Ministry of Higher Education and Scientific Research
Djilali BOUNAAMA University - Khemis Miliana(UDBKM)
Faculty of Science and Technology
Department of Mathematics and Computer Science

Chapter 2

Simple Sequential Algorithm

MI-L1-UEF121 : Algorithms and Data Structures I

Noureddine AZZOUZA

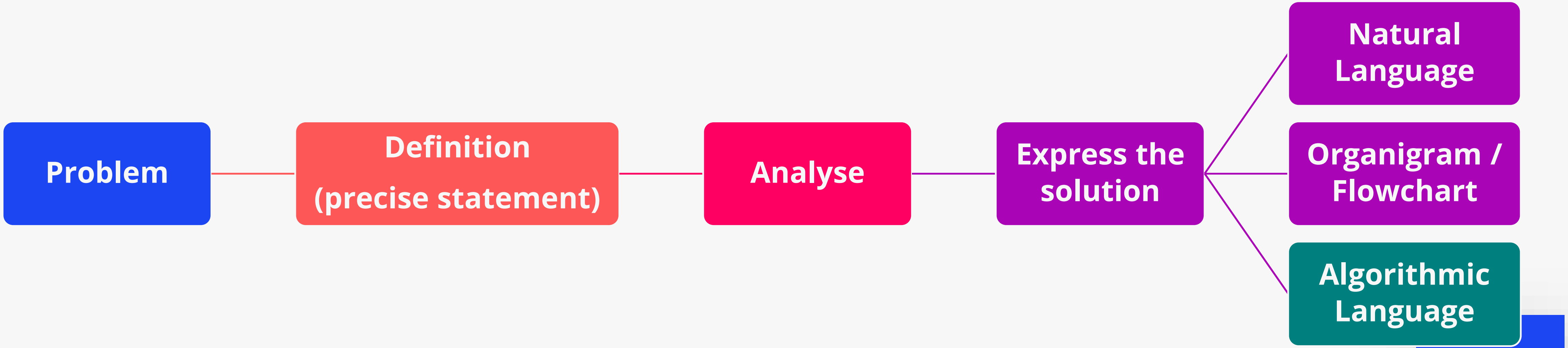
n.azzouza@univ-dbkm.dz

Course Topics

1. Structure of an algorithm
2. Objects: Constants, Variables and Types
3. Basic actions
   - 3.1 Assignment
   - 3.2 Expressions
   - 3.3 Inputs / Outputs
4. Representation of an algorithm by a flowchart
5. Translation into Programming Language

Structure of an algorithm

Need for an algorithmic language



- ✓ **Natural Language** (literary descriptions): imprecision, ambiguity, own rules and conventions, different explanations of the same concept …
- ✓ **Flowchart :** clutter, hard to modify and update, …

Algorithmic Language

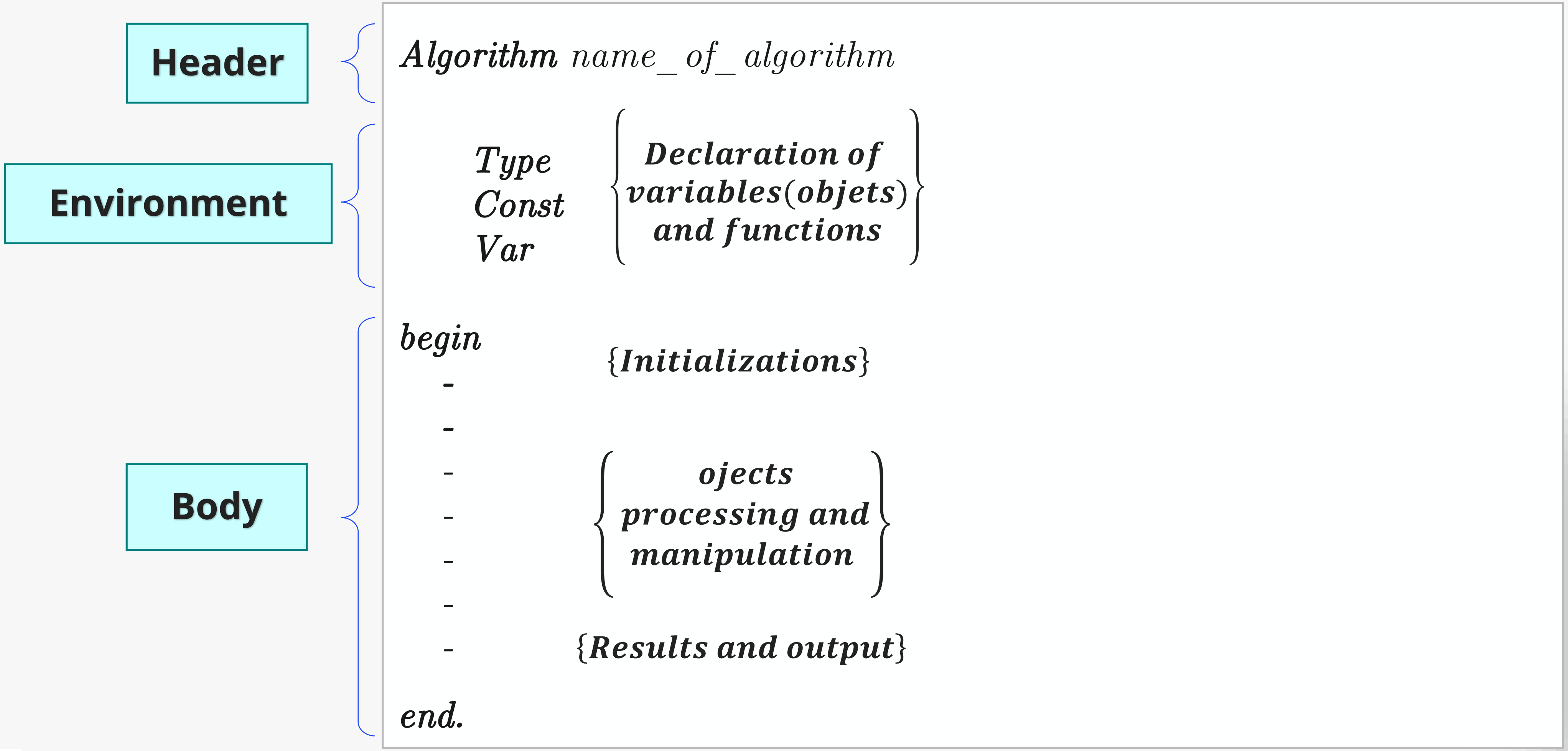
- ✓ An **algorithmic language** (or formalism) is a set of ***conventions*** (or ***rules***) in which we can express any solution to a given problem.

- ✓ Also called: **Pseudo Code**,

- ✓ **Properties :**

  - ➢ A ***common*** language;
  - ➢ Principle of ***communication***;
  - ➢ ***Precision*** and clarity (non-ambiguity)

Structure of an Algorithme



```
Algorithm name_of_algorithm

    Type        ( Declaration of          )
    Const       ( variables(objets)       )
    Var         ( and functions           )

begin
                {Initializations}
    -
    -
    -           ( ojects                  )
    -           ( processing and          )
    -           ( manipulation            )
    -
    -           {Results and output}
end.
```

