Limbajul de programare C

Prezentare generală a limbajului C

- C este un limbaj de programare creat la începutul anilor ’70 (1969 - 1973) de către Dennis Ritchie;
- Deși este un limbaj de nivel înalt, care respectă principiile programării structurate, arhitectura sa conține elemente foarte asemănătoare instrucțiunilor cod-mașină;
- Această proprietate a făcut ca C să fie folosit pe scară largă pentru crearea de software de bază, incluzând sisteme de operare, browsere, drivere etc.;
- C nu este numai unul dintre cele mai folosite limbaje de programare din toate timpurile, ci și cel mai influent: C#, Java, Objective C, PHP, Python și multe alte limbaje au preluat construcțiile sale de bază.

Etapele realizării unui program C

- Limbajul C este unul compilat, adică programatorul scrie instrucțiuni specifice într-un fișier text, numit fișier sursă, cu extensia .c;
- Apoi un alt program, numit compilator ”traduce” textul respectiv (numit ”cod sursă”, sau pur și simplu ”sursă”);
- Se obține un alt fișier care poate fi înțeles și executat de către sistemul de operare;
- Nu orice cod sursă compilat poate fi executat; există fișiere cu extensia .c care sunt doar folosite de către alte fișiere .c.

Structura unui program C

- Prin ”program C” vom înțelege un fișier sursă care poate fi executat;
- În general programele C conțin o primă zonă, cu așa numitele instrucțiuni de ”precompilare” – în cazul programului alăturat:

  #include <stdio.h>
- Aceste instrucțiuni indică secvențe de cod care vor fi inserate în codul sursă (fără ca programatorul să vadă codul inserat) sau înlocuiri care urmează să fie făcute în etapa de compilare;
- Biblioteca stdio.h conține funcții speciale pentru citirea și afișarea datelor – prin urmare instrucțiunea de includere a acesteia va fi practic prezentă în orice program pe care îl vom scrie în C.

```
1  #include <stdio.h>
2
3  int main()
4  {
5      printf("Hello world!\n");
6      return 0;
7  }
8
```

Structura unui program C

- Orice program C este de fapt o colecție de module, numite ”funcții” sau ”subprograme” care interacționează;
- Fiecare funcție este formată din antet și blocul de instrucțiuni, delimitat de acolade;
- Antetul conține tipul rezultatului returnat, numele funcției și lista parametrilor (care poate fi vidă), delimitată de paranteze;
- Funcția **int main()** este punctul de pornire al oricărui program;
- În lipsa acesteia, compilarea fișierului sursă nu are ca efect obținerea unui program executabil;
- Prin urmare, fiecare program pe care îl vom scrie va conține funcția **int main()** și blocul de instrucțiuni corespunzător.

```
1  #include <stdio.h>
2
3  int main()
4  {
5      printf("Hello world!\n");
6      return 0;
7  }
8
```

Structura unui program C

- Programul alăturat conține două instrucțiuni;
- Prima dintre ele:

  printf(”Hello world!\n”);

  are ca efect afișarea pe ecran a mesajului de salut corespunzător (este un program devenit standard pentru cel mai simplu program în orice limbaj);
- Prima instrucțiune **NU** este obligatorie și va lipsi probabil din următoarele noastre programe;
- A doua instrucțiune:

  return 0;

  este obligatorie la sfârșitul funcției **int main()**, prin urmare va apărea în toate programele pe care urmează să le scriem.

```
1  #include <stdio.h>
2
3  int main()
4  {
5      printf("Hello world!\n");
6      return 0;
7  }
8
```

Citirea datelor în C

- Pentru citirea datelor, în C se folosește funcția scanf:

  scanf(”%format”, &variabila);

Exemplu:

```
int x;
scanf(”%d”, &x);
```

- Pot fi citite mai multe variabile cu un singur apel al funcției scanf, chiar dacă variabilele sunt de tipuri diferite:

  scanf(”%format1%format2”, &var1, &var2);

Exemplu:

```
int x;
double y; // variabila de tip real cu formatul lf
scanf(”%d%lf”, &x, &y);
```

- Lista completă a formatelor pentru citire și afișare cu funcții din biblioteca stdio.h urmează să fie studiate ulterior, odată cu tipurile de date din limbajul C.

Afișarea datelor în C

- Pentru afișarea datelor, în C se folosește funcția printf:

  printf(”%format”, expresie);

Exemplu:

```
int x;
...
printf(”%d”, x+2);
```

- Pot fi afișate expresii complexe, care să conțină și mesaje de tip text cu un singur apel al funcției printf:

  scanf(”text0%format1text1%format2text2”, expr1, expr2);

Exemplu:

```
int x;
double y; // variabila de tip real cu formatul lf
...
scanf(”x este egal cu %d, iar y+1 =%lf”, x, y+1);
```

Declararea variabilelor în C

- Spre deosebire de scheme logice și pseudocod, în C fiecare variabilă trebuie să fie declarată înainte de a fi folosită;
- Declararea presupune precizarea tipului și numelui variabilei:

  tip numeVariabila;
- Declararea are ca efect rezervarea unei zone de memorie (în memoria internă) care va fi identificată cu ajutorul numelui variabilei;
- Tipul variabilei va determina dimensiunea memoriei alocate și operațiile permise (de exemplu, pentru o variabilă de tip real, nu va fi permisă operația ”%” care determină restul împărțirii întregi);
- Tipul poate fi unul simplu, predefinit (int, char, float, double etc.) sau unul definit anterior în cadrul programului;
- Numele variabilei trebuie să fie format din litere, cifre și caracterul underscore (_) și trebuie să nu înceapă cu o cifră

Instrucțiunea de atribuire în C

- Atribuirea are în C următoarea formă:

  variabila = expresie;

Exemplu:

```
int x;
x = 18;
```

- De fapt în C atribuirea nu este o instrucțiune propriu-zisă ci o expresie (care poate face parte din alte expresii) – acesta e un aspect asupra căruia urmează să revenim.

