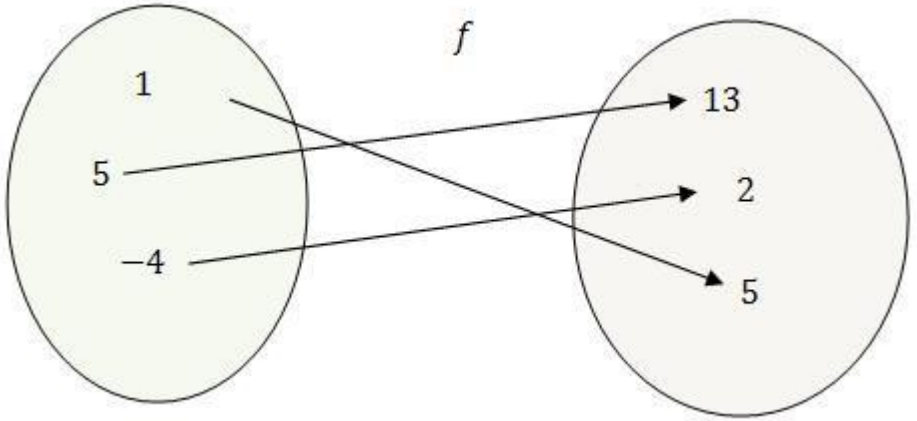
Funkcje w C++

Funkcja w matematyce

Dzięki funkcjom możemy opisywać otaczający nas świat oraz badać zjawiska jakie w nim zachodzą.
Najczęściej **funkcję definiuje** się jako przyporządkowanie każdemu elementowi jednego zbioru, dokładnie jednego elementu drugiego zbioru.

Przedstawienie funkcji za pomocą **grafu**



Graf składa się z dwóch zbiorów. W pierwszym znajdują się argumenty x, a w drugim wartości y. Strzałki pokazują sposób działania funkcji (przyporządkowują każdemu argumentowi dokładnie jedną wartość).

Funkcje w C++

Tabelka dla funkcji przedstawionej za pomocą grafu

x	-4	1	5
y	2	5	13

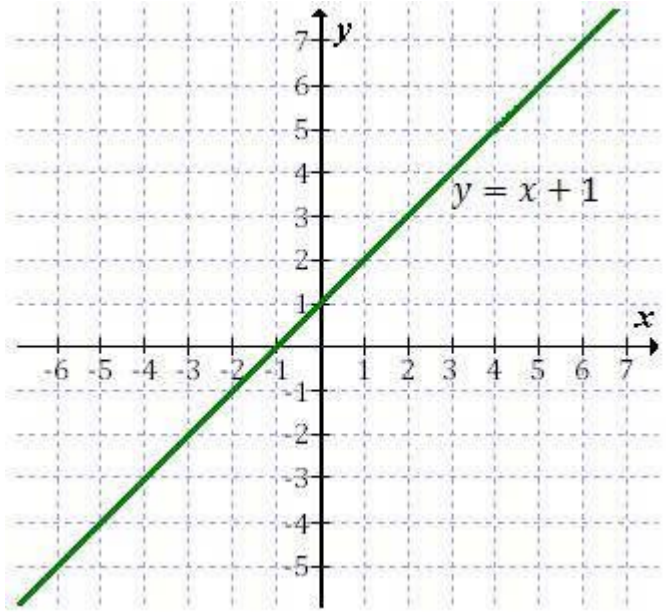
Tabelkę często wykorzystuje się podczas rysowania wykresów funkcji. Gdy znamy wzór funkcji, np. y=3x, to możemy sporządzić następującą tabelkę:

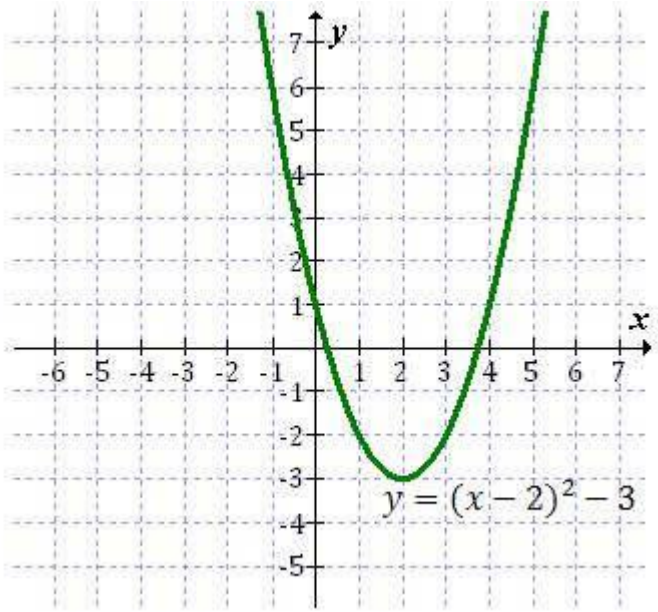
x	-2	-1	0	1	2	3
$y = 3x$	-6	-3	0	3	6	9

Dolny rząd tabelki uzupełniamy podstawiając do wzoru funkcji pod x-a odpowiednie argumenty z pierwszego rzędu.