Structure of an Algorithme

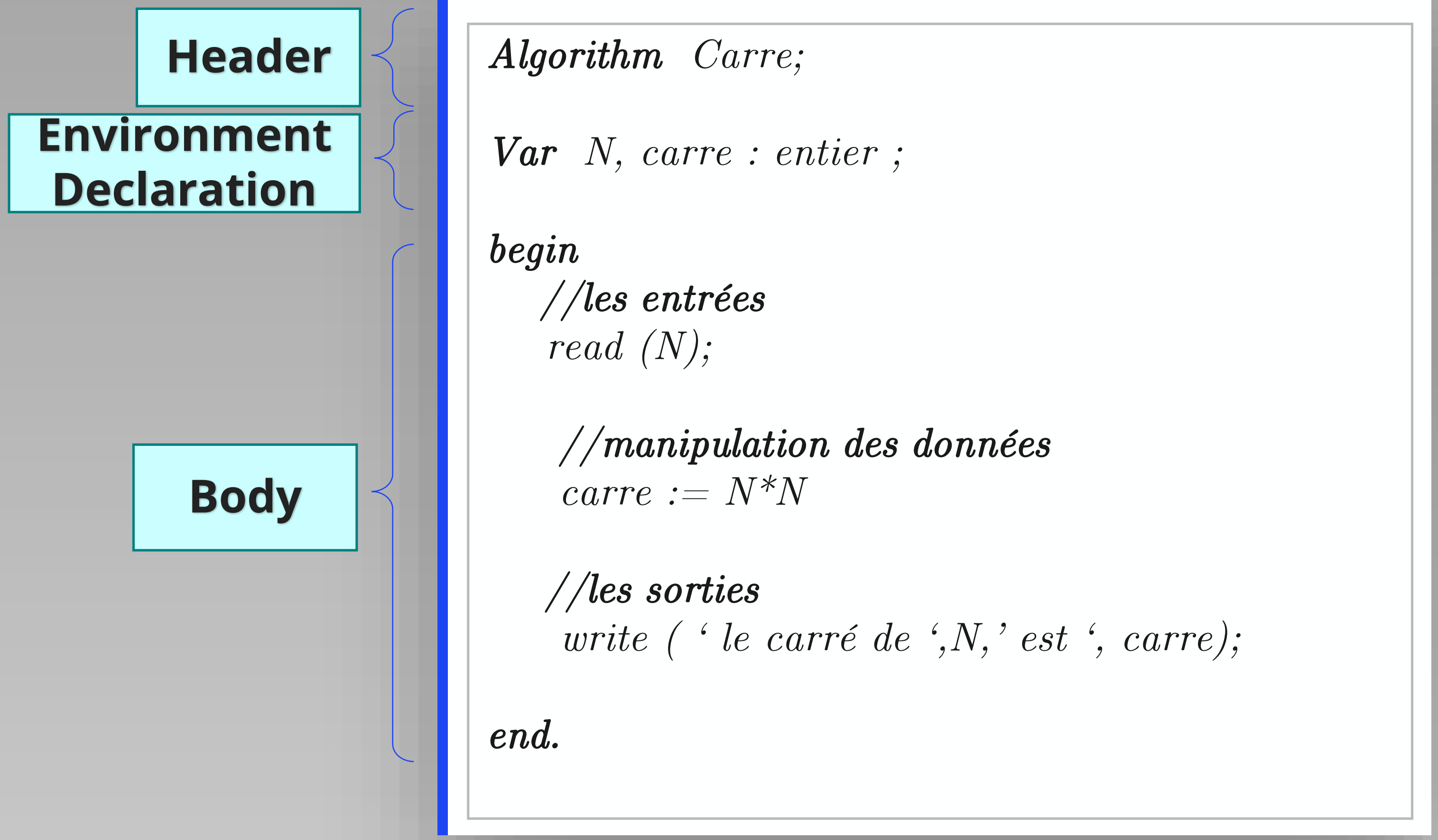
Comments

- ✓ Gives a human description in machine code.
- ✓ Simplified code maintenance and therefore, accelerate debugging
- ✓ Important when writing functions for other users

Syntax

```
//        a Single line comment
/*        Comment on
multiple lines */
```

Example : Square of a Number

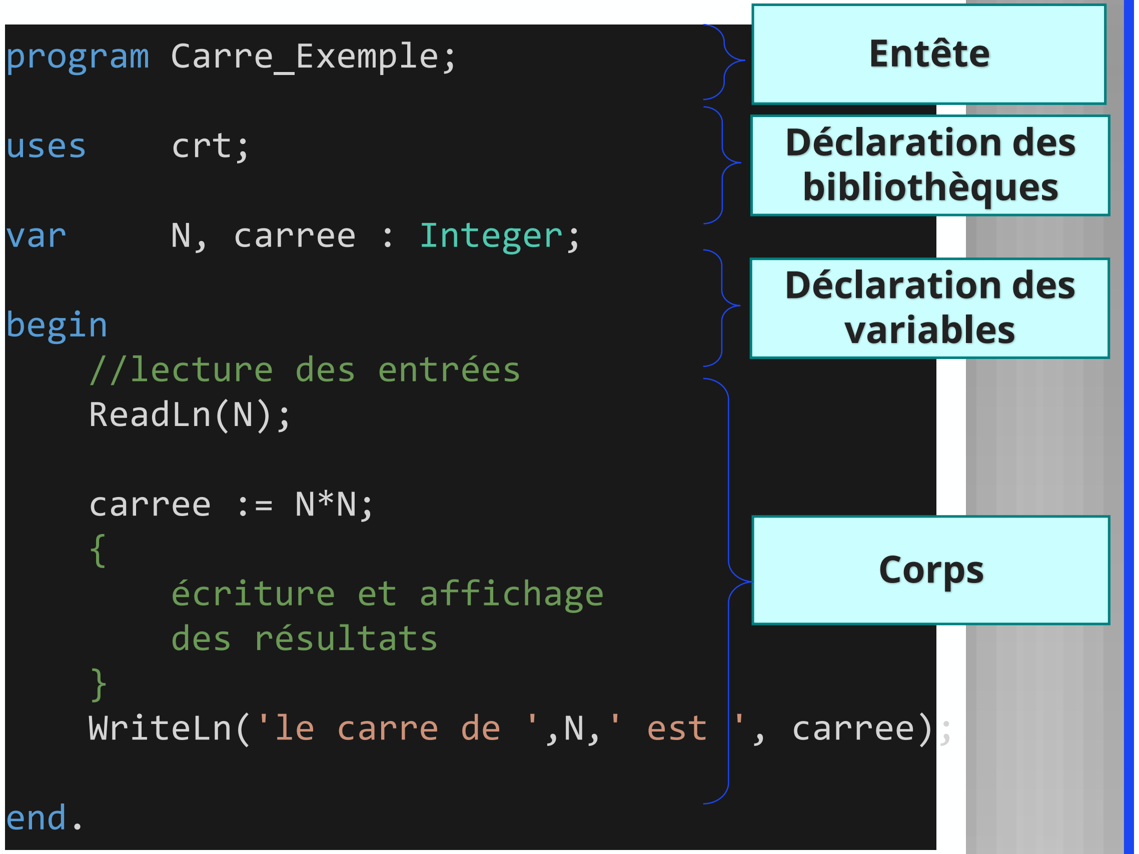


```
Algorithm  Carre;

Var  N, carre : entier ;

begin
    //les entrées
    read (N);

     //manipulation des données
     carre := N*N

    //les sorties
     write ( ‘ le carré de ‘,N,’ est ‘, carre);

end.
```

Structure of an Algo

Example : Square of a Number

Programme en langage PASCAL



```
program Carre_Exemple;

uses    crt;

var     N, carree : Integer;

begin
    //lecture des entrées
    ReadLn(N);

    carree := N*N;
    {
        écriture et affichage
        des résultats
    }
    WriteLn('le carre de ',N,' est ', carree);

end.
```

