# Instructiunile limbajului C.

**Responsabili:**

- Emil Racec (2012) [mailto:emil.racec@gmail.com]
- Octavian Rînciog (2010) [mailto:octavian.rinciog@gmail.com]

## Obiective

În urma parcurgerii acestui laborator studentul va fi capabil să:

- scrie programe C utilizând structuri condiţionale si de repetiţie

## Noţiuni teoretice

### Instrucţiuni condiţionale

If-else

`if…else` este cea mai simplă instrucţiune condiţională. Poate fi folosită în mai multe forme:

```
if (condition) {
  // instructions
  //...
}
```

```
if (condition) {
  // instructions
  // ...
} else {
  // other instructions
  // ...
}
```

```
if (condition1) {
  ...
} else if (condition2) {
  ...
}
  ...
else if (conditionN) {
  ...
}
```

Instrucţiunea evaluează expresia `condition` şi execută instrucţiunile dintre acolade doar dacă rezultatul este nenul. În varianta cu `else`, pentru rezultat nul este executat blocul de instrucţiuni aflat după `else`.

În al treilea caz, sunt evaluate pe rând condiţiile şi este executat blocul corespunzător primei condiţii adevărate. Un exemplu de folosire este:

```
if (a == b) {
  printf("Numerele sunt egale");
} else if (a > b) {
  printf("A este mai mare");
}
```

Switch

`switch` este o instrucţiune menită să simplifice structurile condiţionale cu mai multe condiţii.

```
switch (expression) {
  case constant1:
    // instructions1
  case constant2:
    // instructions2
  ...
  default:
    // instructions
}
```

Valoarea `expression` este evaluată la un tip intreg, apoi această valoare este comparată cu fiecare constantă; este rulat blocul de instrucţiuni al valorii găsite. În caz ca numărul nu este egal cu nici una dintre constante, este executat blocul aflat după `default`.

După executarea ultimei instrucţiuni dintr-un bloc `case`, execuţia nu continua după blocul `switch`, ci la inceputul următorului bloc `case` *`constant`* sau `default`. Pentru a ieşi din blocul `switch`, se foloseşte instrucţiunea `break`.

```
int main() {
  char c;
  printf("Alegeți o opțiune:\n\t[a] afișare\n\t[s] ștergere\n\t[e] ieșire\n");
  scanf("%c", &c);
  printf("Ați ales: ");

  switch (c) {
    case 'a':
      printf("afișare");
      break;

    case 's':
      printf("ștergere");
      break;

    case 'e':
      printf("ieșire");
      break;

    default:
      printf("O opțiune inexistentă");
      break;
  }
  return 0;
}
```

```
int main() {
  int n, n2;
  printf("Introduceți o valoare între 0 și 5:");
  scanf("%d", &n);
  n2 = 1;
  switch (n) {
    case 5:
      n2 *= 2;
      /* fără break, continuă la următoarea instrucțiune */
    case 4:
      n2 *= 2;
    case 3:
      n2 *= 2;
    case 2:
      n2 *= 2;
    case 1:
      n2 *= 2;
    case 0:
      printf("2 la puterea %d este %d\n", n, n2);
      break;

    default:
      printf( "Valoare invalidă\n" );
  }
  return 0;
}
```

## Instrucţiuni de repetiţie

while

`while` execută un bloc de instrucţiuni atâta timp cât o anumită condiţie este adevărată. Forma generală a unui ciclu while este:

```
while (expression) {
  // instructions
}
```

Câtă vreme `expression` are o valoare nenulă, instrucţiunile din blocul de după `while` sunt executate. Expresia este reevaluată după fiecare ciclu. Un astfel de ciclu poate să se execute o dată, de mai multe ori sau niciodată, în funcţie de valoarea la care se evaluează expresia.

do ... while

`do … while` este o instrucţiune repetitivă similara cu cea precedentă, singura diferenţa fiind că expresia este evaluată după executarea instrucţiunilor, nu înainte. Astfel, blocul va fi executat cel puţin o dată.

```
do {
  // instructions
} while (expression);
```

for

`for` reprezintă o formă mai simplă de a scrie un while însotit de o expresie iniţiala şi de o expresie de incrementare. Forma sa este:

```
for (expression1; expression2; expression3) {
  // instructions
}
```

Secvenţa de cod de mai sus este echivalentă cu:

```
expression1
while (expression) {
  instructions
  expression
}
```

În cazul instrucţiunii `for`, oricare dintre cele 3 expresii poate lipsi. Lipsa expresiei condiţionale este echivalentă cu o buclă infinită, cum ar fi:

```
for ( ; ; ) {
  /* instrucţiunile de aici sunt intr-o buclă infinită */
}
```

În acest caz, ieşirea din buclă trebuie făcută explicit, cu ajutorul instrucţiunii `break`.

