Laboratorium 1

Przeciążanie funkcji

W języku C++ można stosować tę samą nazwę dla funkcji o różnej treści, pod warunkiem, że funkcje te mają
różne parametry (różny jest typ lub liczba parametrów). Jest to przeciążanie nazwy funkcji.

Kompilator analizuje listę typów argumentów aktualnych i dobiera (o ile nie ma niejednoznaczności) odpowiednią wersję funkcji. Dzięki takiemu rozwiązaniu, użytkownik musi pamiętać tylko jedną nazwę i nią się posługiwać w programie. Wybór właściwej funkcji należy do kompilatora.

Przykład 1:

```
#include <iostream>
using namespace std;
// Obydwie funkcje mają nazwę srednia
// bez przeciążania musiałyby mieć inne nazwy

double srednia(double n1, double n2) {
  return ((n1 + n2)/2.0);
}

double srednia(double n1, double n2, double n3) {
  return ((n1 + n2 + n3)/3.0);
}

int main () {
  cout << "Srednia liczb 2.5 i 4.5 wynosi: ", srednia(2.5,4.5);
  cout << "Srednia liczb 2.5, 4.5 i 6.5 wynosi: ", srednia(2.5,4.5,6.5);
  cin.get();
  return 0;
}
```

Przykład 2:

```
#include <iostream>
using namespace std;
// Obydwie funkcje mają nazwę srednia
// bez przeciążania musiałyby mieć inne nazwy

double srednia(int t[],int ile) {
  int suma=0;
  for (int i=0;i<ile;i++)
  suma += t[i];
  return (double)suma/ile;
}
double srednia(double t[],int ile) {
  double suma=0;
  for (int i=0;i<ile;i++)
  suma += t[i];
   return suma/ile;
}

int main () {
  int t1[]={1,2,3}; // tablica liczb całkowitych
  double t2[]={4.,5.,6.}; // tablica liczb rzeczywistych
  cout << "calkowite: "<< srednia(t1,3) << endl;
```

```
  cout << "rzeczywiste: "<< srednia(t2,3) << endl;
  return 0;
}
```

O czym należy pamiętać podczas przeciążania funkcji?

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

double litry(double kilometry, double litry) {
// Oblicza ile samochod pali na 100 km
  return (litry/kilometry*100);
}

// int litry(int stan_baku_pocz, int stan_baku_kon) {
// Oblicza zuzycie bezyny
//  return (stan_baku_pocz – stan_baku_kon);
// }

int main () {
  cout << "Zuzycie benzyny: ", litry(100.,12.);
  cout << "Zuzycie benzyny: ", litry(100,8);

// Co będzie jeśli wprowadzimy przeciążoną funkcję litry?

}
```

Pytania sprawdzające:

1. Kiedy używać przeciążania funkcji?

2. Załóżmy, że mamy dwie definicje funkcji:

```
double wynik(double czas, double odleglosc);
int wynik(double punkty);
```

   Która definicja funkcji zostanie użyta w wywołaniu:

```
double x, y;
y=wynik(x);
```

3. Załóżmy, że mamy następujące definicje:

```
double f(double n1, double n2);
double f(double czas, int licznik);
double x,y;
```

   Która definicja będzie użyta w następujących wywołaniach funkcji:

```
a) x=f(y,3.0);
b) x=f(2,3);
c) x=f(2,3.0);
```

Argumenty domyślne

Argument domyślny to wartość, która jest używana automatycznie wtedy, kiedy pominiemy odpowiedni argument rzeczywisty w wywołaniu funkcji.

Przykład:

a) *z parametrami obowiązkowymi*: w prototypie (deklaracji) i definicji funkcji podaje się listę argumentów; dla każdego umieszczonego na liście argumentu należy podczas wywołania funkcji dostarczyć wartość zgodną z typem podanym w deklaracji

```
// Definicja funkcji
int suma(int a, int b, int c)
{
return a+b+c;
}
...
// Tylko jeden sposób wywołania funkcji
z=suma(1,3.0,6.0);// wartości przesłane w argumentach
```

b) *z parametrami domyślnymi*: w prototypie (deklaracji) i w definicji funkcji podaje się listę argumentów z przypisanymi im wartościami domyślnymi argumentów; w wywołaniu takiej funkcji można nie podawać argumentów, przy czym obowiązuje zasada:
   - jeśli w wywołaniu podany zostanie argument, jego wartość zostanie w funkcji wykorzystana
   - jeśli w wywołaniu nie podany zostanie argument, w funkcji zostanie przyjęta wartość domyślna tego argumentu

