Część XVII C++ Funkcje

Funkcja bezargumentowa

Najprostszym przypadkiem funkcji jest jej wersja bezargumentowa.
Spójrzmy na przykład.

```
int pobierzLN();

int main(int argc, char *argv[])
{
    int a;
    a = pobierzLN();
    cout << "Zmienna a wynosi: " << a << endl;
    cout << "Zmienna wynosi: " << pobierzLN() << endl;

    system("PAUSE");
    return EXIT_SUCCESS;
}

int pobierzLN()
{
    int liczba;
    while(1)
    {
        cout << "Podaj liczbe wieksza od 0: " ;
        cin >> liczba;
        if (liczba > 0) return liczba;
    }
}
```

Tworzymy deklarację i definicję funkcji o nazwie pobierzLN() Funkcja będzie zwracała liczbę całkowitą. Do funkcji nie przekazujemy żadnych argumentów.

Działanie funkcji pobierzLN() polega na sprawdzeniu, czy wprowadzona przez użytkownika liczba jest większa od zera. Jeśli liczba będzie większa od zera, zostanie ona zwrócona przez funkcję (za pomocą instrukcji return, w innym wypadku zostanie wyświetlona prośba o ponowne wprowadzenie liczby.

Naszą funkcję możemy teraz wywołać w każdym miejscu programu, w którym chcemy pobrać od użytkownika liczbę większą od zera.

Ćwiczenie 1

1. Utwórz nowy projekt w Dev C++ i zapisz go na pulpicie w folderze o nazwie **nazwisko59**
2. Wprowadź do projektu modyfikacje tak aby wyglądał jak poniżej

```
#include <cstdlib>
#include <iostream>

using namespace std;

int pobierzLN();

int main(int argc, char *argv[])
{
    int a;
    a = pobierzLN();
    cout << "Zmienna a wynosi: " << a << endl;
    cout << "Zmienna wynosi: " << pobierzLN() << endl;

    system("PAUSE");
    return EXIT_SUCCESS;
}

int pobierzLN()
{
    int liczba;
    while(1)
    {
        cout << "Podaj liczbe wieksza od 0: " ;
        cin >> liczba;
        if (liczba > 0) return liczba;
    }
}
```

3. Z menu Uruchom wybieramy Kompiluj i uruchom.
4. Postępuj zgodnie z poleceniami wprowadzając liczby z klawiatury.

Wiemy, że możliwe jest stworzenie funkcji bezargumentowej. Widzieliśmy też kilka przykładów funkcji **dodawanie()** która pobierała dwa argumenty. Funkcja **main** jest natomiast przykładem funkcji o zmiennej liczbie argumentów. Oczywiście możemy stworzyć własną funkcję o zmiennej liczbie argumentów.

Funkcja nic niezwracająca

To, że do funkcji nie musimy przekazywać żadnego argumentu, już wiemy. Warto jeszcze wiedzieć, że funkcja nie musi również niczego zwracać. Pojawia się tylko jedno pytanie: W jaki sposób zadeklarować taką funkcję? Do tego celu służy specjalny typ danych void, który oznacza... brak typu.

Jeżeli ciekawi nas, co się stanie, gdy zadeklarujemy i zdefiniujemy funkcję bez podania żadnego typu danych zwracanego przez funkcję, sprawdźmy to. Dowiemy się, że mimo braku określenia typu, funkcja będzie zwracała wartości całkowitoliczbowe - inaczej mówiąc, domyślnym typem dla funkcji jest podobnie jak dla zmiennych typ **int**.

