Creative Commons License: Attribution Share Alike

Funkcje

Pewne zestawy operacji, przykładowo zależne od zmiennych, możemy zebrać w grupy (funkcje) i wywoływać tak jak `circle()` i `line()`. Przykład z poprzedniego ćwiczenia możemy zamknąć w funkcji:

```
void obrazek(int h, int r) {
  line( 10, 0, 0, h);
  line( 10, 0, 2 * r, h);
  circle( 10 + r, h, r);
}
```

Pierwsza linia deklaruje funkcję, która jest zależna od dwóch parametrów: `h` i `r`. Taką funkcję, możemy przykładowo wywołać tak: `obrazek(100, 50);`. Spowoduje to wykonanie powyższych trzech operacji dla $h = 100$ i $r = 50$.

Pamiętaj: Nową funkcję napisz przed funkcją `main`.

W funkcji `main` wywołujemy funkcję `obrazek`, tak jak `circle` czy `line`:

```
void main() {
    graphics(200, 200);
    obrazek(100, 50);
    wait();
}
```

Ćwiczenia

Napisz i wywołaj dowolne dwie z poniższych funkcji:

- `prostokat(x, y, a, b)` — Rysuje prostokąt o bokach a i b i środku w (x, y).
- `kwadrat(x, y, r)` — Rysuje kwadrat o boku $2r$, środku w (x, y) i wpisane koło o promieniu r.
- `ludzik(x, y, h)` — Rysuje ludzika o wysokości h i środku głowy w (x, y).
- `olimpiada(x,y)` — Rysuje koła olimpijskie o środku w (x, y).
- `okno(a)` — Używając funkcji do rysowania prostokąta rysuje okno o boku a.

Trochę więcej szczegółów

Omówmy pewne rzeczy trochę dokładniej.

Typy

W C i C++ *musimy* deklarować zmienne, tzn. musimy powiedzieć, jakich będziemy używać zmiennych i jakich będą one typów. Deklaracje piszemy `typ zmienna1, zmienna2, ...;`. Najważniejsze typy to:

- `int` — Liczba całkowita (32-bitowa, od -2^{31} do 2^{31}).
- `float` — Liczba zmiennoprzecinkowa. Może opisywać ułamki dziesiętne z ok. 7 cyframi znaczącymi (32-bity)
- `double` — Liczba zmiennoprzecinkowa. Ma 16 cyfr znaczących (64-bity).

Pamiętaj: Jeśli używasz liczb rzeczywistych (a nie całkowitych), używaj typu `double`.

Pierwszym przykładem niech będzie:

```
double a;
a = 0;

while (a < 2 * 3.14) {
    circle(a * 100, sin(a) * 100 + 100, 3);
    a = a + 0.001;
}
```

Program ten, za pomocą kółek o promieniu 3, narysuje wykres funkcji sinus przeskalowany o 100.

Ćwiczenia

Używając analogicznej pętli, wykonaj dowolne dwa z poniższych zadań:

- Narysuj wykres a^2.
- Narysuj punkty o współrzędnych $x = 100 \cdot \sin a + 100$ i $y = 100 \cdot \cos a + 100$.

Creative Commons License: Attribution Share Alike

- Narysuj punkty o współrzędnych $x = 100 \cdot \sin a \cos 4a + 100$ i $y = 100 \cos a \cos 4a + 100$.
- Narysuj punkty o współrzędnych $x = 100 \cdot r \cdot \sin a + 100$ i $y = 100 \cdot r \cdot \cos a + 100$, gdzie $r = \frac{\cos a + 2}{3}$ (niech r będzie kolejną zmienną).

Typy — pułapki

Ważne, aby pamiętać, że liczby bez przecinka dziesiętnego, są uważane za całkowite, tzn. wykonywane są na nich działania tak jak dla liczb całkowitych. Dlatego `1/4` da jako wynik 0! Wynik 0.25 zostanie obcięty do liczby całkowitej. Żeby tego uniknąć, możemy napisać `1.0/4` lub jeszcze lepiej `1.0/4.0`. Możemy także bezpośrednio 'rzutować' zmienną z typu `int` na typ `double` pisząc: `(double) zmienna`.