```
// Definicja funkcji – każdemu argumentowi przypisano
// wartość domyślną
int suma(int a=0, int b=0, int c=0)
{
return a+b+c;
}


// Różne wywołania funkcji
z=suma();         // wszystkie wartości domyślne,
                  // wynik z wynosi 0
z=suma(5);        // b i c wartości domyślne
                  // wynik z wynosi 5
z=suma(5,3);      // c wartość domyślna
// wynik z wynosi 8
z=suma(1,3,6);    // wszystkie wartości przesłane w argumentach
                  // wynik z wynosi 10
```

- Ograniczenia stosowania parametrów domyślnych: jeśli parametr nie ma wartości domyślnej, to nie może jej posiadać żaden z poprzedzających go parametrów

```
// poprawne
int objetosc(int dlugosc, int szerokosc=1, int wysokosc=1);
// NIEPOPRAWNE !!!
int objetosc(int dlugosc=1, int szerokosc=1, int wysokosc);
```

Zadanie 1

Opracować zestaw przeciążonych funkcji, które pozwalają

a) wczytać do tablicy jednowymiarowej całkowitej lub rzeczywistej zestaw liczb całkowitych lub rzeczywistych (funkcje mają nazwę Czytaj),
b) wydrukować zawartość odpowiedniej tablicy (funkcje mają nazwę Drukuj);
c) obliczyć sumę wszystkich elementów w tablicy (funkcje mają nazwę Suma).

Napisać program testujący opracowane funkcje.

Zadanie 2

Napisać program pozwalający scalać dwie tablice jednowymiarowe o różnych rozmiarach. Założenia:

Tablice przechowywane są w postaci struktury:

```
struct Tab

{
  unsigned n; // liczba elementów
  int *t; // elementy tablicy
};
```

W programie wykorzystywane są funkcje:

```
Tab NewTab(unsigned m, unsigned n); // tworzenie tablicy m x n,
// wypełnionej zerami
void wypelnij(Tab T, int x=0); // wypełnianie tablicy wartoscią x
void drukuj(Tab T, char* nazwa); // drukowanie tablicy
Tab scal(Tab T1, Tab T2); // scalanie tablic
void UsunTab(Tab T1); // usuwanie tablic
int max(int a, int b){ return (a>b)?a:b; } // funkcja pomocnicza:
// max z dwóch liczb
```

Zadania domowe

Zadanie 1.

Niech będą dane następujące deklaracje:

```
int fun(int);
int fun(double);
void fun(int, double);
void fun(double, int);
int n, p;
double x,y;
char c;
float z;
```

Które z podanych poniżej wywołań są poprawne, które funkcje zostaną wywołane, jakie będą wykonane
konwersje?

a) fun(n)	b) fun(x)	c) fun(n,x)
d) fun(x,n)	e) fun(c)	f) fun(n,p)
g) fun(n,c)	h) fun(n,z)	i) fun(z,z)

Zadanie 2.

Czy przeciążanie funkcji można stosować do referencji? Czy kompilator rozróżni te funkcje?

```
      void fun(int x);
      void fun(int &x);
```

Zadanie 3.

Czy rozróżnione zostaną następujące funkcje?

```
      void fun(char *x);
      void fun(const char *x);
```

Zadanie 4.

Zmodyfikować zadanie 2 z laboratorium tak, dotyczyło tablic dwuwymiarowych. Zadanie ma teraz postać: Opracować zestaw przeciążonych funkcji, które pozwalają

   d) wczytać do tablicy dwuwymiarowej całkowitej lub rzeczywistej zestaw liczb całkowitych lub rzeczywistych (funkcje mają nazwę Czytaj),
   e) wydrukować zawartość odpowiedniej tablicy (funkcje mają nazwę Drukuj);
   f) obliczyć sumę wszystkich elementów w tablicy lub we wskazanym wierszu (funkcje mają nazwę Suma).

Napisać program testujący opracowane funkcje.

Zadanie 5

Napisać program pozwalający scalać dwie tablice dwuwymiarowe o różnych rozmiarach. Założenia:

Tablice przechowywane są w postaci struktury:

```
      struct Tab

      {
      unsigned n; // liczba kolumn
      unsigned m; // liczba wierszy
      int *t; // elementy tablicy
      };
```

W programie wykorzystywane są funkcje:

