

Informatyka 2 (EZ1E3012)

Politechnika Białostocka - Wydział Elektryczny
Elektrotechnika, semestr III, studia niestacjonarne I stopnia
Rok akademicki 2021/2022

Pracownia nr 5 (20.11.2021)

dr inż. Jarosław Forenc

Program w języku C

- Program w języku C składa się z **funkcji i zmiennych**
 - funkcje zawierają instrukcje wykonujące operacje
 - zmienne przechowują wartości

```
#include <stdio.h>    /* przekatna kwadratu */
#include <math.h>

int main(void)
{
    float a = 10.0f, d;

    d = a * sqrt(2.0f);
    printf("Bok = %g, przekatna = %g\n", a, d);

    return 0;
}
```

Bok = 10, przekatna = 14.1421

Program w języku C

- Program w języku C składa się z **funkcji i zmiennych**
 - funkcje zawierają instrukcje wykonujące operacje
 - zmienne przechowują wartości

```
#include <stdio.h>    /* przekatna kwadratu */
#include <math.h>

int main(void)
{
    float a = 10.0f, d;

    d = a * sqrt(2.0f);
    printf("Bok = %g, przekatna = %g\n", a, d);

    return 0;
}
```

definicja funkcji

Program w języku C

- Program w języku C składa się z **funkcji i zmiennych**
 - funkcje zawierają instrukcje wykonujące operacje
 - zmienne przechowują wartości

```
#include <stdio.h>    /* przekatna kwadratu */
#include <math.h>

int main(void)
{
    float a = 10.0f, d;

    d = a * sqrt(2.0f);
    printf("Bok = %g, przekatna = %g\n", a, d);

    return 0;
}
```

wywołania funkcji

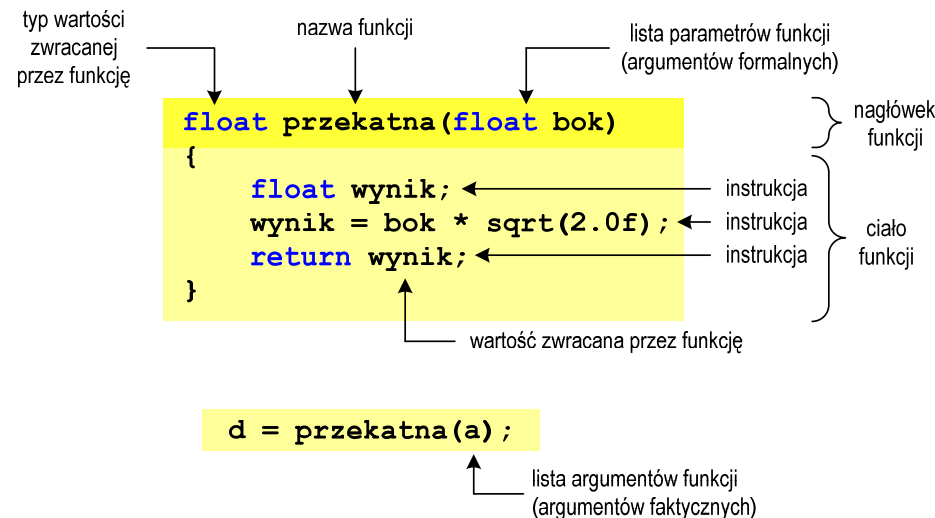
Funkcje w języku C

```
#include <stdio.h> /* przekatna kwadratu */
#include <math.h>

float przekatna(float bok)
{
    float wynik;
    wynik = bok * sqrt(2.0f);
    return wynik;
}

int main(void)
{
    float a = 10.0f, d;
    d = przekatna(a);
    printf("Bok = %g, przekatna = %g\n", a, d);
    return 0;
}
```

Ogólna struktura funkcji w języku C



Argumenty funkcji

- Argumentami funkcji mogą być stałe liczbowe, zmienne, wyrażenia arytmetyczne, wywołania innych funkcji

```
d = przekatna(a);
d = przekatna(10);
d = przekatna(2*a+5);
d = przekatna(sqrt(a)+15);
```

- Wywołanie funkcji może być argumentem innej funkcji

```
printf("Bok = %g, przekatna = %g\n",
      a, przekatna(a));
```

Parametry funkcji

- Parametry funkcji traktowane są tak samo jak zmienne zadeklarowane w tej funkcji i zainicjalizowane wartościami argumentów wywołania

```
float przekatna(float bok)
{
    float wynik;
    wynik = bok * sqrt(2.0f);
    return wynik;
}
```

- Funkcję `przekatna()` można zapisać w prostszej postaci:

```
float przekatna(float bok)
{
    return bok * sqrt(2.0f);
}
```

Parametry funkcji

- Jeśli funkcja ma kilka **parametrów**, to dla każdego z nich podaje się:
 - typ parametru
 - nazwę parametru
- Parametry oddzielane są od siebie przecinkami

```
/* przekatna prostokata */  
float przekatna(float a, float b)  
{  
    return sqrt(a*a+b*b);  
}
```

