Programowanie II

Lista 2

Funkcje, tablice i inne takie

Nowsza wygodniejsza postać nagłówka funkcji, wprowadzona przez C++ 11

```
4
5   #include "stdafx.h"
6   #include <iostream>
7    using namespace std;
8
9   auto dodaj(int a, double b)->decltype(a)
10  {
11      return a + b;
12      /*
13      nie musimy dzieki temu jednoznacznie określać typu zwracanego przez funkcję
14      możemy po wskazać po prostu, że typ wyniku będzie zgodny z typem jednego z argumentów
15      */
16  }
17
18
19  int main()
20  {
21      cout << "Wynik dodawania 5 i 6 wynosi " << dodaj(5, 6) << endl;
22      system("pause");
23      return 0;
24  }
```

```
Wynik dodawania 5 i 6 wynosi 11
Press any key to continue . . .
```

Przekazywanie parametrów przez referencję

C++ wprowadza pojęcie przekazywania parametrów do funkcji przez referencję (w C wymagało to określenia argumentu jako wskaźnika.

```
5   #include "stdafx.h"
6   #include <iostream>
7   #include <string>
8    using namespace std;
9
10  void zmien(int par1, int &par2)
11  {
12      /*
13      Zmiany dokoknane na argumencie poprzedzonym & będa ustrwalone po zkończeniu
14      działania funkcji
15      */
16      cout << "Parametry przed zmianami - " << par1<<", "<<par2 << endl;
17      par1 = 0;
18      par2 = 0;
19      cout << "Parametry po zmianach - " << par1 << ", " << par2 << endl;
20  }
21  int main()
22  {
23      int a = 4, b = 6;
24      cout << "Parametry przed wywlolaniem funkcji - "<< a << ", " << b << endl;
25      zmien(a, b);
26      cout << "Parametry po wywlolaniu funkcji - " << a << ", " << b << endl;
27      system("pause");
28
29      return 0;
30  }
```

```
Parametry przed wywlolaniem funkcji - 4, 6
Parametry przed zmianami - 4, 6
Parametry po zmianach - 0, 0
Parametry po wywlolaniu funkcji - 4, 0
Press any key to continue . . .
```

Domniemane argumenty funkcji

```cpp
#include "stdafx.h"
#include <iostream>
using namespace std;

/*
funkcja może posiadać dowolną liczbę argumentów domniemanych, jednak muszą
one wystapić na końcu listy. Funksja potęgowa oczekuje dwóch argumentów, ale
jesli podnosimy do kwadratu to możemy sobie uprościć wywołanie.
*/
double potega(double podstawa, int wykladnik=2)
{
    long wynik = podstawa;
    for (int i = 1; i < wykladnik; i++)
        wynik = wynik * podstawa;
    return wynik;
}
void funkcja_wiecej_param(int param1, int param2 = 0, int param3 = 0)
{
    cout << "Wartosci parametrow param1, param2, param3 to: " << param1 << ", " << param2 << ", " << param3 << endl;
}

int main()
{
    cout <<"2 do potegi 3 to "<< potega(2, 3)<<endl;
    //ustalamy aktualną wartość drugiego parametru
    cout << "2 do potegi 2 to " << potega(2) << endl;
    //nie ustalamy aktualnej wartości drugiego parametru
    cout << "Wywolanie tylko z 1 param. ";
    funkcja_wiecej_param(1);
    cout<< endl;
    cout << "Wywolanie z 2 param. ";
    funkcja_wiecej_param(1, 2);
    cout<<endl;
    cout << "Wywolanie z 3 param. ";
    funkcja_wiecej_param(1,2,3);
    cout << endl;
    system("pause");
    return 0;
```

```
2 do potegi 3 to 8
2 do potegi 2 to 4
Wywolanie tylko z 1 param. Wartosci parametrow param1, param2, param3 to: 1, 0, 0

Wywolanie z 2 param. Wartosci parametrow param1, param2, param3 to: 1, 2, 0

Wywolanie z 3 param. Wartosci parametrow param1, param2, param3 to: 1, 2, 3
```

Lokalne zmienne

Zmienne lokalne w funkcjach powoływane są do "życia" wraz z każdym uruchomieniem funkcji (to skutkuje oczywiście usunięciem poprzednich ich wartości). Aby temu zapobiec można określić je jako statyczne.

Przykład pokazuje również przypadek, w którym ciało funkcji użytkownika znajduje się poniżej funkcji main().

```cpp
#include "stdafx.h"
#include <iostream>
using namespace std;


void funkcja_a(void);
void funkcja_b();

int main()
{
    funkcja_a();
    funkcja_b();
    funkcja_a();
    funkcja_b();
    system("pause");
    return 0;
}

void funkcja_a(void)
{
    int x=1;
    static int y=0;
    //zmienna y zostanie zainicjowana wartośći 0 ale tylko przy pierwszym wywołaniu
    y+=10;
    // w kolejnym wywołaniu funkcji jest ona tylko zwiększana - "pamięta" wartość z poprzedniego wywołania
    cout << "Wywolanie funkcji a, wartosc x = " << x << ", wartosc y = " << y << endl;
}

void funkcja_b()
{
    int i = 1;
    static int j=0;
    j += 5;
    cout << "Wywolanie funkcji b, wartosc i = " << i << ", wartosc j = " << j<<endl;
}
```

```
Wywolanie funkcji a, wartosc x = 1, wartosc y = 10
Wywolanie funkcji b, wartosc i = 1, wartosc j = 5
Wywolanie funkcji a, wartosc x = 1, wartosc y = 20
Wywolanie funkcji b, wartosc i = 1, wartosc j = 10
Press any key to continue . . .
```

