Programowanie komputerowe

Zajęcia 2

Funkcje

- Funkcje są podstawowym składnikiem programów w C++.
- Każda funkcja jest fragmentem programu, który można używać wielokrotnie i niezależnie od pozostałych funkcji.
- Funkcje mogą mieć parametry od których zależy ich działanie.
- Funkcję mogą zwracać wynik do funkcji wywołującej.
- Programy należy dzielić na funkcje bo:
  - są krótsze
  - łatwiej je testować i poprawiać
  - łatwiej zrozumieć ich działanie

Definicja funkcji

typ nazwa(lista_parametrów) {
    ciało_funkcji
}

- typ – rodzaj wyniku zwracanego przez funkcję; jeśli `void`, to funkcja nic nie zwraca
- lista_parametrów – lista postaci (może być pusta)
  typ_1 $nazwa_1$, typ_2 $nazwa_2$, ... , typ_n $nazwa_n$
  są to parametry, o podanych typach i nazwach, od których zależy działanie funkcji
- ciało funkcji – lista instrukcji do wykonania

Wywołanie funkcji

Funkcje wywołujemy przy pomocy instrukcji wywołania:

nazwa_funkcji(parametr$_1$, parametr$_2$, … , parametr$_n$);

przy czym parametry muszą być zgodne z listą parametrów danej funkcji. Funkcja powinna być zdefiniowana przed wywołaniem. Wywołanie funkcji powoduje:

- wstrzymanie wykonanie obecnie wykonywanej funkcji,
- przekopiowanie wartości podanych w nawiasach na zmienne funkcji wywoływanej,
- wykonanie funkcji wywoływanej,
- powrót do miejsca wywołania i ew. przekazanie wartości.

Przykłady definicji i wywołań funkcji

- `void f() { … }` // funkcja f bez parametrów, nie zwraca wyniku
- `int a(double x) { … }`
  // funkcja `a` z jednym parametrem typu `double`, zwraca wartość typu `int`
- `void b(int a, char x, double y) { … }`
  // trzy parametry różnych typów, nie zwraca wyniku

Przykłady wywołań:

- `a(3.5);` // dobrze
- `a(5);` // też dobrze, `5` typu `int` zostanie zamienione na `5.0` typu `double`
- `a("ABC");` // źle, zły typ parametru
- `b(2, 'Q');` // źle, za mało parametrów

Instrukcja `return`

W każdej funkcji może znajdować się instrukcja `return`, być może więcej niż jedna. Jej wykonanie powoduje natychmiastowe zakończenie działania funkcji i powrót do miejsca jej wywołania (w funkcji `main` – zakończenie programu). Postać instrukcji zależy od typu zwracanego przez funkcję:

- Jeśli typ funkcji to `void` (tj. funkcja nie zwraca wartości), to piszemy
  `return;`
- Jeśli typ jest inny, to piszemy
  `return` wartość`;`
  gdzie wartość zostaje zwrócona jako wynik działania funkcji. Musi być ona zgodna z typem funkcji. Wykonanie funkcji, która zwraca wartość **musi** kończyć się instrukcją `return` (wyjątek: funkcja `main`).

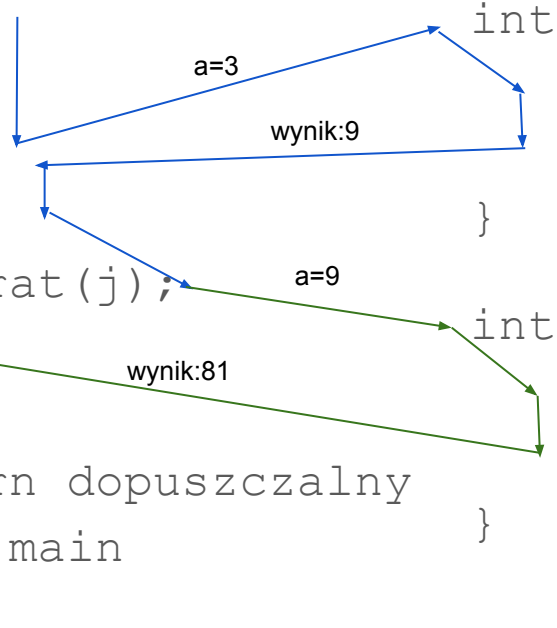
Wywołanie funkcji – przykład

```
int main() {
    int j;
    j=kwadrat(3);
    cout << j;
    int k = kwadrat(j);
    cout << k;

    // brak return dopuszczalny
    // w funkcji main
}
```

```
int kwadrat(int a) {
    int b=a*a;
    return b;
}
```

```
int kwadrat(int a) {
    int b=a*a;
    return b;
}
```

