



Discipline : Informatique (Algorithmes et Programmation)

Classe : XD

Niveau : B1

Thème du programme roumain : Les boucles prétestées et posttestées

Ressources documentaires et références :

- <http://www.pise.info/algo/boucles.htm>
- <http://carl.seleborg.free.fr/cpp/cours/chap1/boucles2.html>
- manuel scolaire

Objectifs :

- expliquer le principe du fonctionnement des instructions répétitives "while" et "do..while" ;
- transcrire correctement les boucles du langage pseudo-code dans le langage C++ ;
- identifier les situations où on doit appliquer les instructions répétitives ;
- élaborer d'algorithmes qui utilisent des boucles.

Tâches à compléter :

- écrire un algorithme pseudo-code avec des boucles à partir d'une situation concrète ;
- élaborer un programme C++ à partir du pseudo-code ;
- implémenter le programme C++ sur un ordinateur, dans un EDI (Environnement de Développement Intégré) ;

Termes à expliquer :

Français	Roumain	Explications
boucle	buclă, instrucțiune repetitivă	în lb. franceză se poate spune deoparte "instruction repetitivă", însă "boucle" e utilizată mai des
répéter .. jusqu'à	repetă..până când	este o structură repetitivă standard a limbajului pseudo-cod
tant que .. faire	cât timp .. execută	este o structură repetitivă standard a limbajului pseudo-cod

Vocabulaire :

Français	Roumain
boucle prétestée	repetitivă cu test inițial
boucle posttestée	repetitivă cu test final
facteur premier	factor prim
moyenne arithmétique	medie aritmetică
quotient, reste	cât, rest

Mots clés : informatique, boucles, répétitives

Déroulement :

Le professeur révisé les notions liées aux boucles dans le langage pseudo-code : boucles prétestées et posttestées et la modalité de transcription dans le langage C++.

Parcours :

Distribuer la fiche élève avec les énoncés des exercices.

Activité 1 :

- Préciser les particularités de transcription, telles que les instructions groupées en bloc, la boucle posttestée, la déclaration des variables.

- Piste de correction :

```
#include<iostream.h>
void main()
{
    long n,i,s;
    cin>>n;
    i=1;
    s=0;
    while (i<=n)
    {
        s=s+i;
        i++;
    }
    cout<<s;
}

#include<iostream.h>
void main()
{
    long nr;
    cin>>nr;
    do
    {
        cout<<nr%10<<" ";
        nr=nr/10;
    }while (nr!=0);
}
```

Activité 2 :

- Préciser les modalités algorithmiques qui permettent d'obtenir les chiffres d'un nombre (la division entière par 10, les rôles des quotients et des restes) ;

- Piste de correction :

Le pseudo-code :

```
lire nr
s ← 0
┌ répéter
│   s ← s + nr%10
│   nr ← [nr/10]
└ jusqu'à nr=0
écrire s
```

Le programme C++ :

```
#include<iostream.h>
void main()
{
    long nr,s;
    cin>>nr;
    s=0;
    do
    {
        s=s+nr%10;
        nr=nr/10;
    }while(nr!=0);
    cout<<s;
}
```

Activité 3 :

- Préciser la modalité algorithmique qui permet de déterminer la multiplicité d'un facteur dans un nombre : on part du nombre et, tant qu'il est divisible par le facteur, on le divise et on compte le nombre d'itérations dans la boucle. Si dès le début le nombre n'est pas divisible par deux, la boucle ne doit faire aucune itération, c'est pourquoi on préfère la boucle prétestée.

- Piste de correction :

Le pseudo-code :

```
lire nr
m2 ← 0
┌ tant que nr%2=0
│   m2 ← m2+1
│   nr ← [nr/2]
└
écrire m2
```

Le programme C++ :

```
#include<iostream.h>
void main()
{
    long nr,m2;
    cin>>nr;
    m2=0;
    while(nr%2==0)
    {
        m2++;
        nr=nr/2;
    };
    cout<<m2;
}
```

Activité 4 :

- Préciser la modalité algorithmique qui permet d'obtenir, pour une variable i, des valeurs croissantes de pas 1, à partir d'une limite initiale en allant jusqu'à une limite finale. Préciser aussi qu'il est préférable d'utiliser la boucle prétestée bien qu'on puisse utiliser également la boucle posttestée, mais seulement si on est sûr que la limite initiale n'est pas supérieure à la limite finale.

- Piste de correction :

Le pseudo-code :

```
lire n
scp ← 0
i ← 1
┌ tant que i<=n
│   scp ← scp+i*i
│   i ← i+1
└
écrire scp
```

Le programme C++ :

```
#include<iostream.h>
void main()
{
    long n,scp,i;
    cin>>n;
    scp=0;
    i=1;
    while(i<=n)
    {
        scp=scp+i*i;
        i++;
    };
    cout<<scp;
}
```

Activité 5 :

- Pareil à l'activité 2, on doit obtenir les chiffres du nombre saisi. De plus, il faut tester chaque chiffre s'il est pair ou non, en l'additionnant à la somme seulement s'il l'est.

- **Piste de correction :**

Le pseudo-code :

```
lire nr
scp ← 0
  répéter
    c ← nr%10
    si c%2=0 alors
      scp ← scp+c
    nr ← [nr/10]
  jusqu'à nr=0
écrire scp
```

Le programme C++ :

```
#include<iostream.h>
void main()
{
  long nr,scp,c;
  cin>>nr;
  scp=0;
  do{
    c=nr%10;
    if(c%2==0) scp=scp+c;
    nr=nr/10;
  }while(nr!=0);
  cout<<scp;
}
```