Prototyp funkcji

- Czy można zmienić kolejność definicji funkcji w kodzie programu?

```
#include <stdio.h> /* przekatna prostokata */  
#include <math.h>  
  
float przekatna(float a, float b)  
{  
    return sqrt(a*a+b*b);  
}  
  
int main(void)  
{  
    float a = 10.0f, b = 5.5f, d;  
    d = przekatna(a,b);  
    printf("Przekatna prostokata = %g\n",d);  
    return 0;  
}
```

definicja funkcji

definicja funkcji

Prototyp funkcji

- Czy można zmienić kolejność definicji funkcji w kodzie programu?

```
#include <stdio.h> /* przekatna prostokata */  
#include <math.h>  
  
int main(void)  
{  
    float a = 10.0f, b = 5.5f, d;  
    d = przekatna(a,b);  
    printf("Przekatna prostokata = %g\n",d);  
    return 0;  
}  
  
float przekatna(float a, float b)  
{  
    return sqrt(a*a+b*b);  
}
```

definicja funkcji

definicja funkcji

Prototyp funkcji

- Czy można zmienić kolejność definicji funkcji w kodzie programu?

```
#include <stdio.h> /* przekatna prostokata */  
#include <math.h>  
  
int main(void)  
{  
    float a = 10.0f, b = 5.5f, d;  
    d = przekatna(a,b);  
    printf("Przekatna prostokata = %g\n",d);  
    return 0;  
}  
  
float przekatna(float a, float b)  
{  
    return sqrt(a*a+b*b);  
}
```

definicja funkcji

error C3861: 'przekatna':
identifier not found

Prototyp funkcji

```
#include <stdio.h> /* przekatna prostokata */  
#include <math.h>
```

```
float przekatna(float a, float b);
```

prototyp funkcji

```
int main(void)
```

definicja funkcji

```
{  
    float a = 10.0f, b = 5.5f, d;  
    d = przekatna(a,b);  
    printf("Przekatna prostokata = %g\n",d);  
    return 0;  
}
```

```
float przekatna(float a, float b)
```

definicja funkcji

```
{  
    return sqrt(a*a+b*b);  
}
```

Funkcje - argumenty/parametry, zwracana wartość

- Prezentowane funkcje miały argumenty i zwracały wartości

```
typ nazwa(parametry)  
{  
    instrukcje;  
    return wartość;  
}
```

```
float przekatna(float bok)  
{  
    float wynik;  
    wynik = bok * sqrt(2.0f);  
    return wynik;  
}
```

```
typ zm;  
zm = nazwa(argumenty);
```

```
float d;  
d = przekatna(a);
```

- Można zdefiniować także funkcje, które nie mają argumentów i/lub nie zwracają żadnej wartości

Funkcje - argumenty/parametry, zwracana wartość

- Funkcja bez argumentów i nie zwracająca wartości

```
void nazwa(void)  
{  
    instrukcje;  
    return;  
}
```

```
void nazwa()  
{  
    instrukcje;  
    return;  
}
```

```
void nazwa(void)  
{  
    instrukcje;  
}
```

```
void nazwa()  
{  
    instrukcje;  
}
```

- Wywołanie funkcji:

```
nazwa();
```

Przekazywanie argumentów do funkcji

```
#include <stdio.h>  
void fff(int kopiax, int *wsky)  
{  
    kopiax = 20;  
    *wsky = 20;  
}  
int main(void)  
{  
    int x = 10, y = 10;  
    fff(x,&y);  
    printf("x = %d\n",x);  
    printf("y = %d\n",y);  
    return 0;  
}
```

```
x = 10  
y = 20
```

- Przekazywanie argumentów do funkcji:
x - przez **wartość**
y - przez **wskaźnik**

Parametry funkcji - wektory

- Wektory przekazywane są do funkcji przez **wskaźnik**
- Nie jest tworzona kopia tablicy, a wszystkie operacje na jej elementach odnoszą się do tablicy z funkcji wywołującej
- W nagłówku funkcji podaje się typ elementów tablicy, jej nazwę oraz nawiasy kwadratowe z liczbą elementów tablicy lub same nawiasy kwadratowe

```
void fun(int tab[5])  
{  
    ...  
}
```

```
void fun(int tab[])  
{  
    ...  
}
```

- W wywołaniu funkcji podaje się tylko jej nazwę (bez nawiasów kwadratowych)

```
fun(tab);
```

Parametry funkcji - macierze

- Macierze przekazywane są do funkcji przez **wskaźnik**
- W nagłówku funkcji podaje się typ elementów tablicy, jej nazwę oraz w nawiasach kwadratowych liczbę wierszy i kolumn lub tylko liczbę kolumn

```
void fun(int tab[2][3])  
{  
    ...  
}
```

```
void fun(int tab[][3])  
{  
    ...  
}
```

- W wywołaniu funkcji podaje się tylko jej nazwę (bez nawiasów kwadratowych)

```
fun(tab);
```

Parametry funkcji - struktury

- Struktury przekazywane są do funkcji przez **wartość** (nawet jeśli daną składową jest tablica)

```
#include <stdio.h>  
#include <math.h>  
  
struct pkt  
{  
    float x, y;  
};  
  
float odl(struct pkt pkt1, struct pkt pkt2)  
{  
    return sqrt(pow(pkt2.x-pkt1.x, 2) +  
               pow(pkt2.y-pkt1.y, 2));  
}
```

Parametry funkcji - struktury (przykład)

```
int main(void)  
{  
    struct pkt p1 = {2,3};  
    struct pkt p2 = {-2,1};  
    float wynik;  
  
    wynik = odl(p1,p2);  
  
    printf("Punkt nr 1: (%g,%g)\n",p1.x,p1.y);  
    printf("Punkt nr 2: (%g,%g)\n",p2.x,p2.y);  
    printf("Odleglosc = %g\n",wynik);  
  
    return 0;  
}
```

```
Punkt nr 1: (2,3)  
Punkt nr 2: (-2,1)  
Odleglosc = 4.47214
```