Semelbauer Ștefan

VARIABILE

Introducere

Pentru a putea prelucra informațiile pe care le dăm calculatoarelor, ele au nevoie de o metodă de a le stoca. Odată stocate, informațiile pot fi accesate și modificate, iar pe urmă stocate din nou. De exemplu după ce facem o poză pe smartphone, o putem edita și să salvăm noua variantă.

Un alt exemplu este atunci când trimiți un mesaj sau un email. Acesta este stocat în mesaje trimise până când decizi să îl ștergi.

În programele pe care le scriem avem nevoie de o modalitate ușoară de a reține informație pe care să o putem prelucra. Înainte să stocăm informație, trebuie să informăm calculatorul în legătură cu tipul de informație pe care vrem să o stocăm. Făcând aceasta, îi spunem că trebuie să pregătească memorie pentru a stoca informația.

O **variabilă** ne ajută să facem toate aceste lucruri atunci când scriem programe. Pe parcursul execuției programului putem schimba informația reținută de către variabilă.

Pentru început vom lucra cu variabile care stochează doar numere întregi (numere pozitive și negative fără virgulă).

TIPUL DE DATE PENTRU NUMERE ÎNTREGI

Exemplu

Pentru a informa calculatorul că urmează să stocăm numere, putem proceda în felul următor:

```
#include <iostream>
using namespace std;

int main() {
        int variabila1;
        int variabila2, alta_variabila;
        return 0;
}
```

Declarare

Pentru a informa calculatorul că urmează să folosim variabile de tip întreg, este necesar să adăugăm instrucțiuni în codul sursă care să facă acest lucru. O astfel de instrucțiune se numește declarare de variabilă.

Pentru a declara variabile este necesar să precizăm tipul variabilelor și pe urmă să spunem cum se vor numi variabilele.

În cazul de mai sus `int` este tipul variabilelor (număr întreg), iar `variabila1`, `variabila2` și `alta_variabila` sunt numele variabilelor.

Când folosim o instrucțiune de declarare, putem declara o singură variabilă sau mai multe, caz în care trebuie să separăm numele variabilelor prin virgule. Deși nu este obligatoriu, majoritatea programatorilor pun spațiu după virgulă când declară variabile deoarece face codul sursă mai ușor de citit și înțeles.

La finalul instrucțiunii de declarare trebuie să punem caracterul punct virgulă `;`

Pentru fiecare variabilă declarată, calculatorul va aloca un spațiu în memorie pentru a o stoca.

Denumirea variabilelor

Pentru a putea fi identificate mai ușor pe măsură ce scriem programul, în limbajul C++ trebuie să dăm un nume variabilei. Un nume poate fi format din litere

mici și mari ale alfabetului englez, numere și caracterul underscore (_), dar nu pot să înceapă cu o cifră și nu au voie să conțină spațiu.

În plus este foarte indicat să nu folosești denumiri rezervate ca nume de variabile. De exemplu nu poți să ai o variabilă numită `int`.

Exemple

```
int variabila_mea, alta_variabila, litereMariSiMici;
int a, b, x, yz, qwerty;
int a1, b23, q99werty;
```

Capacitate

Întrucât memoria calculatorului este limitată, numerele pe care le putem păstra în variabile au un număr limitat de cifre. Pentru `int` putem să ne gândim că nu putem stoca numere mai mici decât `-2 000 000 000` și nici numere mai mari decât `2 000 000 000`. Aceasta e o aproximare, intervalul real fiind puțin mai mare, între `-2 147 483 648` și `2 147 483 647`.

De exemplu oricare dintre numerele `-2`, `0`, `205`, `100003` poate fi stocat într-un int, în schimb `-2 147 483 649` sau `1 000 000 000 000` nu pot fi stocate într-o variabilă de tip `int`.

ATRIBUIREA VALORILOR

Acum că știm cum să declarăm variabile, este timpul să învățăm cum să stocăm valori în ele.

```
#include <iostream>
using namespace std;

int main() {
    int a, b, c;
    a = 50;
    c = -32;
    b = 11;
    return 0;
}
```

Pentru a stoca o valoare într-o variabilă vom folosi operatorul `=` (egal). În stânga lui vom scrie numele variabilei, iar în dreapta vom scrie valoarea pe care dorim să o punem în variabilă.

Pe lângă valori numerice, în variabile putem stoca valoarea altor variabile.

De exemplu

```
#include <iostream>
using namespace std;

int main() {
    int a, b;
    b = 3;
    a = b;
    return 0;
}
```

În exemplul de mai sus, variabila `a` va lua valoarea `3` după instrucțiunea `a = b;`

Și în cazul instrucțiunii de atribuire, trebuie să punem la finalul ei `;` (punct virgulă).

Atenție!

Atunci când atribuim unei variabile valoarea unei alte variabile, se va lua în calcul valoarea variabilei atribuite **la momentul atribuirii**.

În cazul următor, variabila `a` va lua valoarea `3`, iar valoarea lui a nu se va mai schimba.

```
#include <iostream>
using namespace std;

int main() {
    int a, b;
    b = 3;
    a = b;
    b = 5; // Valoarea lui a ramane 3, doar b se schimba in 5!
    return 0;
}
```

Putem inițializa variabilele și în momentul declarării:

În exemplul de mai jos, variabila `a` va lua valoarea `5` și `b` va lua valoarea `2`.