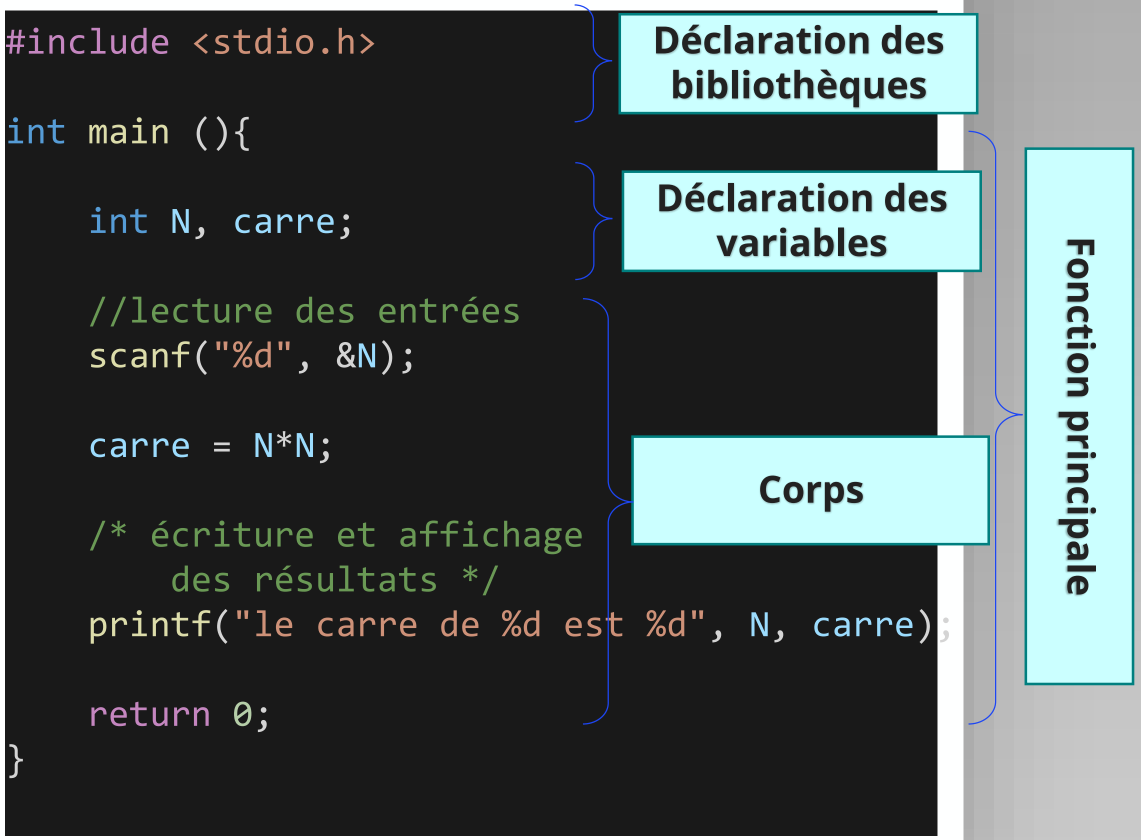
Entête

Déclaration des bibliothèques

Déclaration des variables

Corps

Programme en langage C



```
#include <stdio.h>

int main (){

    int N, carre;

    //lecture des entrées
    scanf("%d", &N);

    carre = N*N;

    /* écriture et affichage
        des résultats */
    printf("le carre de %d est %d", N, carre);

    return 0;
}
```

Déclaration des bibliothèques

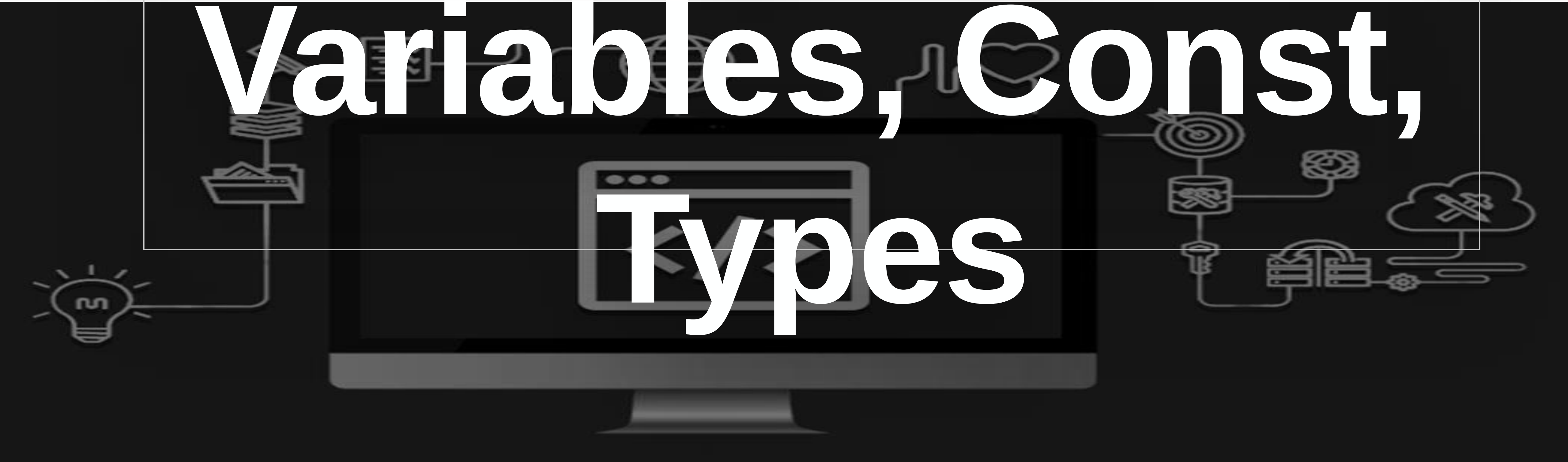
Déclaration des variables

Corps

Fonction principale

Objects

Variables, Const, Types



Objects: Memory Representation

- ✓ Any **object** manipulated by an **algorithm** (or a **program**) is ***stored*** in ***central memory*** (***RAM***).

- ✓ ***Central memory*** is made up of a ***series of contiguous boxes*** called memory "**boxes**" or "**cells**".

- ✓ Each **memory box** is characterized by :
  - ➢ An ***address***: ***unique*** which references the box
  - ➢ A ***space***: to store object ***values***

Objects: Memory Representation

Adress @	Memory (Valeur)	Object (Var, Const, ..)
63999		
63998		
63997		
14792	10	i
14791	14.75	Note
14790	0	J
14789		
14788	Ali	Nom
2		
1		
0		

Objects: Definition

- ✓ All constituent **objects** of an **algorithm** must be ***described*** or ***declared*** in the ***environment*** (or the “***declaration***” part).

- ✓ Each object is characterized by :

  - ➢ A **Name**: “***unique identifier***”: a series of ***alphanumeric characters*** which allows it to be designated and distinguished

  - ➢ A **Type**: which indicates the ***nature*** of the set in which it takes its values

  - ➢ A **value**: which indicates the ***size*** taken by an object at a given moment

Objects: identifiers

- ✓ A **name** or **identifier** is a sequence of ***alphanumeric characters*** whose first character is ***alphabetic***
  - ➢ The identifier can be: program name, variable and constant names, function names
  - ➢ Can contain letters and numbers
  - ➢ No (most) punctuation marks
- ✓ **Examples :**
  - ➢ **Valid identifiers**: product, i, j, T1, L_21, surface, student_name,
  - ➢ **Invalid identifiers**: 5students, release date, x+y, T1, ...

Programming: Identifiers

Identificateurs en langage PASCAL

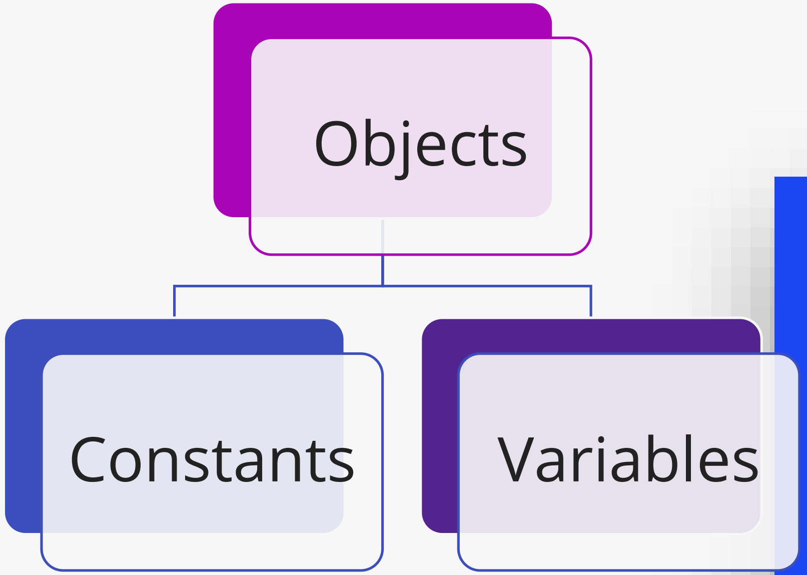
1. L’identificateur est “Unique”
2. Il ne doit pas contenir des:
   - Caractères accentués, ni d'espace, ni des caractères tels que %,?,*,.,-,
3. Il doit exclusivement être composé des:
   - 26 lettres de l'alphabet, des 10 chiffres et le caractère de soulignement
4. Un chiffre ne peut pas être placé au début d’un identificateur.
5. ne différencie pas entre majuscule et minuscule.

Identificateurs en langage C

1. Les meme conditions que le langage PASCAL
2. fait la distinction entre lettres minuscules et majuscules.
3. Par convention:
   - Les noms des variables et fonctions ne contiennent pas de majuscules
   - Les constantes ne contiennent pas de minuscules
   - utilise le souligné pour séparer les mots

Objets: Categories

- ✓ **Objects** are used to ***store data*** manipulated by the ***algorithm***
- ✓ There are two categories of objects
  - ➢ **Constant** : it is an object whose value is ***invariable***
  - ➢ **Variable** : it is an object that ***can vary*** during the execution of an algorithm .

