Lekcja 19 – przekazywanie parametrów przez funkcje

Temat: Podprogramy – przekazywanie i odbieranie parametrów poprzez funkcje w C++

Kod programu źródłowego:

```
/*************************************************************

Przekazywanie parametrów przez funkcje C++
*************************************************************/
#include <iostream>
#include <math.h>

using namespace std;


void pole (int a1, int b1)
{
    int p1;
    p1 = a1 * b1;
    cout << "Pole prostokąta wynosi " << p1 << endl;
}

float przekatna (int a, int b)
{
    float d;
    d = sqrt (a*a + b*b);
    return d;
}

float obwod (int a, int b)
{
    int o;
    o = 2 * a + 2 *b;
    return o;
}

int main()
{
    int a,b,p;
    float d;
    cout<<"Podaj pierwszy bok prostokąta ";
    cin >>a;
    cout<<"Podaj drugi bok prostokąta ";
    cin >>b;

    pole (a,b);

    d = przekatna (a,b);
    cout << "Przekątna prostokąta wynosi " << d << endl;

    cout << "Odwód prostokąta wynosi " << obwod (a,b) << endl;
```

```
    return 0;
}
```

Przekazywanie danych: Podprogram może samodzielnie zająć się zdobyciem odpowiednich danych do pracy. Można mu również nakazać na jakich danych ma pracować.

Należy wtedy przy wywoływaniu funkcji podać odpowiednie argumenty z jakimi ma pracować.

Budowa funkcji w C++ i przekazywanie parametrów:

```
typ_zwracanej_wartosci nazwa_funkcji( typ_argumentu_1
nazwa_argumentu_1 /*,...*/, typ_argumentu_n nazwa_argumentu_n )
{
    return zwracana_wartosc;
}
```

Przykład 1: wywołując funkcję **pole** podajemy 2 wartości typu **int**: **a** i **b**.

```
pole (a,b);
```

Wywoływana funkcja ma postać:

```
void pole (int a1, int b1)
{
    int p1;
    p1 = a1 * b1;
    cout << "Pole prostokąta wynosi " << p1;
}
```

Wszystkie działania są wykonywane wewnątrz funkcji. Nie zwraca żadnych wartości.

Przykład 2: wywołując funkcję **przekatna** podajemy 2 wartości typu **int**: **a** i **b**.

```
d = przekatna (a,b);
```

Wywoływana funkcja ma postać:

```
float przekatna (int a, int b)
{
    float d;
    d1 = sqrt (a*a + b*b);
    return d;
}
```

Działania są wykonywane wewnątrz funkcji a wynik jest zwracany do wartości d.

Zakres danych:

Dane mogą być globalne lub lokalne. Globalne obowiązują w całym programie. Lokalne obowiązują tylko w zakresie danej funkcji w jakiej je zainicjalizowano.

Dlatego w funkcji **pole** użyto odpowiedników **a1** i **b1** zamiast **a** i **b**. pozwala to nie pomylić zmiennych przy analizie programu.

W funkcji **przekatna** użyto nazw **a** i **b**. Zmienne **a** i **b** w tej funkcji są innymi zmiennymi niż te w funkcji **main**. Jednak wartości przekazywane przez wywołanie funkcji są te same, co ułatwia analizę danych.

Podobieństwo nazw może zrodzić pewne problemy i w wielu wypadkach używa się konwencji zastosowanej w funkcji **pole**.

Zwracanie wartości:

Użycie przy wywołaniu poniższej komendy oznacza, że funkcja **przekatna** ma dokonać obliczeń na zmiennych a i b, a wynik ma zostać przypisany doi zmiennej d.

```
d = przekatna (a,b);
```

Wynik może też być bezpośrednio wypisany na ekranie, co pokazuje poniższy przykład.

```
    cout << "Odwód prostokąta wynosi " << obwod (a,b) << endl;
```

Ćwiczenia

1. Napisz program wczytujący wartość promienia koła **r** liczący właściwości koła:
    a. Pole
    b. Obwód
    c. średnica

    Użyj do obliczeń funkcji z przekazywaniem parametrów.

2. Mamy trójkąt prostokątny i znane przyprostokątne **a** i **b**. Napisz program liczący:
    a. Długość przeciwprostokątnej c,
    b. Pole tego trójkąta,
    c. Obwód tego trójkąta.
    d. Wysokość opuszczoną na przeciwprostokątną
    e. Promień okręgu opisanego
    f. Promień okręgu wpisanego

    Użyj do obliczeń funkcji z przekazywaniem parametrów.