Pe parcursul execuției programului, o variabilă poate lua mai multe valori, o variabilă fiind de fapt un spațiu unde stocăm un număr, iar numărul poate fi schimbat de oricâte ori.

```
#include <iostream>
using namespace std;

int main() {
    int a = 5, b = 2;
    a = b;
    return 0;
}
```

Deși la început `a` are valoarea `5`, după instrucțiunea `a = b`, `a` va avea valoarea `2`.

Atenție

În cazul în care nu atribuim o valoare unei variabile ea va rămâne neinițializată și va avea în ea o valoare arbitrară. De aceea este bine să inițializăm variabilele de fiecare dată înainte să le folosim.

CITIREA DE LA TASTATURA A VARIABILELOR

În programele scrise până acum puteam atribui valori variabilelor doar prin scrierea lor în codul sursă. Astfel la fiecare execuție a programului, variabilele iau aceeași valoare. În plus, utilizatorii programelor nu au acces la codul sursă, deci nu ar putea să modifice valorile din variabile.

De obicei vrem să folosim programe care să producă rezultate personalizate în funcție de datele introduse de fiecare utilizator. De exemplu o aplicație de calculator nu ar fi prea utilă dacă la fiecare pornire a ei ar afișa pe ecran `5`.

Pentru a putea "personaliza" rezultatul execuției unui program, putem să inițializăm variabilele cu valori introduse de la tastatură de către utilizator.

```
#include <iostream>
using namespace std;

int main() {
    int a,b,c;
    cin>>a;
    cin>>b>>c;
    return 0;
}
```

Folosind `cin` putem citi de la tastatură numere cu care vor fi inițializate variabilele.

Practic `cin>>a;` face următoarele lucruri

```
1. citește de la tastatură un număr introdus de către utilizator
2. stochează în variabila "a" numărul introdus de către utilizator
```

Putem citi una sau mai multe valori pe care să le punem în variabile. Instrucțiunea `cin>>a;` citește de la tastatură variabila `a`, iar `cin>>b>>c;` citește variabilele `b` și `c`. Putem citi oricâte variabile dorim folosind `cin`. Pentru a face asta trebuie să punem `>>` înainte de fiecare variabilă pe care dorim să o citim. De asemenea e important să nu uităm să punem `;` (punct virgulă) la final.

În momentul execuției programului, calculatorul va aștepta pentru fiecare variabilă în care vrem să punem o valoare să introducem câte un număr, iar după el

să apăsăm tasta Enter. De exemplu dacă avem `cin>>a>>b` putem introduce `12` apăsăm Enter, introducem `5`, apăsăm Enter.

AFISAREA VARIABILELOR

Pentru a afișa valorile variabilelor pe ecran vom folosi `cout` într-un mod similar cu `cin` pentru citirea variabilelor de la tastatură.

Dacă pentru a afișa mesaje pe ecran este nevoie să folosim `"` (ghilimele), pentru a afișa o variabilă, trebuie doar să scriem numele ei.

La fel ca și în cazul citirii, înainte să afișăm o variabilă trebuie să folosim un simbol. În cazul `cout` simbolul folosit este `<<`.

Pentru a înțelege, cel mai ușor este să execuți următoarele exemple în zona de la finalul lecției.

```
#include <iostream>
using namespace std;

int main() {
        int a=12,b=56;
        cout<<b<<a;
        return 0;
}
```

În cazul acestui exemplu se afișează pe ecran două variabile, prima variabilă afișată având valoarea `56`, iar a doua valoarea `12`. Întrucât nu am afișat spații între ele, pe ecran va apărea `5612`.

```
#include <iostream>
using namespace std;

int main() {
        int a=12,b=56;
        cout<<a<<" "<<b;
        return 0;
}
```

Exemplul acesta este similar cu exemplul anterior, singura diferență este că în acest caz afișăm un spațiu între variabile. Textul afișat pe ecran va fi `12 56`.

```
#include <iostream>
```

```
using namespace std;

int main() {
        int mere=3;
        cout<<"Ana are " <<mere<<" mere!";
        return 0;
}
```

În acest exemplu afișăm text înainte și după o variabilă. Textul afișat va fi `Ana are 3 mere!`. De fiecare dată când folosim `<<nume_variabilă` pe ecran se va afișa valoarea variabilei `nume_variabilă`.

Exemplu cu citire și afișare

Înainte de a trece mai departe, execută următorul program de mai multe ori și introdu diverse numere la tastatură pentru a înțelege cum funcționează citirea și afișarea!

```
#include <iostream>
using namespace std;

int main() {
        int a2, a1;
        cin>>a2>>a1;
        cout<<"prima variabila citita are valoarea: "<<a2;
        cout<<"\na doua variabila citita are valoarea: "<<a1;
        return 0;
}
```

De fiecare data când scriem `\n` între ghilimele, următoarele caractere din afișare se vor afișa pe linia următoare.

Încearcă și tu

Copiază și modifică exemplele de mai sus pentru a înțelege mai bine cum funcționează variabilele și instrucțiunile `cin` și `cout`. Pentru a introduce numere apasă pe triunghiul gri, scrie numerele în zona închisă la culoare și apasă enter pe tastatură.

COMPILATOR ONLINE