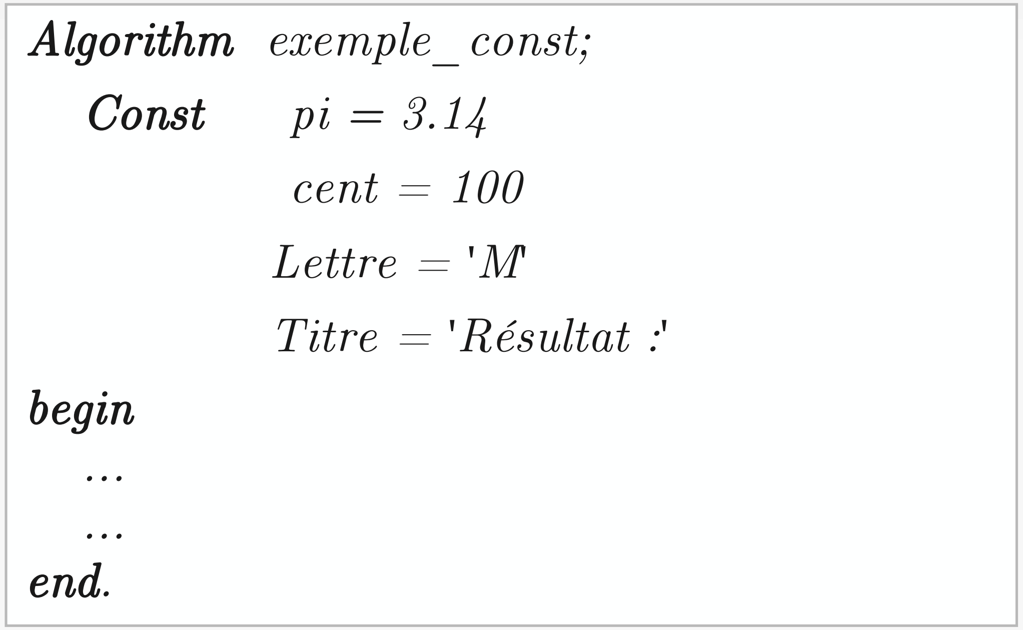


Constants

Declaration

Const *nom_constante* **=** *Valeur*

Examples



```
Algorithm exemple_const;
    Const   pi = 3.14
            cent = 100
            Lettre = 'M'
            Titre = 'Résultat :'
begin
    ...
    ...
end.
```

Variables, Constants and Types

Variables, Constants and Types

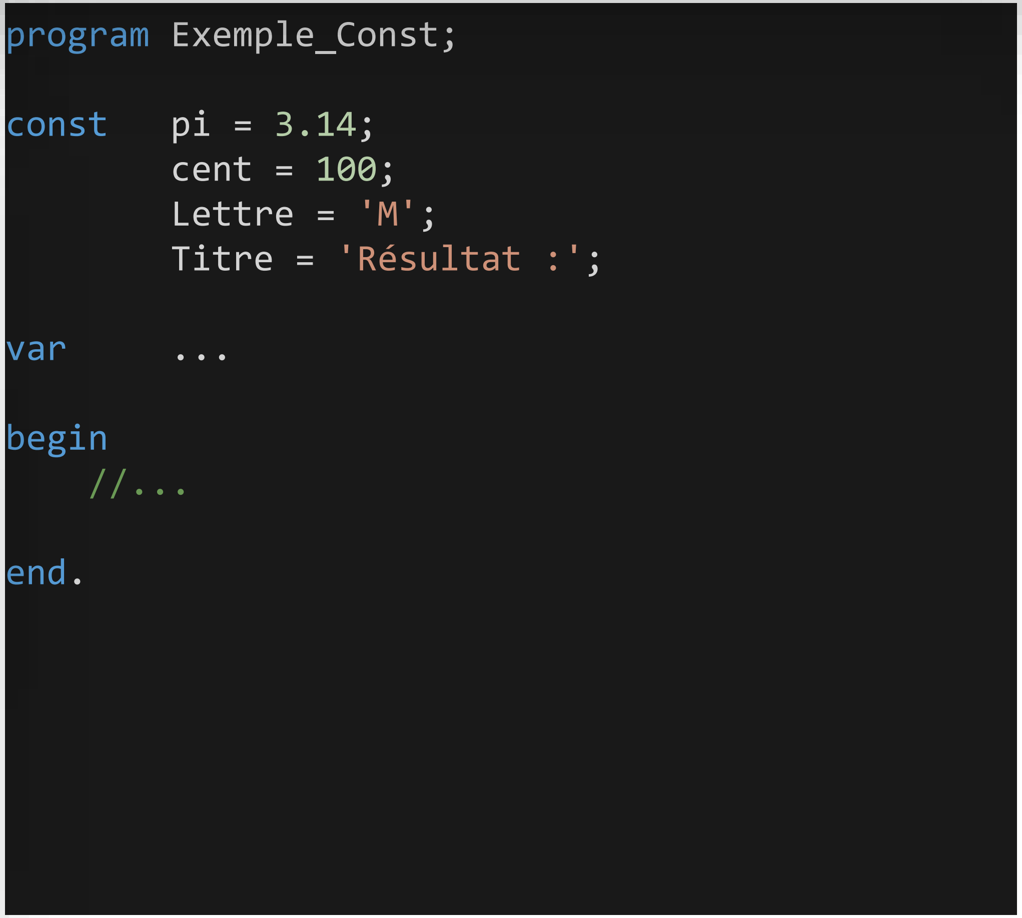
PASCAL

C

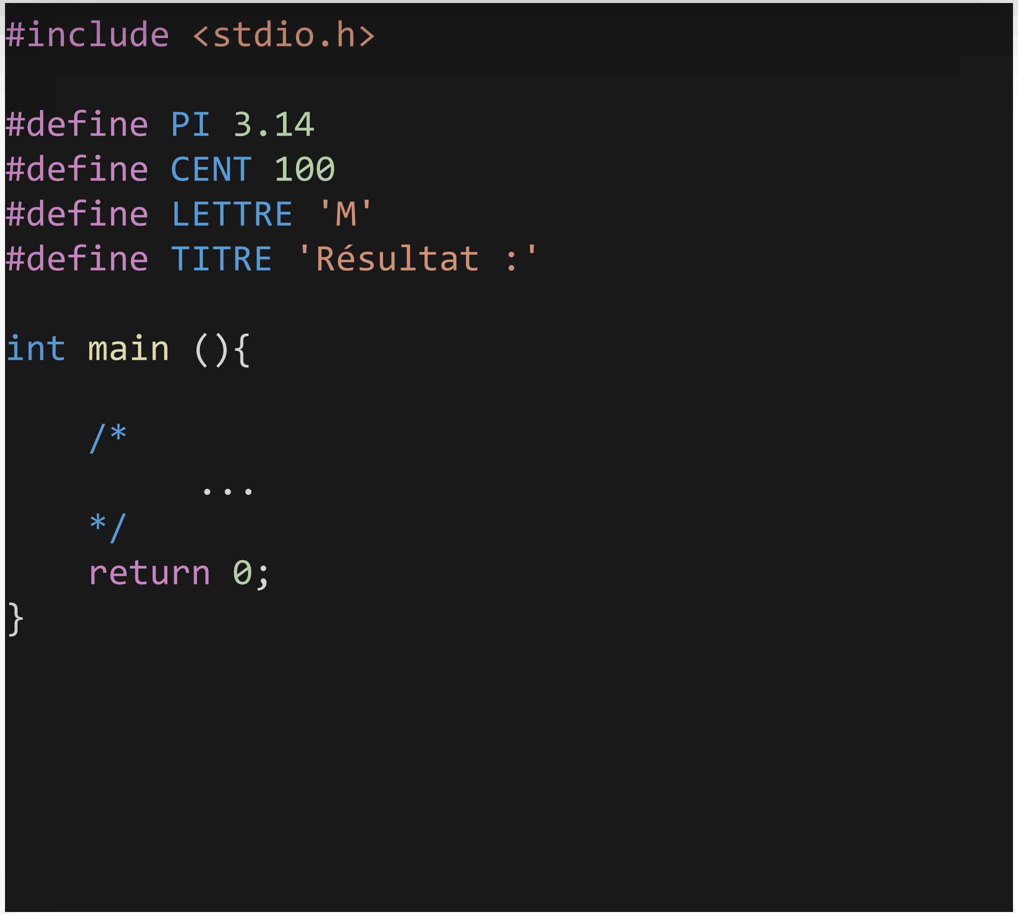
Déclaration

	PASCAL	C
Déclaration	**Const** nom_constante **=** Valeur	**#define** NOM_CONST valeur

