Traduction
du pseudo code au C++

Instructions de base et structure de contrôle en pseudo code

- Entrées-Sorties:
  - **DEMANDER *information***
  - **AFFICHER *information***
  - **IMPRIMER *information***
- Affectation:
  - ***Var ← Expression***
- Structures de contrôle:
  - **SI *cond* ALORS Début *Bloc* Fin**
  - **SI *cond* ALORS Début $Bloc_1$ Fin**
    SINON Début $Bloc_2$ Fin
  - **RÉPÉTER *n fois* Début *Bloc* Fin**
  - **RÉPÉTER $\forall i \in [1..n]$ Début *Bloc* Fin**
  - **TANTQUE *cond* Début *Bloc* Fin**
  - **RÉPÉTER Début *Bloc* Fin TANT QUE *cond***
  - **CAS V de**
    v_1 : $Bloc_1$
    .
    .
    v_n : $Bloc_n$
    Autrement : $Bloc_a$
    FIN CAS

Traduction du bloc principal

Bloc Principal (B1)

{ Description:..

Entrées:..

Sortie:..

Résultat:..

Hypothèses:...

}

début

…..

fin

#include <iostream>

int **main( )**

/**..retranscrire intégralement les commentaires de spécification du bloc principal en commentaires doxygen.*/

{

Déclarations de variables:
demande de ressource mémoire

Traduction de l ’algorithme

return 0;

}

Déclaration de variables

• Toute variable manipulée par un programme doit être déclarée le plus près de son utilisation.

• La déclaration doit stipuler le type de la variable + le nom de la variable.

• Il faut voir qu'une variable est le lien entre une donnée manipulée par un programme et le programme: le nom de la variable est associée à un emplacement de la mémoire où est (sera) stockée la donnée. Le type de la variable précise la « nature » de l 'emplacement de stockage pour aider au codage et décodage de l'information (la donnée).

• Exemples de déclaration de variables:

```
int nbreEtudiants;      // le nombre d 'étudiants: en entier
float tauxHoraire;      // le taux horaire: un réel
char car ;              // car: un caractère
```

Séquence et Commentaires

Séquence	
Inst1 Inst2 Inst3	Inst1; Inst2; Inst3;

Commentaires	
{ Description:…} { Entrées: … } {Sorties:…} …	/ ** * \brief * \param[in] * \param[out] * \return */

UAL (même chose, sauf …)

Opérateurs arithmétiques	
7 / 2 = 3.5 7 ÷ 2 = 3	7.0 / 2.0 = 3.5 7 / 2 = 3

Opérateurs relationnelles	
x = y x ≠ y	x == y x != y

Opérateurs logiques	
C_1 ET C_2 C_1 OU C_2 NON C	C_1 && C_2 C_1 || C_2 !C

Bloc et Affectation

DEBUT … FIN	{ … }

var ← exp	var = exp
i ← 1	i = 1
i ← i + 1	i = i + 1 i++
i ← i - 1	i = i - 1 i--
n ← n + a	n = n + a n += a

-=, *=, /=, %=

Entrées-Sorties console

AFFICHER	**cout**
AFFICHER "message"	cout << "message" ;
AFFICHER i	cout << i ;
AFFICHER " n = ", i	cout << " n = " << i ;

Entrées-Sorties console

DEMANDER	**cin**
DEMANDER i AFFICHER "Entrer une valeur: " DEMANDER i	cin >> i ; cout << " Entrer une valeur:" ; cin >> i ;

Structures de contrôle

SI *cond* **ALORS** *bloc*	**if** *cond bloc*

SI (n > 1)
ALORS
  DEBUT
  i ← i + 1
  FIN
	if (n > 1)
	{
	  i = i + 1;
	}

Structures de contrôle

SI *cond* **ALORS** *bloc1* **SINON** *bloc2*	**if** *cond bloc1* **else** *bloc2*
IF *cond* **ALORS** *exp1* **ELSE** *exp2*	**cond?** *exp1* **:** *exp2*

SI (n > 1)
ALORS
  DEBUT
    i ← i + 1
  FIN
SINON
  DEBUT
    i ← i - 1
  FIN

```
if ( n > 1 )
{
    i = i + 1;
}
else
{
    i = i – 1;
}
```

```
( n > 1 ) ? i++ : i--;
```

Structures de contrôle

RÉPÉTER	**for**
RÉPÉTER ∀**i** ∈ **[2 .. 10]** **DEBUT** … **FIN**	**for** (int i = 2; i <= 10; i++) **{** … **}**
RÉPÉTER *n* **fois** **DEBUT** … **FIN**	**for** (int i = 1; i <= n; i++) **{** … **}**

Structures de contrôle

TANT QUE	while
TANT QUE (i > j) **DEBUT** … **FIN**	**while** (i > j) **{** … **}**

Structures de contrôle

RÉPÉTER … TANT QUE	do … while
RÉPÉTER **DEBUT** … **FIN** **Tant Que** (i > j)	**do** **{** … **}** **while** (i > j) **;**

Structures de contrôle

CAS *v* DE …	**switch(*v*)**
CAS *v* DE 1 : **AFFICHER** "un" 2 : **AFFICHER** "deux" **Autrement** : **AFFICHER** "erreur!" **FINCAS**	**switch(*v*)** **{** **case** 1 : **cout** << "un" ; **break**; **case** 2 : **cout** << "deux" ; **break**; **default** : **cout** << "erreur!" ; }