Funkcje w C++

Wykresy funkcji





Funkcje w C++

Wzór

Jest to najlepszy sposób prezentowania funkcji.
Mając dany wzór funkcji możemy zawsze narysować jej wykres, sporządzić tabelkę, a co najważniejsze - określić wszystkie własności danej funkcji.
Istnieją dwa równoważne sposoby zapisywania wzorów funkcji. Można to zrobić w taki sposób:

y=[tu piszemy wzór funkcji]

albo w taki:

f(x)=[tu piszemy wzór funkcji]

Funkcje w C++

Wartości funkcji

Wartości funkcji - to wszystkie y-ki jakie przyjmuje wykres funkcji.

Zbiór argumentów to **zbiór *x*-ów**.

Zbiór wartości to **zbiór *y*-ów**.

Jeśli mamy podany wzór funkcji, to możemy obliczyć wartość, jaką przyjmuje funkcja dla dowolnego argumentu x.
Wystarczy, że podstawimy we wzorze funkcji pod x-a podaną liczbę, a w rezultacie otrzymamy dla niej szukaną wartość y.

Funkcje w C++

Podobieństwo zapisu funkcji w C++ (i innych językach programowania) do zapisu **matematycznego.**

$$f(x) = x^2$$

Wartość zwracana

Argument funkcji

x=5

f(5) =x*x=5*5 =25

Funkcja przyjmuje argument (liczbę z określonego zbioru) x i zwraca liczbę podniesiona do kwadratu.

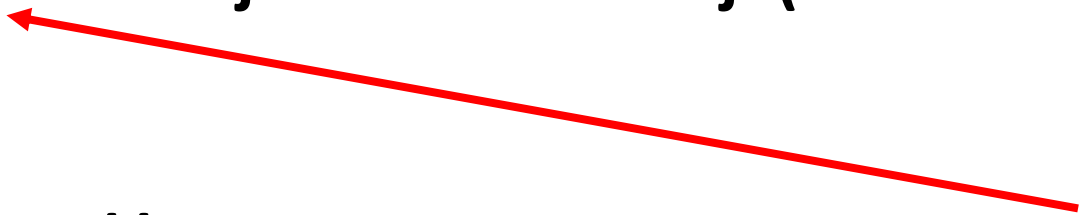
$y = x^2$

y = 25

x	1	2	3	4	5
$y = x^2$	1	4	9	16	25

Funkcje w C++ zwracające wartość-definicja funkcji

```
typWartościZwacanej   nazawaFunkcji (listaParametrów)
{
instrukcje
return wartość; // wartość jest rzutowana na typWartościZwracanej
}
```



Funkcje w C++ zwracające wartość

Definicja funkcji bez argumentu w C++

```
int zrobCos ()
{
return 2+3;
}
```

```
#include <iostream>

using namespace std;

int zrobCos()
{
  return 2+3;
}

int main ()
{
cout << "Wynikiem dodawania 2 do 3 jest " << zrobCos();
}
```

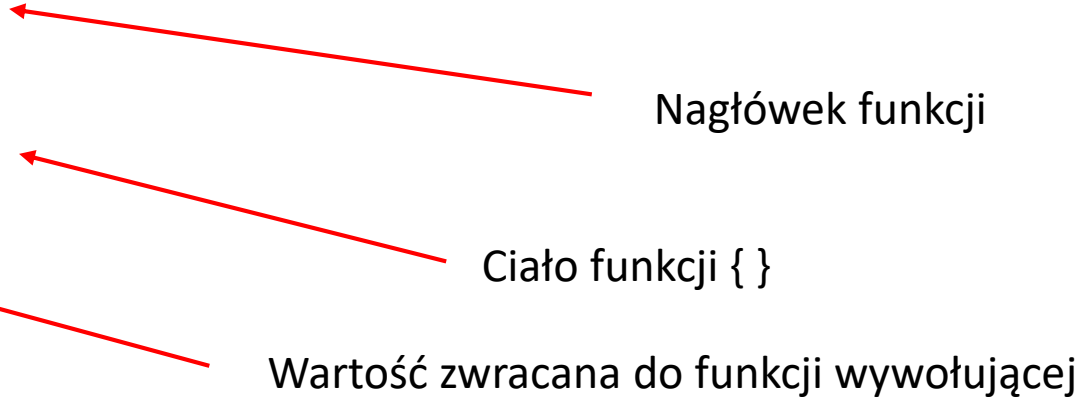
Definicja funkcji

Wywołanie funkcji funkcji

Funkcje w C++

Definicja funkcji z argumentem w C++

```
int iks_kwadrat (int x)
{
return x*x;
}
```



Funkcja iks_kwadrat przyjmuje jeden argument. Przed nazwą funkcji iks_kwadrat podany jest typ zwracanej wartości. Po nazwie, w nawiasie znajduje się **argument** funkcji. W ciele funkcji znajduje się instrukcja obliczająca i zwracająca **wartość** funkcji.