Exemples



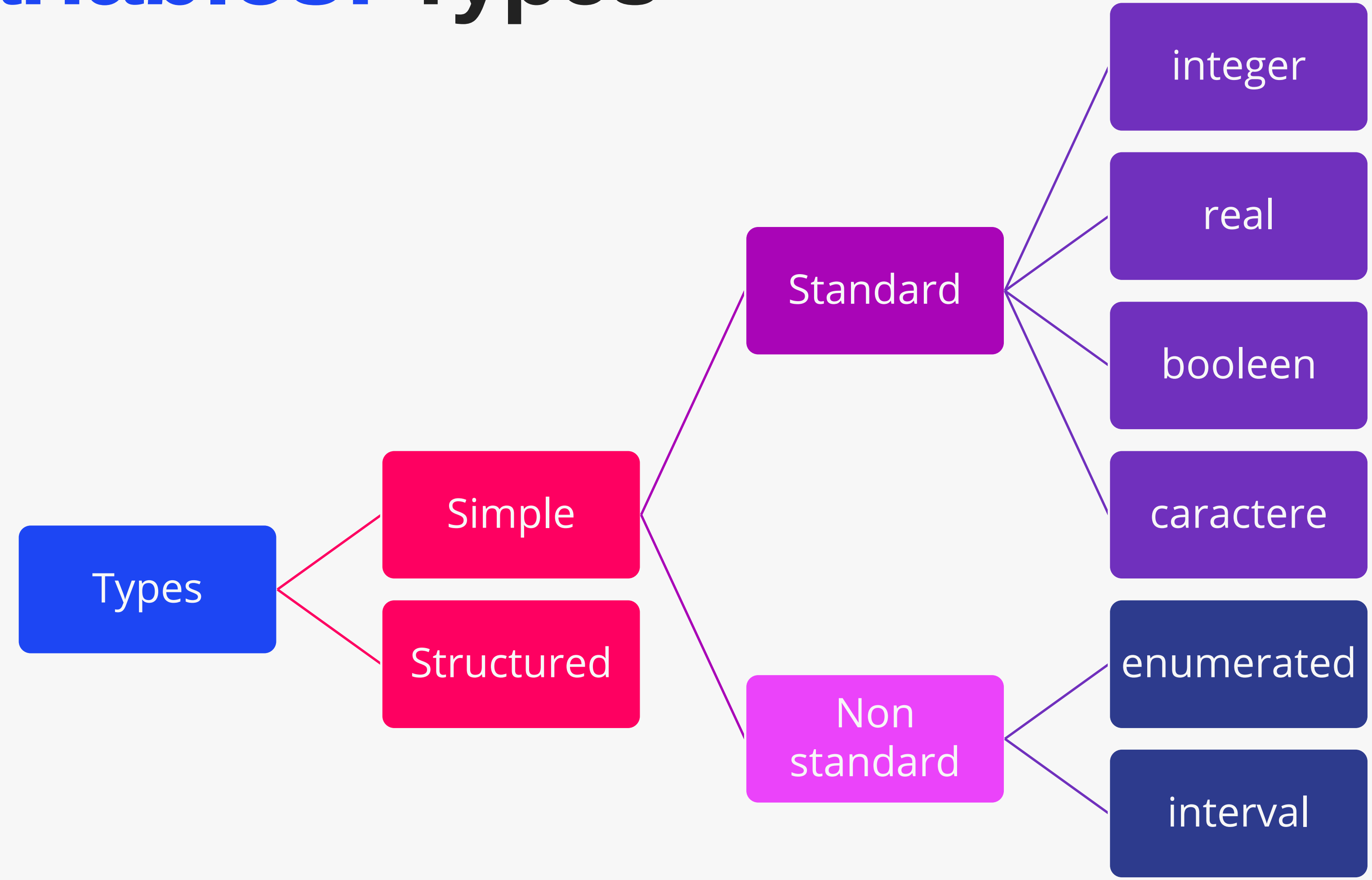
```
program Exemple_Const;

const   pi = 3.14;
        cent = 100;
        Lettre = 'M';
        Titre = 'Résultat :';

var     ...

begin
    //...

end.
```



```
#include <stdio.h>

#define PI 3.14
#define CENT 100
#define LETTRE 'M'
#define TITRE 'Résultat :'

int main (){

    /*
          ...
    */
    return 0;
}
```

Variables, Constants and Types

Variables: Types

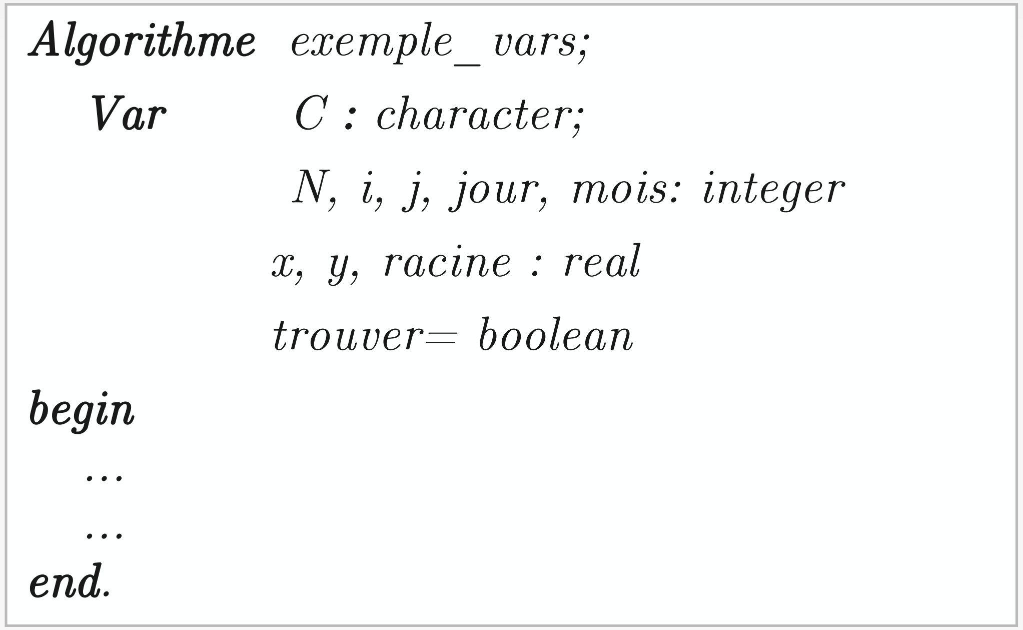


Variables

Declaration

Var *nom_variable* **: Type**

Examples



```
Algorithme exemple_ vars;
    Var     C : character;
            N, i, j, jour, mois: integer
           x, y, racine : real
           trouver= boolean
begin
    ...
    ...
end.
```

Types

Variables, Constants and Types

Numerical Types

- **Integer** : it is the set of relative integers.

- **Real** : It is the set of numbers having a fractional part.

