......: ";
    shtyp(35.3);
     cout << "\n\n";
return 0;
}
```

Këtu nuk bënë që deklarimet t'i vendosim kështu:

```
template <typename T>
template <typename R>
void print(T a)
{
```

```
        cout << a;
}
void shtyp(R x)
{
        cout << x;
}
```

Gjithashtu nuk bënë që të shfrytëzohet kjo formë e deklarimit:

```
template <typename T>
void print(T a)
{
        cout << a;
}
void shtyp(T x)
{
        cout << x;
}
```

Këtu, gjatë ekzekutimit kompjuteri do të na lajmëroj se te funkcioni i dytë tipi T është i padefinuar.

Funkcionet me rezultat

Shembuj të ngjashëm u dhanë edhe më parë.

Shembull

Gjetja e vlerës maksimale.

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

template <typename T>
T max(T x,T y,T z);

int main()
{
        cout << "Rezultatet e krahasimeve";
        cout << "\n\nVlera më e madhe int .....:   "
             << max(3,8,5);
        cout << "\nVlera më e madhe double ..: "
             << max(4.7,6.3,9.4);
        cout << "\nVlera më e madhe char ....:   ";
        char x='a',y='b',z='c';
        cout << max(x,y,z);
        cout << "\n\n";
```

```
return 0;
}

template <typename T>
T max(T x,T y,T z)
{
     if ((x<y) && (x<z)) return x;
     else
          if (y<z) return y;
               else
                    return z;
}
```

Detyrë tjetër ku kemi tip tjetër nga ai i parametrave.

```
// Programi Template1b
#include <iostream>
using namespace std;

template <typename T>
bool barazi(T x,T y);

int main()
{
     cout << "Rezultatet e krahasimeve";
     cout << "\n\nVlerat int janë ";
     if (barazi(3,8)==true)
          cout << "barazi";
     else
          cout << "jobarazi";
     cout << "\nVlerat double janë ";
     if (barazi(5.4,5.4)==true)
          cout << "barazi";
     else
          cout << "jobarazi";

     cout << "\nVlerat char janë ";
     char x='a',y='g';
     if (barazi(x,y)==true)
          cout << "barazi";
     else
          cout << "jobarazi";
     cout << "\n\n";
return 0;
}

template <typename T>
bool barazi(T x,T y)
{
```

```
     if (x==y)
          return true;
     else
          return false;
}
```

Funkcione për sortim si templejt

Shembull

Funkcioni për Selection sort si templejt.

```
// Selection sort
#include <iostream>
#include <iomanip>
#include <ctime>
using namespace std;

void Vektori(int A[],int b);

template <typename T>
void SelectionSort(T Z[],int n);

int const m=100;
int main()
{
  int B[m];
  int i;
  srand(time( NULL));
    Vektori(B, m );
    SelectionSort(B, m );
    for (i=0;i<m;i++)
        cout << setw(6) << B[i] << "   ";
    cout << endl;
return 0;
}

// Mbushja e vektorit me numra të rastit
void Vektori(int A[],int b )
{
  for (int i=0;i<b;i++)
      A[i]=rand();
}

template <typename T>
void SelectionSort(T Z[],int n)
{
    T x;
```

```
    int i,j;
    for(i=0;i<n;i++)
    {
        for(j=i+1;j<n;j++)
        {
            if(Z[i]>Z[j])
            {
                x=Z[i];
                Z[i]=Z[j];
                Z[j]=x;
            }
        }

    }
}
```

Shembull

Funkcioni për Insertion sort si templejt.

```
// Insertion sort
#include <iostream>
#include <iomanip>
#include <ctime>
using namespace std;

void Vektori(int A[],int b);

template <typename T>
void InsertionSort(T Z[],int n);

int const m=100;
int main()
{
  int B[m];
  int i;
  srand(time( NULL));
    Vektori(B,m);
    InsertionSort(B,m);
    for (i=0;i<m;i++)
        cout << setw(6) << B[i] << "    ";
    cout << endl;
return 0;
}

void Vektori(int A[],int b)
{
  for (int i=0;i<b;i++)
```

```
        A[i]=rand();
}

template <typename T>
void InsertionSort(T A[],int n)
{
     T v;
     for (int i=1;i<n;++i)
     {
          if (A[i]<A[i-1])
               {
                    v=A[i];
                    int j=i;
                    do
                         {
                              A[j]=A[j-1];
                              --j;
                         }
                    while ((j>0) && (v<A[j-1]));
                    A[j]=v;
               }
     }
}
```

Shembull

Funkcioni për Bubble sort si templejt.

```
// Bubble sort
#include <iostream>
#include <iomanip>
#include <ctime>
using namespace std;

void Vektori(int A[],int b);

template <typename T>
void BubbleSort(T Z[],int n);

int const m=100;
int main()
{
  int B[m];
  int i;
  srand(time( NULL));
    Vektori(B, m );
    BubbleSort(B, m );
    for (i=0;i<m;i++)
        cout << setw(6) << B[i] << "   ";
    cout << endl;
```

```
return 0;
}

void Vektori(int A[],int b )
{
  for (int i=0;i<b;i++)
      A[i]=rand();
}

template <typename T>
void BubbleSort(T Z[],int n)
{
    bool fundi=false;
    int i,j,x;
    for (i =(n - 1);i >=0; i--)
    {
      if (fundi)
      break;
      fundi=true;
      for (j=1;j<=i;j++)
      {
        if (Z[j-1]>Z[j])
        {
          fundi=false;
          x=Z[j-1];
          Z[j-1]=Z[j];
          Z[j]=x;
        }
      }
    }
}
```

Klasat si templejt

Gjatë definimit të kalsave mund të shfrytëzohen edhe templejtet.

Shembull

```
#include <iostream>
using namespace std;
template <typename T,int n>
class gama
{
public:
    gama()            // Konstruktori
    {
```

```
            cout << "Konstruktori n="
                 << n
                 << endl;
      }

      void vlera(T a)
      {
            cout << "Funkcioni: "
                 << a
                 << endl;
      }
};

int main()
{
// Deklarimi i objektit dita
      cout << "\nRezultatet e objektit dita:\n";
      gama <int,85> dita;
      dita.vlera(66);

// Deklarimi i objektit nata
      cout << "\nRezultatet e objektit nata:\n";
      gama <char,42> nata;
      nata.vlera('h');
     cout << endl;
return 0;
}
```

Rez.

```
Rezultatet e objektit dita:
Konstruktori n=85
Funkcioni: 66

Rezultatet e objektit nata:
Konstruktori n=42
Funkcioni: h
```