Programowanie komputerowe

Zajęcia 3

Instrukcje przypisania

Poza zwykłą instrukcją przypisania, powodującą ustawienie wartości zmiennej na podane wyrażenie, istnieje wiele innych, np.

- `+=` dodaj, `a+=b` jest równoważne `a=a+b`
- `-=` odejmij, `a-=b` jest równoważne `a=a-b`
- `*=` pomnóż, `a*=b` jest równoważne `a=a*b`
- i wiele innych tworzonych analogicznie.

Pozwalają one na skrócenie zapisu programu.

Operator warunkowy `?`

Składnia:

```
w ? a : b
```

- `w` – warunek logiczny
- `a`, `b` – wyrażenia, powinny być tego samego typu

Wartością takiego wyrażenia jest `a` jeśli warunek `w` jest spełniony i `b` w przeciwnym przypadku.

Przykład: wartością wyrażenia

```
a>b ? a : b
```

jest większa z liczb a,b.

Instrukcje `break` i `continue`

Instrukcje `break` i `continue` służą do przerwania pętli.

- `break` powoduje przerwanie pętli i przejście do instrukcji za pętlą. Można też zastosować break do przerwania instrukcji `switch`.
- `continue` powoduje zaniechanie wykonania instrukcji w pętli poniżej. Program przechodzi do sprawdzenie warunku pętli; jeśli ten jest spełniony, wykonuje się kolejny obieg pętli.
- Instrukcji tych należy w miarę możliwości unikać.

Instrukcja warunkowa `switch`

Instrukcja ta jest rzadko stosowana, ale jest wygodniejsza niż `if` jeśli musimy wybrać jeden z wielu wariantów. Składnia

```
switch(a) {    // a – wyrażenie typu int lub char (ew. inny całkowity)
    lista instrukcji; może zawierać klauzule
        case wartość:  // każda wartość musi być stałą typu takiego jak a
        default:    // tylko jedna dozwolona
        break;
}
```

Lista instrukcji jest wykonywana od klauzuli `case` z wartością równą a (jeśli taka jest) lub `default` (jeśli nie ma odpowiedniego `case`) do napotkania `break`.

Instrukcja warunkowa `switch` – przykład

```
int main() {
    char c;
    cout << "Tak czy nie?";
    cin >> c;
    switch(c) {
    case 'T': case 't': case 'y': case 'Y':
        cout << "TAK"; break;
    case 'N': case 'n':
        cout << "NIE"; break;
    default:  cout << "???";
    }
}
```

Rekursja

- Rekursja to technika programowania polegająca na wywoływaniu funkcji przez siebie samą.
- Każdą funkcję, którą można zapisać za pomocą iteracji można zapisać za pomocą rekursji i na odwrót.

Przykład: Funkcję silnia można zdefiniować na dwa sposoby:

- n!=1*2*3*...*n // iteracja
- n!=n*(n-1)! dla n>0 oraz 0!=1 // rekursja

Funkcja silnia na dwa sposoby

Iteracja:

```
int silnia(int n) {
    int w=1;
    for(int i=1; i<=n; i++)
        w=w*i;
    return w;
}
```

Rekursja:

```
int silnia(int n) {
    if(n>0)
        return n*silnia(n-1);
    return 1;
}
```

Ćwiczenia

1. Napisać funkcję
   `int nwd(int a, int b)`
   która oblicza największy wspólny dzielnik dwóch liczb.
2. Napisać funkcje `ciag`, `pytaj` i `potega` z poprzednich ćwiczeń używając rekursji.
3. Napisać dwie funkcje
   `void rozklad(int n)`
   które drukują rozkład liczby na czynniki pierwsze. W jednej użyć iteracji, w drugiej rekurencji.

Ćwiczenia (2)

4. Napisać funkcję
   `int dwumian(int n, int k)`
   która oblicza dwumian Newtona.
5. Napisać funkcję
   `int pierwiastki(double a, double b, double c)`
   która drukuje pierwiastki wielomianu ax^2+bx+c. Wynikiem powinna być liczba pierwiastków.
6. Napisać funkcję
   `int najmniejszyNieDzielnik(int n)`
   która zwraca najmniejszą liczbę naturalną, która nie jest dzielnikiem `n`.

Ćwiczenia (3)

7. Napisać funkcję

   `int sumaKwadratow(int n)`

   która zwraca sumę kwadratów liczb 1 .. n.
8. Napisać funkcję

   `void trojkatyProstokatne(int n)`

   która wypisuje na ekranie wszystkie trójki liczb (a,b,c) takie że 0<a<b<c<n oraz $a^2+b^2=c^2$.