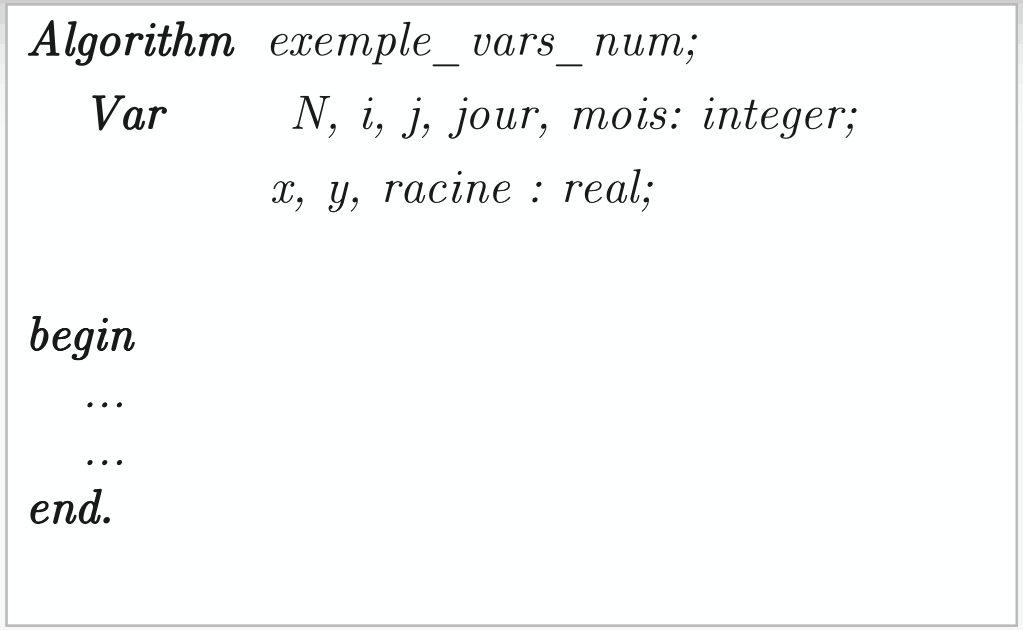
Type	Octets	Plage
Byte (octet)	1	0 à 255
Entier simple	2	-32 768 à 32 767
Entier long	4	-2 147 483 648 à 2 147 483 647
Réel simple	4	-3,40x10^{38} à -1,40x10^{45} pour les valeurs négatives 1,40x10^{-45} à 3,40x10^{38} pour les valeurs positives
Réel double	8	1,79x10^{308} à -4,94x10^{-324} pour les valeurs négatives 4,94x10^{-324} à 1,79x10^{308} pour les valeurs positives

Numerical Types

Declaration

```
Var    nom_variable1: integer
       nom_variable2: real
```

Examples



```
Algorithm exemple_vars_num;
    Var    N, i, j, jour, mois: integer;
           x, y, racine : real;

begin
    ...
    ...
end.
```

Numerical Types

Déclaration

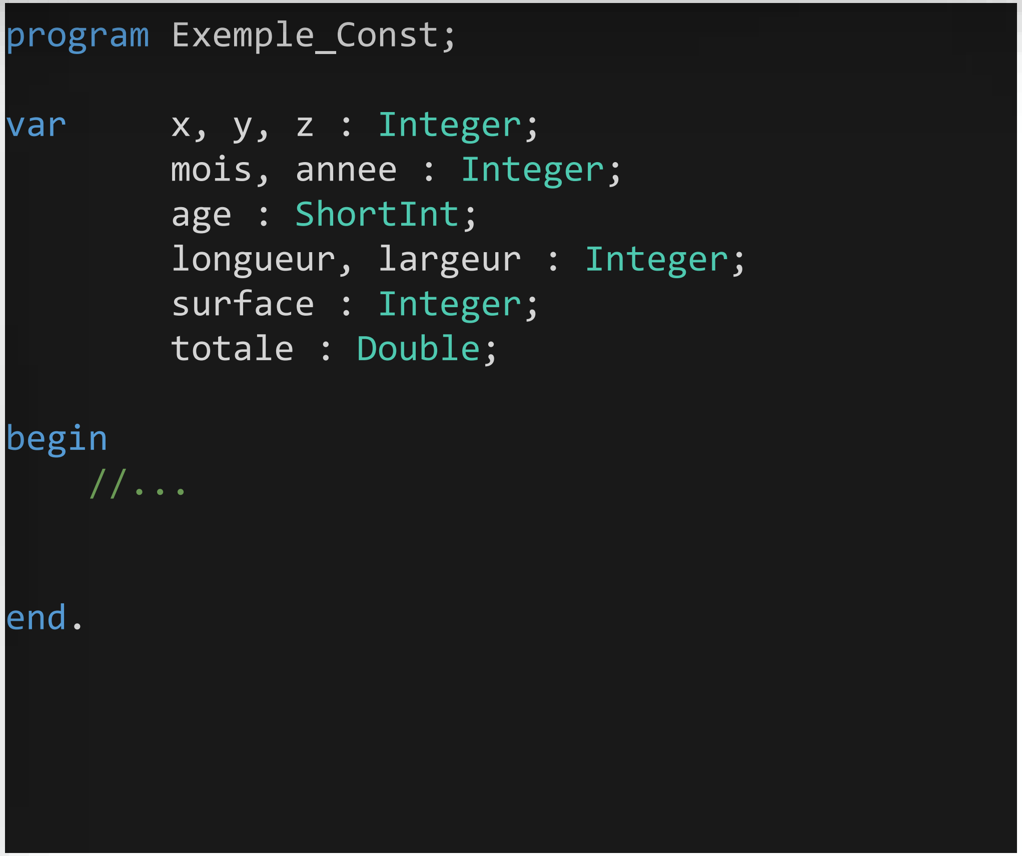
PASCAL

integer : byte, shortint, integer, longint
real : real, double, extended

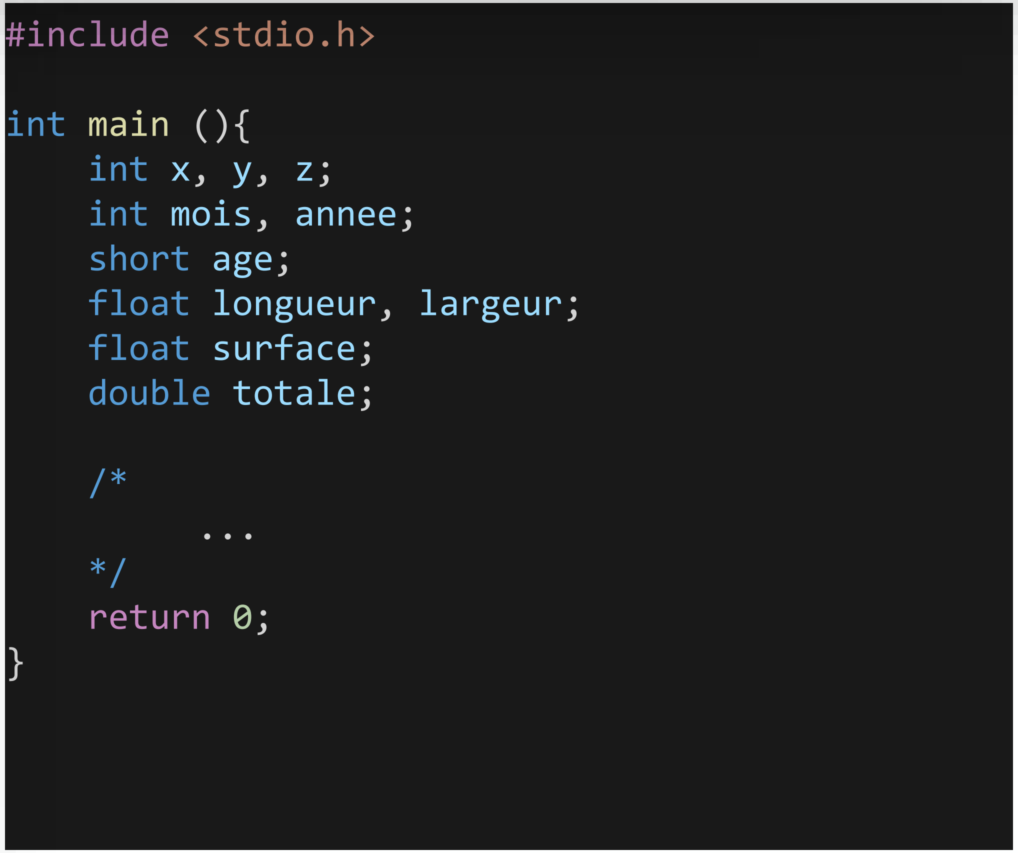
C

integer : short, int, long, double long
real : float, double, long double

Exemples



```pascal
program Exemple_Const;

var     x, y, z : Integer;
        mois, annee : Integer;
        age : ShortInt;
        longueur, largeur : Integer;
        surface : Integer;
        totale : Double;

begin
    //...


end.
```



```c
#include <stdio.h>

int main (){
    int x, y, z;
    int mois, annee;
    short age;
    float longueur, largeur;
    float surface;
    double totale;

    /*
          ...
    */
    return 0;
}
```