```
      Tab NewTab(unsigned m, unsigned n); // tworzenie tablicy m x n,
      // wypełnionej zerami
      void wypelnij(Tab T, int x=0); // wypełnianie tablicy wartoscią x
      void drukuj(Tab T, char* nazwa); // drukowanie tablicy
      Tab scal(Tab T1, Tab T2); // scalanie tablic
      void UsunTab(Tab T1); // usuwanie tablic
      int max(int a, int b){ return (a>b)?a:b; } // funkcja pomocnicza:
      // max z dwóch liczb
```

Zadanie 6.

Napisać zestaw funkcji działających na datach. Napisać program testujący opracowane funkcje.

```
#include <iostream>
using namespace std;
struct DATA {
  int dzien;
  int miesiac;
```

```
  int rok;
};
// Prototypy funkcji
bool CzyPoprawna(const DATA &data);
int DniPomiedzy(const DATA &data1, const DATA &data2);
bool Wczesniejsza(const DATA &data1, const DATA &data2);
void DodajDni(const DATA &data1, DATA &data2, int dni);
ostream &operator<< (ostream &wy, const DATA &data);
// Stałe
const int dniWMiesiacu[12] = {
                              31, 28, 31, 30, 31, 30,
                              31, 31, 30, 31, 30, 31
                             };
// Dni od poczatku roku do poczatku miesiaca
const int dniDoMiesiaca[12] = {
                               0, 31, 59, 90, 120, 151,
                               181, 212, 243, 273, 304, 334
                              };
static char *nazwyMiesiecy[12] = {
                                    "Styczen", "Luty", "Marzec",
                                    "Kwiecien", "Maj", "Czerwiec",
                                    "Lipiec", "Sierpien", "Wrzesien",
                                    "Pazdziernik", "Listopad", "Grudzien"
                                  };
//************************************************************** bool
RokPrzestepny (int rok)
// "rok" musi być z zakresu 1900 i 2999.
// Zwraca prawdę, jeśli "rok" jest przestępny
{ ... }
static int DniOd1900(const DATA &data)
// "data" zawiera poprawną datę
// (pomiedzy 1 Stycznia 1990 i 31 Grudnia 2100)
// Zwraca liczbę dni od 1 stycznia 1990
{ ... }
//**************************************************************
bool CzyPoprawna(const DATA &data)
// Zwraca true, jeśli data jest z zakresu
// 1 Styczen 1990 i 31 Grudzien 2100
{ ... }
int DniPomiedzy(const DATA &data1, const DATA &data2)
// "data1" i "data2" zawierają poprawne dane
// Zwraca liczbę dni pomiędzy "data1" i "data2"
{ ... }
bool Wczesniejsza(const DATA &data1, const DATA &data2)
// "data1" i "data2" zawierają poprawne dane
// Zwraca true, jeśli "data1" jest wcześniejsza od "data2"
// W przeciwnym wypadku zwraca false
{ ... }


void DodajDni(const DATA &data1, DATA &data2, int dni)
// "data1" jest poprawna i "dni" >= 0.
// Wstawia do data2 nową datę: data1 + dni
{ ... }
ostream &operator<< (ostream &wy, const DATA &data) {
// Uzycie: cout << data;
// Drukuje dane w postaci "1 Styczen 1990".
  wy << data.dzien <<' '
  << nazwyMiesiecy[data.miesiac - 1] << ' '
  << data.rok;
```

```
  return wy;
}
//*************************************************************
// Przykład funkcji testującej
int main() {
  DATA data1, data2;
  int dni;
  cout << "Wpisz trzy liczby: dzien, miesiac, rok\n"
       << " (np. 12 10 2002) ==> ";
  cin >> data1.dzien >> data1.miesiac >> data1.rok;
  if (CzyPoprawna(data1))
    cout << data1 << endl;
  else {
    cout << "*** Niepoprawna data ***\n"
         << " (Poprawne daty sa z zakresu: "
         << " 1 Styczen 1990 i\n"
         << " 31 Grudzien 2100).\n";
    return 1;
  }
  cout << "O lie dni zwiekszyc date ==> ";
  cin >> dni;
  DodajDni(data1, data2, dni);
  cout << "Nowa data to " << data2 << endl;
  if (DniPomiedzy(data1, data2) == dni)
    cout << "Test ok.\n";
  else
    cout << "Blad w DodajDni lub DniPomiedzy.\n";
  }
```