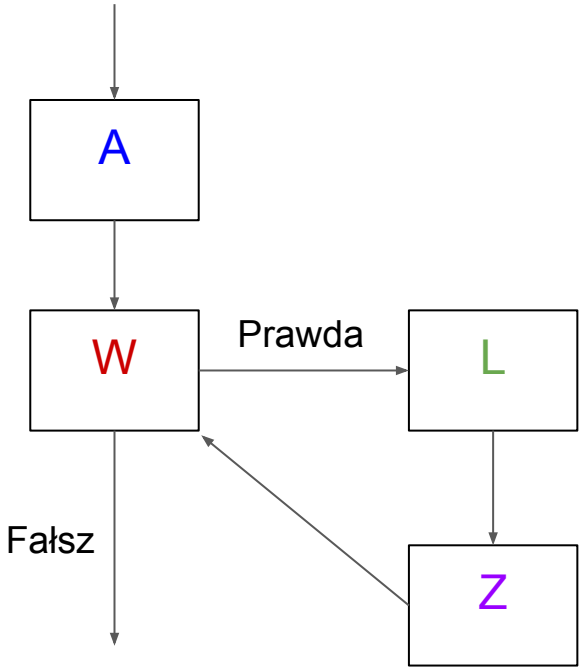


Instrukcja pętli `for`

Składnia:

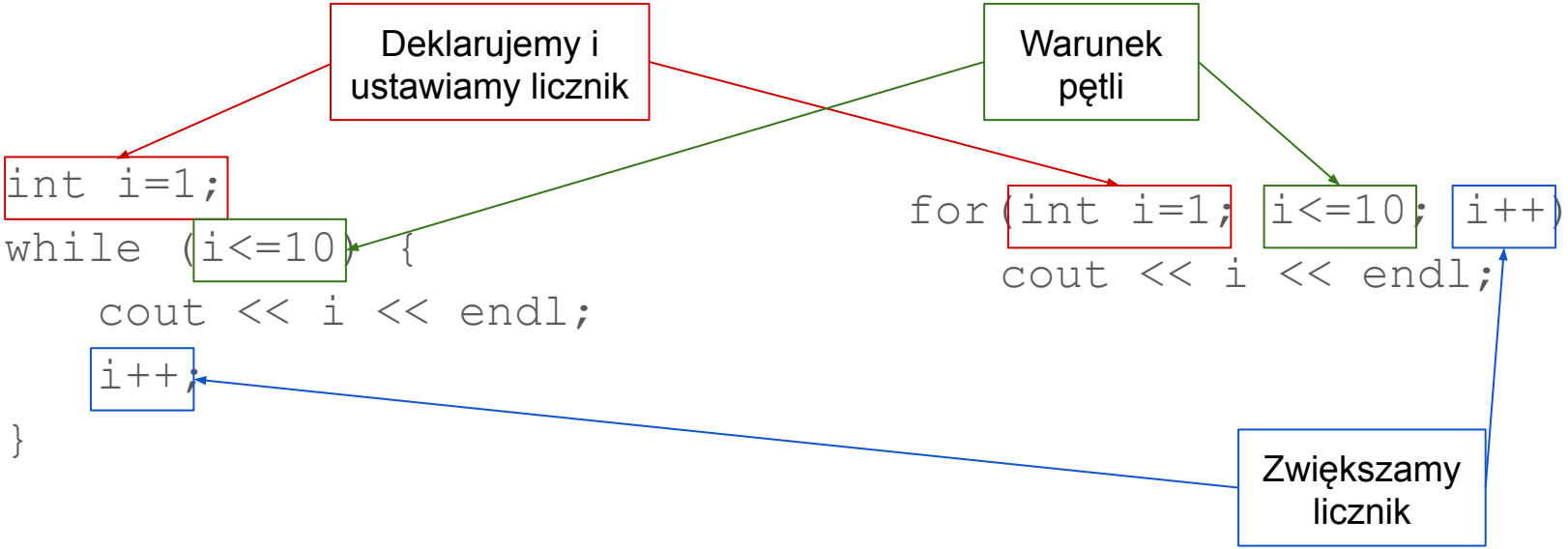
```
for(A; W; Z) {
    L
}
```

- A – wyrażenie, może być deklaracja,
- W – warunek logiczny
- Z – wyrażenie
- L – lista instrukcji (klamerki – jak w if i while)
- A, W, Z i L mogą być puste