Pamiętaj: Wszędzie, gdzie robisz obliczenia, używaj `double`. Unikaj mieszania liczb całkowitych i zmiennoprzecinkowych. Nigdy nie pisz ułamków jako 1/3!

Ćwiczenia

Przeanalizuj (i przetestuj) wynik tego programu. Które linie nie dadzą pożądanego efektu?

```
double a;
a = 0;

while (a < 2) {
    circle(a * 100, sin(a * 3.14) * 100 + 100, 3);
    circle(a * 100, sin(a * (314 / 100)) * 100 + 100, 3);
    circle(a * 100, sin((a * 314) / 100) * 100 + 100, 3);
    a = a + 0.001;
}
```

Funkcje po raz drugi

Zestawy operacji, które powtarzamy w programie wielokrotnie, możemy zamknąć w funkcjach. Taka funkcja „połyka" parametry i coś z nimi robi. Dla przykładu:

```
void kreski(int n, double r) {
  int i;
  i = 0;

  while (i < n) {
    line(i, 0, i, r * i);
    i = i + 1;
  }
}
```

W pierwszej linii mówimy:

- jak nazywa się funkcja — `kreski`,
- jakie ma parametry — `n` typu `int` i `r` typu `double`,
- jakiego typu zwraca wartość — w naszym wypadku `void` oznacza, że nic nie zwraca.

Gdy gdziekolwiek w funkcji `main` użyjemy wywołania `kreski(20, 0.4);` jako efekt działania funkcji otrzymamy 20 pionowych kresek o długości od 0 do 0.4·19 (dlaczego 19 a nie 20?).

Taką funkcję możemy wykonać wielokrotnie dla różnych wartości n i r:

```
void main() {
  graphics(200,200);

  kreski(10, 1.000);
  kreski(20, 0.500);
  kreski(30, 0.333);
  kreski(40, 0.250);

  wait();
}
```

Ćwiczenia

Napisz i wywołaj dwie spośród niżej wymienionych funkcji:

- Funkcja, która narysuje ludzika wysokości h i środku głowy w (x, y).

- Funkcja, która w pętli narysuje tłum (używając poprzedniej funkcji).
- Funkcja, która narysuje n kółek w punkcie (x, y) o coraz większych promieniach.
- Funkcję, która narysuje wielokąt foremny o n bokach.

Instrukcja warunkowa

Kolejnym bloczkiem składowym programowania, jest instrukcja warunkowa. Sprawdza ona warunek i wykonuje pewną część kodu, gdy tylko warunek jest spełniony.

```
double x, y;
x = 2.0;

if (x > 0) {
  y = sqrt(x);
} else {
  y = 0;
}
```

Instrukcje te sprawdzają czy $x > 0$ i jeśli jest to prawdą, wstawiają $\sqrt{x}$ do zmiennej y. Gdy warunek nie jest spełniony, wykonywana jest część po `else`, więc wstawiane jest 0 do y. W ten sposób możemy zabezpieczyć się na przykład przed niemożliwymi obliczeniami, albo uzależnić działanie programu od jakiś wartości.

Zobaczmy prosty przykład:

```
double a;
a = 0;

while (a < 2 * 3.14) {
  if (a < 2) {
    circle(sin(a) * 100 + 100, cos(a) * 100 + 100, 5);
  } else {
    circle(sin(a) * 100 + 100, cos(a) * 100 + 100, 10);
  }

  a = a + 0.001;
}
```


Gdyby nie instrukcja `if`, ten program narysował by koło z małych kółek. Teraz, gdy kąt `a` przekroczy 2 radiany, zmieni promień kółeczka z 5 na 10

Ćwiczenia

Napisz program który:

- Dla parametru w rysuje wykres $x^2 - w$, przeskalowany o 100 w obu kierunkach i przesunięty na środek okna (patrz poprzedni przykład).
- Narysuje większe kółka w miejscach przecięcia wykresu z osią x (*jeżeli* przecina).
- Zmodyfikuj program aby działał dla dowolnych a, b, c i funkcji $a \cdot x^2 + b \cdot x + c$.