Napiszmy funkcję, której zadaniem będzie wyświetlanie na ekranie rzymskiego zapisu przekazanej do funkcji liczby (dla uproszczenia zadania przyjmijmy, że funkcja będzie działać dla liczb od 1 do 10).

```
void rzymska(int liczba);

int main(int argc, char *argv[])
{
    int x;
    cout << "Podaj liczbe od 1 do 10: ";
    cin >> x;
    cout << "Zapis rzymski liczby to: ";
    rzymska(x);
    cout << endl << endl;

    system("PAUSE");
    return EXIT_SUCCESS;
}

void rzymska(int liczba)
{
     switch (liczba){
         case 1: cout << "I"; break;
         case 2: cout << "II"; break;
         case 3: cout << "III"; break;
         case 4: cout << "IV"; break;
         case 5: cout << "V"; break;
         case 6: cout << "VI"; break;
         case 7: cout << "VII"; break;
         case 8: cout << "VIII"; break;
         case 9: cout << "IX"; break;
         case 10: cout << "X"; break;
         default: cout << liczba;
     }
}
```

Na początku zadeklarujmy naszą funkcję .
Zwróćmy uwagę na typ, jaki przypisaliśmy zwracanej przez nią wartości. Słowo kluczowe **void** oznacza, że funkcja nie zwraca wyniku, więc nie musi się w niej pojawiać instrukcja **return**.

Wewnątrz funkcji głównej możemy wywoływać funkcję rzymskie, przekazując do niej liczbę całkowitą.

Poniżej funkcji **main** definiujemy naszą funkcję -jej działanie powinno być jasne dla każdego.

Ćwiczenie nr 2

1. Utwórz nowy projekt w Dev C++ i zapisz go na pulpicie w folderze o nazwie **nazwisko60**
2. Wprowadź do projektu modyfikacje tak aby wyglądał jak poniżej

```
#include <cstdlib>
#include <iostream>

using namespace std;

void rzymska(int liczba);

int main(int argc, char *argv[])
{
    int x;
    cout << "Podaj liczbe od 1 do 10: ";
    cin >> x;
    cout << "Zapis rzymski liczby to: ";
    rzymska(x);
    cout << endl << endl;

    system("PAUSE");
    return EXIT_SUCCESS;
}

void rzymska(int liczba)
{
     switch (liczba){
         case 1: cout << "I"; break;
         case 2: cout << "II"; break;
         case 3: cout << "III"; break;
         case 4: cout << "IV"; break;
         case 5: cout << "V"; break;
         case 6: cout << "VI"; break;
         case 7: cout << "VII"; break;
         case 8: cout << "VIII"; break;
         case 9: cout << "IX"; break;
         case 10: cout << "X"; break;
         default: cout << liczba;
     }
}
```

3. Z menu Uruchom wybieramy Kompiluj i uruchom.
4. Postępuj zgodnie z poleceniami wprowadzając liczbę z klawiatury.

Domyślne argumenty

Twórcy języka C++ umożliwili przypisanie wartości domyślnych do jednego lub wielu argumentów przekazywanych do funkcji. Zobaczmy, jakie możemy dzięki temu osiągnąć korzyści.

► Napiszmy definicję funkcji o nazwie **vat()** , która będzie zwracała wartość pierwszego argumentu powiększoną o liczbę procent przekazaną w drugim argumencie. Jako typ wyniku zwracanego przez funkcję wpisujemy więc **float**

► Pierwszy argument również powinien mieć wartość **float** a drugi (określający stawkę procentową VAT-u) typ **int**.

► Wewnątrz funkcji **vat ()** umieszczamy stosowne obliczenia i zwracamy ich wynik za pomocą instrukcji **return**.
► Ponieważ definicję funkcji **vat()** umieściliśmy na końcu naszego kodu, musimy ją zadeklarować przed pierwszym wywołaniem (najlepiej tuż przed definicją funkcji głównej). W deklaracji funkcji **vat ()** inicjujemy zmienną **stawka** wartością . Dzięki temu, jeśli wywołując funkcję **vat()** , podamy tylko jeden argument (wartość netto), jako domyślna wartość stawki VAT zostanie przyjęta liczba 22. Jest to bardzo wygodne, ponieważ w znacznej większości produktów stawka podatku wynosi właśnie 22 procent.
► Aby sprawdzić działanie funkcji **vat()**, wewnątrz funkcji **main** umieszczamy jej wywołania. W pierwszym jako drugi argument przekazujemy liczbę 22,
w drugim - 7,
w trzecim nie przekazujemy w ogóle drugiego argumentu.

```
#include <cstdlib>
#include <iostream>

using namespace std;

float vat(float x, int stawka = 22);

int main(int argc, char *argv[])
{
    cout << "Cena=100 zl, VAT=22%, wartosc brutto wynosi " << vat(100,22) << endl;
    cout << "Cena=100 zl, VAT=7%, wartosc brutto wynosi " << vat(100,7) << endl;
    cout << "Cena=100 zl, VAT=22%, wartosc brutto wynosi " << vat(100) << endl;
```