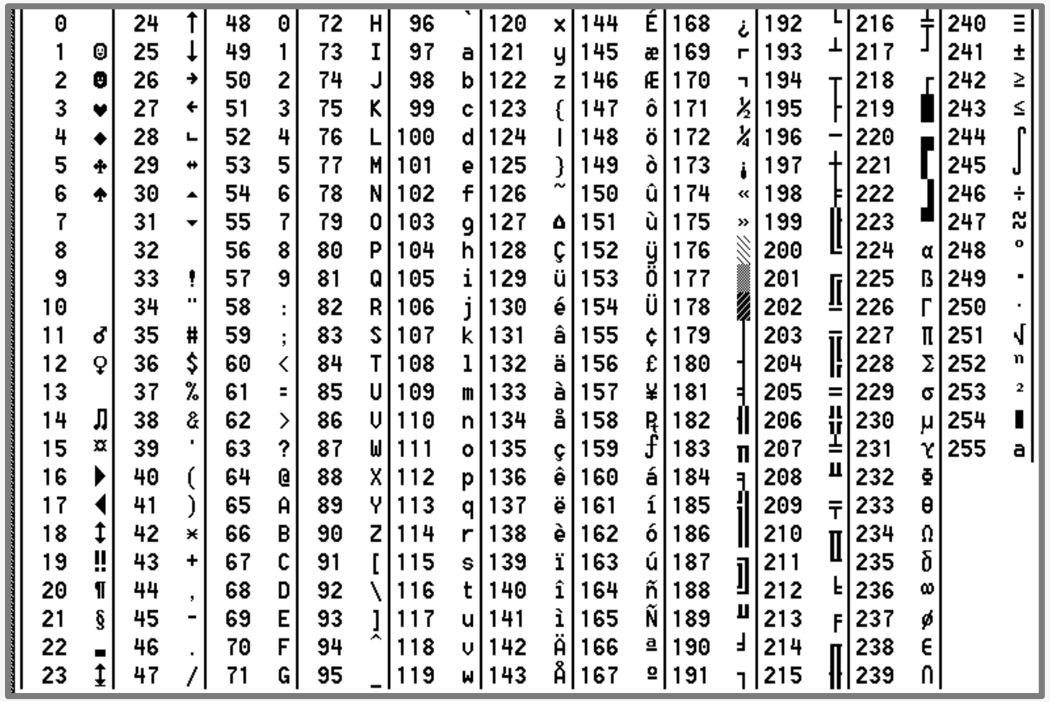
Alphanumeric Types

- **CHARACTER** : it matches a single character (depending on the system). Character sets may vary.
  - Placed between two single quotes.
  - It includes all alphabetical, numerical characters, punctuation marks, special signs, space (blank), empty character, etc.

- **STRING** : It is a set (group) of characters.

Alphanumeric Types

- Each character has an ASCII code
- The ASCII code is a table of 0 to 255 characters which contains: lowercase and uppercase letters, numbers, punctuation, special symbols and graphics …



0		24	↑	48	0	72	H	96	`	120	x	144	É	168	¿	192	└	216	╪	240	≡
1	☺	25	↓	49	1	73	I	97	a	121	y	145	æ	169	⌐	193	┴	217	┘	241	±
2	☻	26	→	50	2	74	J	98	b	122	z	146	Æ	170	¬	194	┬	218	┌	242	≥
3	♥	27	←	51	3	75	K	99	c	123	{	147	ô	171	½	195	├	219	█	243	≤
4	♦	28	∟	52	4	76	L	100	d	124	|	148	ö	172	¼	196	─	220	▄	244	⌠
5	♣	29	↔	53	5	77	M	101	e	125	}	149	ò	173	¡	197	┼	221	▌	245	⌡
6	♠	30	▲	54	6	78	N	102	f	126	~	150	û	174	«	198	╞	222	▐	246	÷
7		31	▼	55	7	79	O	103	g	127	⌂	151	ù	175	»	199	╟	223	▀	247	≈
8		32		56	8	80	P	104	h	128	Ç	152	ÿ	176	░	200	╚	224	α	248	°
9		33	!	57	9	81	Q	105	i	129	ü	153	Ö	177	▒	201	╔	225	ß	249	∙
10		34	"	58	:	82	R	106	j	130	é	154	Ü	178	▓	202	╩	226	Γ	250	·
11	♂	35	#	59	;	83	S	107	k	131	â	155	¢	179	│	203	╦	227	π	251	√
12	♀	36	$	60	<	84	T	108	l	132	ä	156	£	180	┤	204	╠	228	Σ	252	ⁿ
13		37	%	61	=	85	U	109	m	133	à	157	¥	181	╡	205	═	229	σ	253	²
14	♫	38	&	62	>	86	V	110	n	134	å	158	₧	182	╢	206	╬	230	µ	254	■
15	☼	39	'	63	?	87	W	111	o	135	ç	159	ƒ	183	╖	207	╧	231	γ	255	a
16	►	40	(	64	@	88	X	112	p	136	ê	160	á	184	╕	208	╨	232	Φ		
17	◄	41	)	65	A	89	Y	113	q	137	ë	161	í	185	╣	209	╤	233	Θ		
18	↕	42	*	66	B	90	Z	114	r	138	è	162	ó	186	║	210	╥	234	Ω		
19	‼	43	+	67	C	91	[	115	s	139	ï	163	ú	187	╗	211	╙	235	δ		
20	¶	44	,	68	D	92	\	116	t	140	î	164	ñ	188	╝	212	╘	236	∞		
21	§	45	-	69	E	93	]	117	u	141	ì	165	Ñ	189	╜	213	╒	237	φ		
22	▬	46	.	70	F	94	^	118	v	142	Ä	166	ª	190	╛	214	╓	238	ε		
23	↨	47	/	71	G	95	_	119	w	143	Å	167	º	191	┐	215	╫	239	∩		

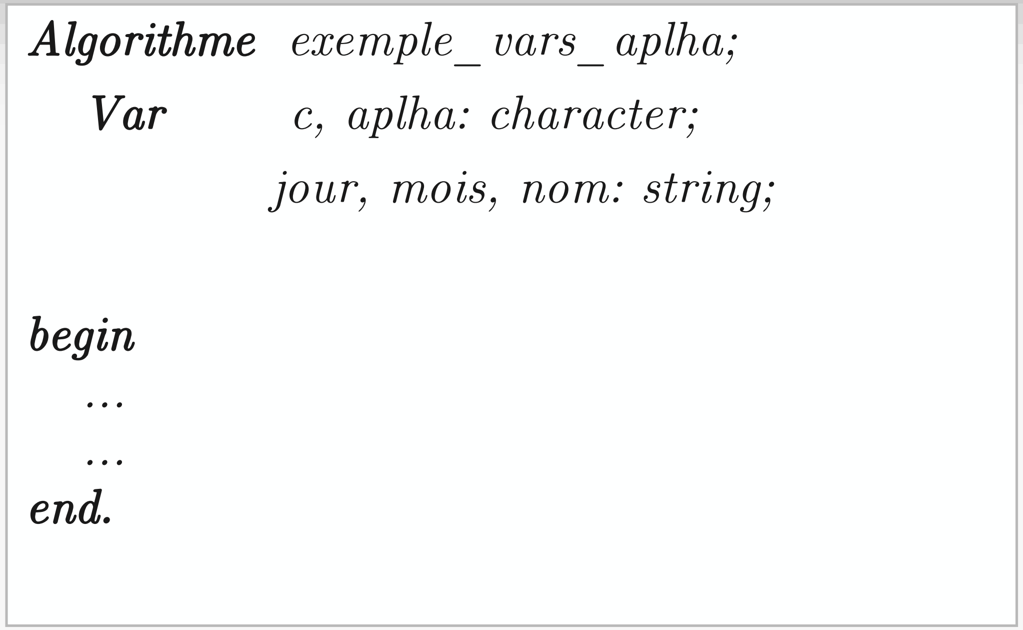
ASCII Code

Alphanumeric Types

Declaration

```
Var     nom_variable1: character
        nom_variable2: string
```

Examples



```
Algorithme exemple_vars_aplha;
    Var     c, aplha: character;
            jour, mois, nom: string;

begin
    ...
    ...
end.
```