Funkcje w C++

```
#include <iostream>

using namespace std;

int main ()

{
int x;
cout << " Podaj liczbe calkowita \n";
cin >> x;
cout << " Wynik podniesienia liczby do kwadratu =  " << x*x;

return 0;
}
```

Program obliczający kwadrat liczby całkowitej bez użycia funkcji

Funkcje w C++

```
#include <iostream>

using namespace std;

int iks_kwadrat (int x)
{
    return x*x;
}

int main ()
{
cout << "Wynik podniesienia liczby do kwadratu =  " << iks_kwadrat ( 2 );

return 0;
}
```

Definicja funkcji

Wywołanie funkcji z argumentem

Funkcja główna

Program podnoszący liczbę całkowitą do kwadratu - zawiera definicję funkcji i wywołanie funkcje iks_kwadrat

Funkcje w C++

```
#include <iostream>

using namespace std;

int dodaj (int x, int y)
{
    return x + y;
}

int main ()
{
cout << "Wynikiem dodawania 3 do 4 jest " << dodaj( 3, 4 );
return 0;
}
```

Przykład funkcji dwuargumentowej

Funkcje w C++ nie zwracające wartości

Funkcja , która nie zwraca wartości w C++ nosi nazwę **void**, czyli pustka, próżnia. Składnia funkcji:

```
void nazawFunkcji
(listaParametrów)
{
instrukcje
return;// opcjonalnie
}
```

```
void witaj (int n)
{
for (int i=0;  i<n; i++)
cout << ”Witaj!”;
cout <<endl;
```

Funkcje w C++ nie zwracające wartości

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

void witaj (int n)
{
for (int i=0;  i<n; i++)
cout << "Witaj!" << endl;

}

int main()
{
    witaj(3);
    return 0;
}
```

Przykład funkcji nie zwracającej wartości

Definicja funkcji void

Wywołanie funkcji void

Zasięg zmiennych w C++

Wszystkie zmienne w C++ można podzielić na zmienne **globalne** i zmienne **lokalne**.

W dużym uproszczeniu różnica między tymi dwoma rodzajami zmiennych jest następująca: zmienne globalne są to takie zmienne, które są dostępne w **całym programie i przez cały czas jego działania**, natomiast zmienne lokalne są dostępne tylko w **pewnej części programu**, zazwyczaj tylko w **pewnej chwili działania programu**, a nie przez cały czas.

ZMIENNA GLOBALNA

Zaletą zmiennych globalnych jest to, że są one widoczne w całym programie.

Zmienne globalne deklaruje się pomiędzy blokiem dołączonych plików nagłówkowych a funkcją main.

```
#include <iostream>

using namespace std;

int liczba;
// zmienna liczba zostala domyslnie zainicjalizowana wartoscia 0

int main()
{
  liczba+=5; // 0 + 5 = 5
  cout <<"Liczba wynosi "<<liczba<<endl;

  cout <<endl<<"Nacisnij ENTER aby zakonczyc..."<<endl;
  getchar();
  return 0;
}
```

Zasięg zmiennych w C++

ZMIENNA LOKALNA

Aby utworzyć zmienną lokalną, można ją utworzyć w dowolnym miejscu danej funkcji, czyli również w funkcji main.

Nawiasy klamrowe wyznaczają zasięg lokalny

Zmienne które zostały zadeklarowane pomiędzy nawiasami klamrowymi są widoczne tylko między tą parą nawiasów. Po nawiasie zamykającym, zmienna przestaje być widoczna, bowiem przestaje ona już istnieć.

```
#include <iostream>

using namespace std;

int main()
{
  int liczba;

  // zmienna liczba ma wartosc nieokreslona
  liczba+=5; // ? + 5 = ?
  cout <<"Liczba wynosi "<<liczba<<endl;

  cout <<endl<<"Nacisnij ENTER aby zakonczyc..."<<endl;
  getchar();
  return 0;
}
```

Prototypowanie funkcji – prawidłowy zapis programu z funkcją

Aby zwiększyć szanse na prawidłowe wykonanie programu zawierającego funkcje stosuje się prototypy – informacje dla kompilatora o funkcji. Prototypy zapewniają, że :

- Kompilator prawidłowo obsłuży zwracaną wartość.
- Kompilator sprawdzi, czy przekazano prawidłową liczbę parametrów.
- Kompilator sprawdzi typy parametrów, jeśli będą niezgodne, w miarę możliwości przeprowadzi wymagane konwersje.

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

int dodaj (int x, int y) // prototyp funkcji

int main ()
{
cout << "Wynikiem dodawania 3 do 4 jest " << dodaj( 3, 4 ); // wywołanie funkcji
return 0;
}

int dodaj (int x, int y) //definicja funkcji , funkcje umieszczamy pod funkcja główną
{
    return x + y;
}
```

Na podstawie tego programu napisz programy z użyciem funkcji, które obliczą:

1. Pole kwadratu.
2. Pole trójkąta.
3. Pole prostokąta.
4. Pole trapezu.
5. Objętość sześcianu.