Instrukcja pętli `for` (2)

Instrukcja `for` pozwala na zapisanie pętli w bardziej zwarty sposób niż `while`:

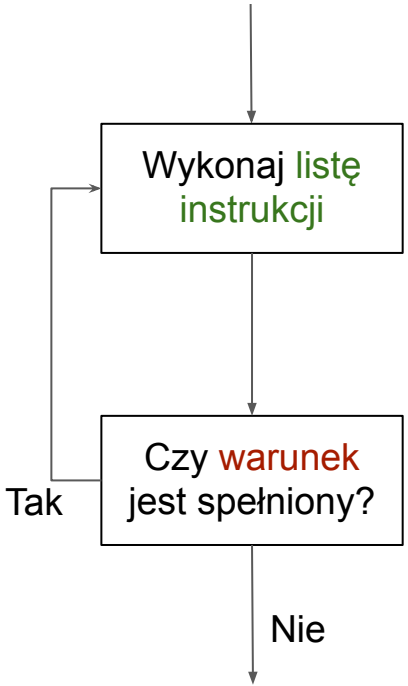


Instrukcja pętli `do`

Składnia:

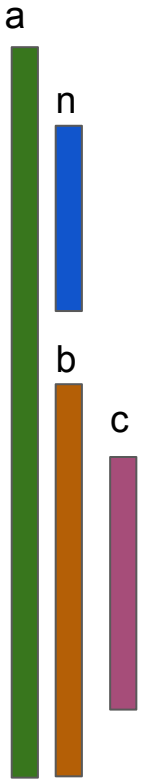
```
do {
    lista_instrukcji
} while(warunek);
```

Działa podobnie jak while, ale lista instrukcji jest wykonywana przed sprawdzeniem warunku.



Zasięg zmiennych

- Zasięg zmiennej to część programu, w którym ta zmienna jest widoczna i może być używana.
- Zasięg zaczyna się w miejscu deklaracji zmiennej i kończy w miejscu zakończenia bloku w którym zmienna jest zadeklarowana.
- Należy dbać o to, aby zasięg zmiennej nie był większy niż to konieczne.




```
int a=3;
void f(int n) {
    return n*n;
}
int main() {
    int b=20;
    for(int c=a;c<b;c++){
        cout << f(c);
        b=b-1;
    }
}
```

Zadania

Napisać funkcje:

1. `int wieksza(int a, int b)`
   która zwraca większy z parametrów.
2. `int najwieksza(int a, int b, int c)`
   która zwraca największy z parametrów.
3. `void ciag(int p, int r, int n)`
   która drukuje n-elementowy ciąg arytmetyczny o początkowym wyrazie p i różnicy r.
4. `int pytajOWiek()`
   która pyta użytkownika ile ma lat i zwraca odpowiedź, chyba, że ta będzie ujemna – wówczas ponawia pytanie, aż do skutku.

Zadania (2)

5. `double potega(double x, int d)`
   która oblicza x w potędze d (d może być ujemne).
6. `int silnia(int n)`
7. Funkcja exp może być obliczona przy pomocy wzoru
   $\exp(x)=1+x+x^2/2!+x^3/3!+x^4/4!+...$
   Napisać funkcję
   ```
   double e(double x)
   ```
   która oblicza przybliżoną wartość exp(x).
   Podpowiedź: nie używać funkcji `potega` i `silnia`.
8. `int sumaDzielnikow(int n)` // suma dzielników właściwych, np. `sumaDzielnikow(10)`=1+2+5=8

Zadania (3)

9. Liczby całkowite a i b są zaprzyjaźnione jeśli
   `a==sumaDzielnikow(b)` i `b==sumaDzielnikow(a)`
   Napisać funkcję, która znajduje wszystkie pary liczb zaprzyjaźnionych nie większe niż parametr.
10. Napisać funkcję
    `void drukuj(int n, int p)`
    która drukuje liczbę `n` w systemie pozycyjnym o podstawie `p`. Może być wspak.