Po skompilowaniu i uruchomieniu programu widzimy, że w wypadku nieprzekazania drugiego argumentu została obliczona wartość dla ustawionej przez nas domyślnie stawki 22 procent.

Ćwiczenie nr 3

1. Utwórz nowy projekt w Dev C++ i zapisz go na pulpicie w folderze o nazwie **nazwisko61**
2. Wprowadź do projektu modyfikacje tak aby wyglądał jak poniżej

```
#include <cstdlib>
#include <iostream>

using namespace std;

float vat(float x, int stawka = 22);

int main(int argc, char *argv[])
{
    cout << "Cena=100 zl, VAT=22%, wartosc brutto wynosi " << vat(100,22) << endl;
    cout << "Cena=100 zl, VAT=7%, wartosc brutto wynosi " << vat(100,7) << endl;
    cout << "Cena=100 zl, VAT=22%, wartosc brutto wynosi " << vat(100) << endl;

    system("PAUSE");
    return EXIT_SUCCESS;
}

float vat(float x, int stawka)
{
    return x * (1+float(stawka)/100);
}
```

3. z menu Uruchom wybieramy Kompiluj i uruchom.

Dzięki zdobytej wiedzy możemy teraz dużo prościej i czytelniej napisać pogram obliczający pola powierzchni różnych figur.

Ćwiczenie nr 4

1. Utwórz nowy projekt w Dev C++ i zapisz go na pulpicie w folderze o nazwie **nazwisko62**
2. Wprowadź do projektu modyfikacje tak aby wyglądał jak poniżej

```
#include <cstdlib>
#include <iostream>

using namespace std;

void oblicz_pole(char figura);
float pobierz_wartosc();
float pole_prostokata(float x, float y);
float pole_kola(float x);
float pole_trojkata(float x, float y);

int main(int argc, char *argv[])
{
    char decyzja;
    while(1)
    {
        cout << "Wybierz figure: " << endl;
        cout << "p - prostokat, k - kolo, t - trojkat, 0 - koniec " << endl;
        cin >> decyzja;

        if ((decyzja=='p')||(decyzja=='k')||(decyzja=='t')) oblicz_pole(decyzja);
        else if (decyzja=='0') break;
        else cout << "Zly wybor" << endl;
   }

    cout << endl << endl;

    system("PAUSE");
    return EXIT_SUCCESS;
}

void oblicz_pole(char figura)
{
     float a,b;
     cout << "Podaj dlugosc";
     switch (figura){
     case 'p': cout << "i bokow prostokata: "; a=pobierz wartosc(); b=pobierz wartosc();
```

```
            cout << "Pole prostokata wynosi: " << pole_prostokata (a, b) << endl; break;
      case 'k': cout << " promienia kola: "; a=pobierz_wartosc();
            cout << "Pole kola wynosi: " << pole_kola (a) << endl; break;
      case 't': cout << " podstawy i wysokosci trojkata: "; a=pobierz_wartosc(); b=pobierz_wartosc();
            cout << "Pole trojkata wynosi: " << pole_trojkata (a, b) << endl; break;
      }
}

float pobierz_wartosc()
{
      while (1)
      {
            float a;
            cin >> a;
            if (a > 0) return a;
      }
}

float pole_prostokata(float x, float y)
{
      return x*y;
}

float pole_kola(float x)
{
      return 3.1415*x*x;
}

float pole_trojkata(float x, float y)
{
      return x*y/2;
}
```

3. Skompiluj i uruchom programu

Wszystkie pliki z nazwiskiem i kolejnym numerem umieszczamy w swoim folderze nazwiskocplusplus na serwerze.