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;
}
```

Referencje

Jeśli T jest pewnym typem, to T& jest typem referencyjnym do T. Wartością typu referencyjnego jest **zmienna** typu T.

Typu referencyjnego możemy użyć jako parametru funkcji. Wówczas nie powstaje kopia wartości, a funkcja wywoływana używa zmiennej zadeklarowanej w innej funkcji. Robimy tak jeśli:

- chcemy zmieniać zmienne z funkcji wywołującej,
- nie chcemy (lub nie możemy) tworzyć kopii istniejącej zmiennej.

Przekazanie parametru za pomocą referencji nazywamy przekazaniem **przez zmienną**, a w zwykły sposób przekazaniem **przez wartość**.

Referencje – przykład

Poniższa funkcja zamienia wartości dwóch zmiennych:

```
void zamien(int& a, int & b) {
    int c=a;
    a=b;
    b=c;
} // Przykład użycia poniżej. Co się stanie jeśli pominiemy znaczki & ?
int main() {
    int x=3, y=7;
    zamien(x,y);
    cout << "x=" << x << ", y=" << y << endl;
}
```

Tablice

- Tablice służą do przechowywania wielu zmiennych tego samego typu.
- Tablicę deklaruje się następująco:

  typ nazwa[liczba_elementów];

  Tak zadeklarowana tablica zawiera elementy
  nazwa[0], nazwa[1], nazwa[2], … , nazwa[liczba_elementów-1],
  wszystkie one są typu podanego w deklaracji.
- Liczba elementów tablicy musi być stałą znaną w momencie kompilacji programu.
- Poszczególne elementy tablicy zachowują się jak zwykłe zmienne.
- Można ustawić elementy tablicy w momencie deklaracji, np. tak:

```
int t[5]={3,7,-1,2,4};
```

Tablice jako parametry funkcji

- Tablice są zawsze przekazywane przez zmienną – funkcja nie działa na kopii tylko na oryginalnej tablicy.
- Nie jest przekazywany rozmiar tablicy; żeby go przekazać należy użyć dodatkowego parametru, np.

  ```
  int f(int tablica[], int rozmiar) { … }
  ```
- Innym sposobem jest przekazanie **wskaźnika** do tablicy, np. tak:

  ```
  int f(int* tablica, int rozmiar) { … }
  ```

  O wskaźnikach – później.
- Napisy są zapamiętywane jako tablice znaków (`char`).

Ćwiczenia

1. Napisać funkcję
   `int nwd(int a, int b)`
   która oblicza największy wspólny dzielnik dwóch liczb.
2. Napisać funkcję
   `void ciag(int p, int r, int n)`
   która drukuje ciąg arytmetyczny o początkowym wyrazie p, różnicy r i n wyrazach. Nie używać pętli.
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
   `void drukuj(int t[], int r)`
   która drukuje elementy tablicy t o długości r na ekranie.
5. Napisać funkcję
   `int max(int t[], int r)`
   która zwraca największy element tablicy t o długości r.
6. Napisać funkcję
   `int suma(int t[], int r)`
   która zwraca sumę elementów tablicy t o długości r.

Zadanie domowe

- Zadanie domowe znajduje się pod adresem http://www.mimuw.edu.pl/~ziemians/pk2016/zd1.cpp
- Należy uzupełnić funkcje zastępując części oznaczone `/*(  )*/`
- Rozwiązania odesłać na adres kziemianski.edu@gmail.com w terminie ustalonym na zajęciach.
- W temacie e-maila wpisać pk1-XXXXX, gdzie XXXXXX jest numerem indeksu.