Instrucțiunea de decizie (alternativă) în C

- Echivalentul din C pentru structura alternativă din pseudocod este pur și simplu o traducere în engleză a acesteia (”dacă” se traduce prin ”if” și ”altfel” prin ”else”);
- În plus, condiția va fi totdeauna plasată între paranteze ; ca și în cazul pseudocodului, ramura ”else” poate lipsi

```
if (condiție)
{
        instrucțiuni1
}
else
{
        instrucțiuni2
}
```

Exemplu:

```
int x;
scanf (”%d”, &x);
if (x%2==0)
{
        printf(”%d este par”, x);
}
else
{
        printf(”%d este impar”, x);
}
```

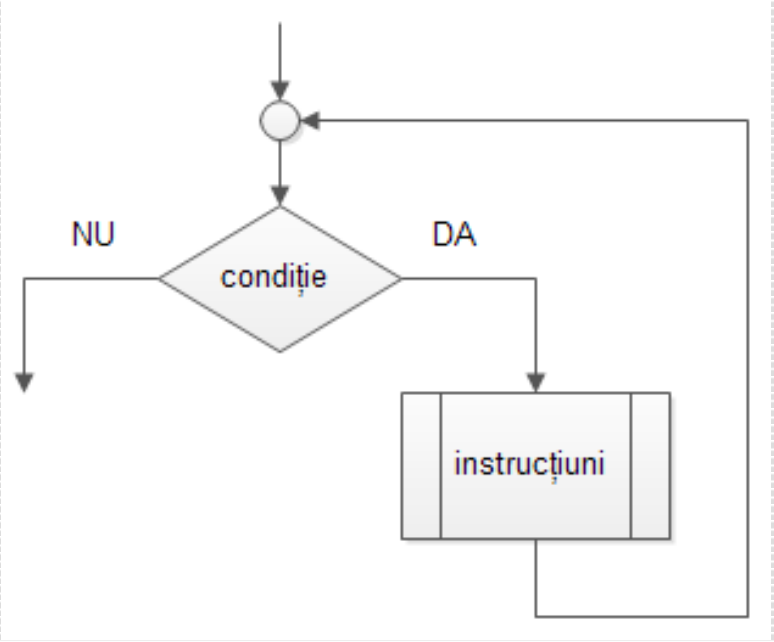
Instrucțiunea repetitivă ”while”

- Așa cum ”if” este pur și simplu o traducere a lui ”dacă” din pseudocod, instrucțiunea C ”while” este o traducere a lui ”cât timp”;
- Secvența pseudocod de mai jos

  cât timp condiție
  {
  instrucțiuni
  }

  corespunzătoare schemei logice alăturate se scrie în C în felul următor:

  while (condiție)
  {
  instrucțiuni
  }



- ”while” nu este singura structură repetitivă, dar orice algoritm poate fi exprimat cu ajutorul ei;
- Celelalte structuri și instrucțiuni repetitive urmează să fie studiate în viitor

Exemple de algoritmi pseudocod și C echivalenți

- Algoritmul de determinare a maximului a n numere are în pseudocod forma de mai jos, iar în C cea din imaginea alăturată:

```
citește n
citește a
max ← a
n ← n – 1
cât timp n > 0
{
        citește a
        dacă a > max
        {
                max ← a
        }
        n ← n -1
}
scrie max
```

```c
#include <stdio.h>

int main()
{
    int n, a, max;
    scanf( "%d", &n );
    scanf( "%d", &a );
    max = a;
    n = n - 1;
    while( n > 0 )
    {
        scanf( "%d", &a );
        if ( a > max )
        {
            max = a;
        }
        n = n - 1;
    }
    printf( "%d", max );
    return 0;
}
```

Exemple de algoritmi pseudocod și C echivalenți

- Algoritmul care verifică dacă un șir de n numere este ordonat crescător are în pseudocod forma de mai jos, iar în C cea din imaginea alăturată:

```
citește n
citește p
cr ← 1
n ← n – 1
cât timp n > 0
{
        citește u
        dacă u < p
        {
                cr ← 0
        }
        n ← n -1
        p ← u
}
dacă cr = 1
{
        scrie “DA”
}
altfel
{
        acrie “NU”
}
```

```c
#include <stdio.h>

int main()
{
    int n, p, u, cr;
    scanf( "%d", &n );
    scanf( "%d", &p );
    cr = 1;
    n = n - 1;
    while( n > 0 )
    {
        scanf( "%d", &u );
        if ( u < p )
        {
            cr = 0;
        }
        n = n - 1;
        p = u;
    }
    if( cr == 1 )
    {
        printf( "DA" );
    }
    else
    {
        printf( "NU" );
    }
    return 0;
}
```