Exemplul următor prezintă un ciclu cu funcţionalitate identică (tipărirea primelor 10 numere naturale), folosind cele 3 instrucţiuni repetitive:

```
int main() {
  short i;
  printf("Ciclu for\n");

  for (i = 1; i <= 10; i++) {
    printf("i=%d\n", i);
  }

  printf("Ciclu while\n");
  i = 1;
  while (i <= 10) {
    printf("i=%d\n", i);
    i++;
  }
```

```
  printf("Ciclu do while\n");
  i = 0;
  do {
    i++;
    printf("i=%d\n", i);
  } while(i < 10);
}
```

Pentru blocuri de o singură instrucţiune (cum este si cazul instructiunii executate de for in exemplul de mai sus) nu este nevoie sa folosim acolade. Totusi, folosirea acoladelor, chiar si in aceasta situatie, este recomandata pentru o depanare mai usoara si o lizibilitate mai buna a programelor.

## Instrucţiuni speciale

### break

`break`, pe lângă utilizarea descrisă la instrucţiunea switch, poate fi folosită pentru a ieşi forţat dintr-o instrucţiune de repetiţie. Secventa următoare este echivalentă cu cele de mai sus:

```
i = 0;
for( ; ; ) {
  i++;
  if (i > 10) {
    break; /* ieşire forţată din bucla */
  }
  printf( "i=%d\n", i );
}
```

### continue

`continue` forţează terminarea iteraţiei curente a buclei si trecerea la iteraţia următoare. În cazul instrucţiunii `for`, acest lucru presupune executarea instrucţiunii de incrementare; apoi se evaluează condiţia de continuare a buclei. Exemplul următor demonstrează implementarea unei bucle infinite cu ajutorul instrucţiunii `continue`:

```
for (i = 0; i < 10; ) {
  if (i == 0) {
    continue;
  }
  i++;
}
```

### return

`return` este instrucţiunea de terminare a funcţiei curente. Aceasta poate fi apelată in forma `return`; în cazul funcţiilor care returnează `void` şi în forma `return` *`result`*; pentru funcţiile care întorc o valoare.

### goto

`goto` este o instrucţiune de salt a execuţiei. Instrucţiunea primeşte ca parametru o etichetă; următoarea instrucţiune executată după `goto` este cea de la eticheta dată.

```
int main() {
  goto et;
  printf("Asta nu apare la executie\n");

et:
  printf("Asta apare la rulare\n");
  return 0;
}
```

În majoritatea cazurilor, utilizarea instrucţiunii `goto` nu este recomandată şi poate fi evitată folosind alte instrucţiuni de control şi funcţii. Programele care folosesc această instrucţiune pentru a sări între secvenţe îndepărtate de cod sunt dificil de depanat şi analizat.

# Exerciţii de Laborator

1. [1.5p] Se citeşte de la tastatură un număr întreg si pozitiv N. Să se scrie un program care determină care număr cuprins intre 2 si N are suma divizorilor nebanali maximă (adică printre divizori nu sunt considerate numerele 1 si N). Dacă există mai multe asemenea numere se va afişa numai primul dintre ele.
   **Exemplu**

```
N=100
OUT: 96 //are suma divizorilor 155
```

2. [1.5p] De la tastatură se introduc mai multe numere întregi si pozitive, terminate printr-un număr negativ. După fiecare număr introdus, se va afişa lista divizorilor lui nebanali sau textul PRIM. La sfârşit se va afişa numărul de numere prime găsite.
   **Exemplu**

```
35
OUT: 5 7
36
OUT: 2 3 4 6 9 12 18
17
OUT: PRIM
2
OUT: PRIM
12
OUT: 2 3 4 6
25
OUT: 5
53
OUT: PRIM
-4
OUT: S-au găsit 3 numere prime.
```

3. [2p] Fie funcţia `f:[u,v] -> [min,max],f(x) = a * x2 + b * x + c` , în care a, b, c, u si v sunt date. Determinaţi min si max, apoi rezolvaţi ecuaţia f(x)=0 în mulţimea numerelor reale. Afişaţi doar rădăcinile din intervalul `[u,v]`.
4. [2p] Scrieţi un program care să convertească numere din baza 2 în baza 10 şi invers. Numerele se introduc de la tastatură până la intâlnirea unui număr negativ. Pentru fiecare număr introdus, se va afişa pe o linie rezultatul.
5. [3p] Scrieţi un program care verifică dacă un număr citit de la tastatură este palindrom. Un număr se consideră palindrom dacă citit invers este identic cu numărul iniţial.

## Bonus

1. [2p] De la tastatură se introduc N(fiind dat şi el de la tastatură) numere citite pe rând. Găsiţi cel mai mare divizor comun al acestor N numere, fără a folosi vectori.

# Referinţe

- C - Control Statements [http://www.tutorialspoint.com/ansi_c/c_control_statements.htm]