Przekazywanie jednowymiarowych tablic do funkcji

Przekazywanie tablic do funkcji realizowane powinno być przez referencję – np. w taki sposób:

```cpp
#include "pch.h"
#include <iostream>
#include <random>
using namespace std;


constexpr int roz_tab = 6;
int tablica[roz_tab];
void WyswietlTablice(int rozmiar, int wyswietlana[])
{
    for (int i = 0; i < rozmiar; i++)
        cout << wyswietlana[i] << endl;
}

void WypelnijTablice(int rozmiar, int wypelniana[])
{
    for (int i = 0; i < rozmiar; i++)
     wypelniana[i] =rand() % 10 +1;
    //tutaj oczywiście uzyskujemy liczby losowe z zakresu od 1 do 10
}
int main()
{
    WypelnijTablice(roz_tab, tablica);
    WyswietlTablice(roz_tab, tablica);
    system("pause");
}
```

```
2
8
5
1
10
5
Press any key to continue . . .
```

Przekazywanie wielowymiarowych tablic do funkcji

```
7   using namespace std;
8
9   constexpr int wiersze = 10;
10  constexpr int kolumny= 10;
11  char tablica[wiersze][kolumny];
12
13  void WypelnijTablice(int il_wierszy, int il_kolumn, char wypelniana[wiersze][kolumny])
14  {
15      char znak = '!';
16      for (int licz_wierszy = 0; licz_wierszy < il_wierszy; licz_wierszy++)
17      {
18          for (int licz_kolumn = 0; licz_kolumn < il_kolumn; licz_kolumn++)
19          {
20              wypelniana[licz_wierszy][licz_kolumn] = znak++;
21          }
22      }
23  }
24
25  void WyswietlTablice(int il_wierszy, int il_kolumn, char wyswietlana[][kolumny])
26  {
27
28      for (int licz_wierszy = 0; licz_wierszy < il_wierszy; licz_wierszy++)
29      {
30          for (int licz_kolumn = 0; licz_kolumn < il_kolumn; licz_kolumn++)
31          {
32              cout<<"|"<<wyswietlana[licz_wierszy][licz_kolumn];
33          }
34          cout << "|" << endl;
35      }
36  }
37
38  int main()
39  {
40      WypelnijTablice(wiersze, kolumny, tablica);
41      WyswietlTablice(wiersze, kolumny, tablica);
42
43      system("pause");
44  }
```

```
|!|"|#|$|%|&|'|(|)|*|
|+|,|-|.|/|0|1|2|3|4|
|5|6|7|8|9|:|;|<|=|>|
|?|@|A|B|C|D|E|F|G|H|
|I|J|K|L|M|N|O|P|Q|R|
|S|T|U|V|W|X|Y|Z|[|\|
|]|^|_|`|a|b|c|d|e|f|
|g|h|i|j|k|l|m|n|o|p|
|q|r|s|t|u|v|w|x|y|z|
|{|||}|~|⌂|Ç|ü|é|â|ä|
```

Zadanie 1 (6 pkt.)

Napisz program, który pozwoli określić jak długo należy przetrzymać lokatę kapitałową w banku, tak aby osiągnąć zadaną przez użytkownika wartość odsetek. Pamiętać należy, iż wysokość kwoty początkowej jak i oprocentowanie w skali roku oraz okres kapitalizacji i oczekiwana kwota odsetek podawane są przez użytkownika (nie wolno wykorzystywać wzoru na tzw. procent składany). Poszukiwana wartość powinna być wyznaczona przez stosowną funkcję.

Zadanie 2 (6 pkt.)

Napisz program, który wczyta 15 liczb różnych liczb całkowitych do jednowymiarowej tablicy, a następnie

- znajduje największą i najmniejszą z nich, a także ich pozycje w zbiorze (jedna funkcja liczy te 4 wartości)
- wyznaczy średnią wartość w tablicy
- zwróci pozycję wartości podanej przez użytkownika

Oczywiście powyższe funkcjonalności realizowane powinny być przez odpowiednie funkcje. Pamiętamy, że funkcje wyznaczają jakieś wartości, nie wyświetlają ich. Wyświetlane jest realizowane przez inny fragment kodu.

Zadanie 3 (8 pkt.)

Napisz program umożliwiający realizację następujących zadań na macierzy kwadratowej o ustalonym dowolnym wymiarze:

- Wyznaczenie sumy wartości poniżej przekątnej
- Wyznaczenie sumy wartości powyżej przekątnej
- Wyznaczenie sumy we wskazanym przez użytkownika wierszu lub kolumnie

Zawartość i analizowany zakres macierzy wczytywane mają być od użytkownika (maksymalny rozmiar jest stały, dowolnie ustalony przez programistę użytkownik może jednak chcieć pracować np. tylko na macierzy 3x3). Poza wyliczonymi sumami wyświetl również samą macierz po każdorazowej operacji. Program kończy swoje działanie dopiero po jednoznacznym wskazaniu tego przez użytkownika – pozwala wielokrotnie wyliczać poszczególne sumy