Variables, Constants and Types

Alphanumeric Types

Declaration

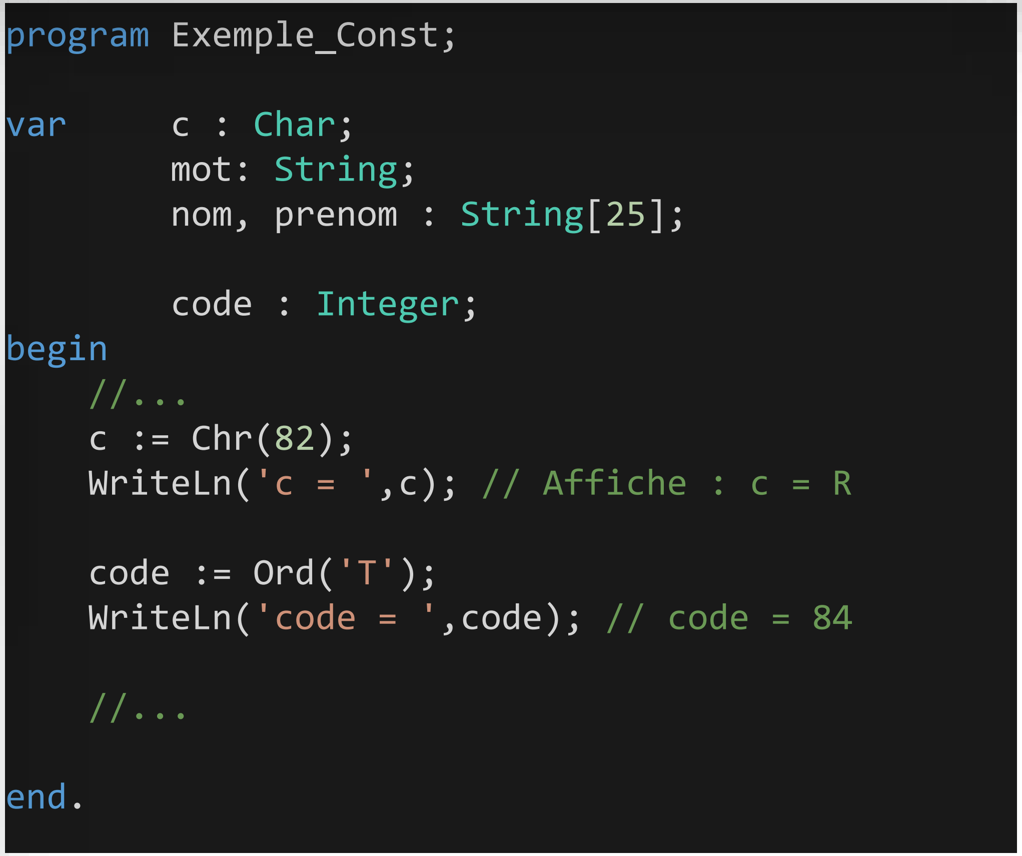
PASCAL

character : Char
String: String

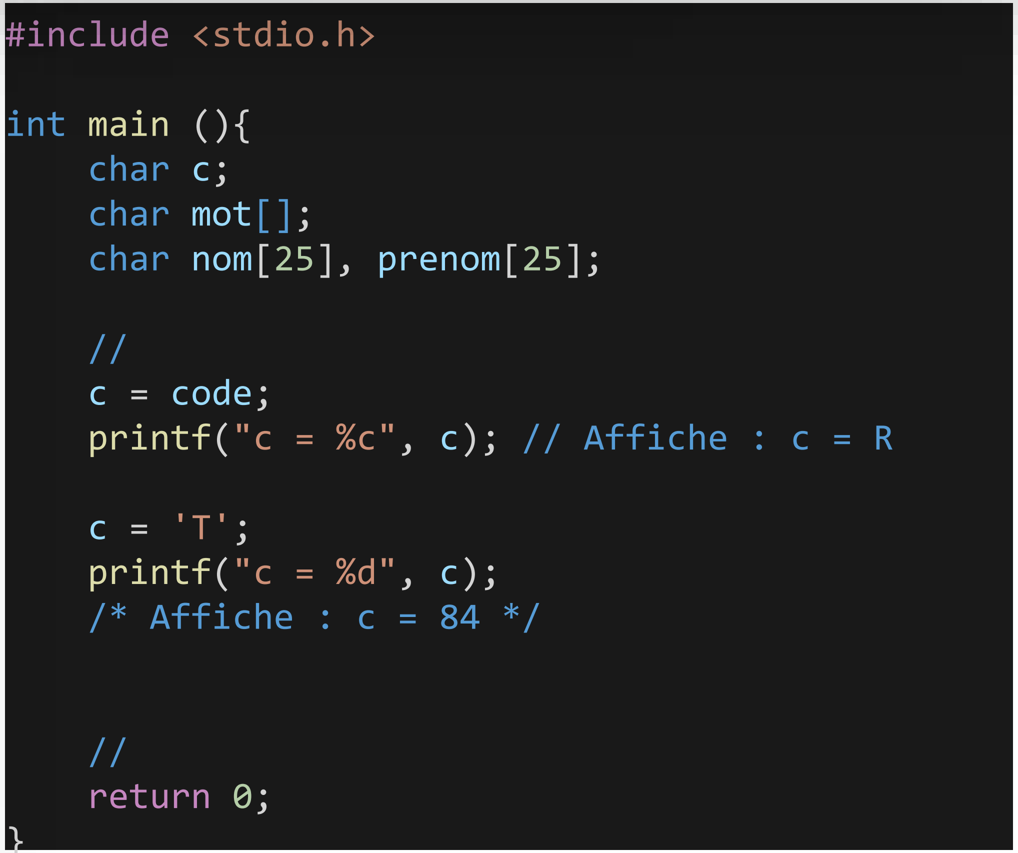
C

character : char
String : table if characters

Examples



```
program Exemple_Const;

var      c : Char;
         mot: String;
         nom, prenom : String[25];

         code : Integer;
begin
    //...
    c := Chr(82);
    WriteLn('c = ',c); // Affiche : c = R

    code := Ord('T');
    WriteLn('code = ',code); // code = 84

    //...

end.
```



```
#include <stdio.h>

int main (){
    char c;
    char mot[];
    char nom[25], prenom[25];

    //
    c = code;
    printf("c = %c", c); // Affiche : c = R

    c = 'T';
    printf("c = %d", c);
    /* Affiche : c = 84 */


    //
    return 0;
}
```

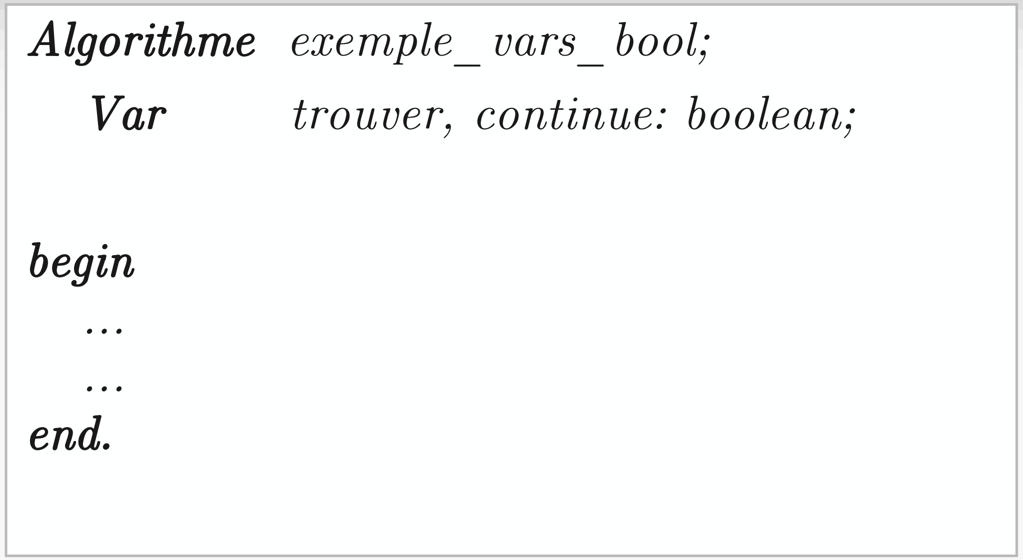
Boolean Type

- ➢ **BOOLEAN** : it is the set of values {TRUE, FALSE}.

Declaration

```
Var     nom_variable1: Boolean
```

Examples



```
Algorithme  exemple_ vars_ bool;
    Var     trouver, continue: boolean;

begin
    ...
    ...
end.
```

Boolean Type

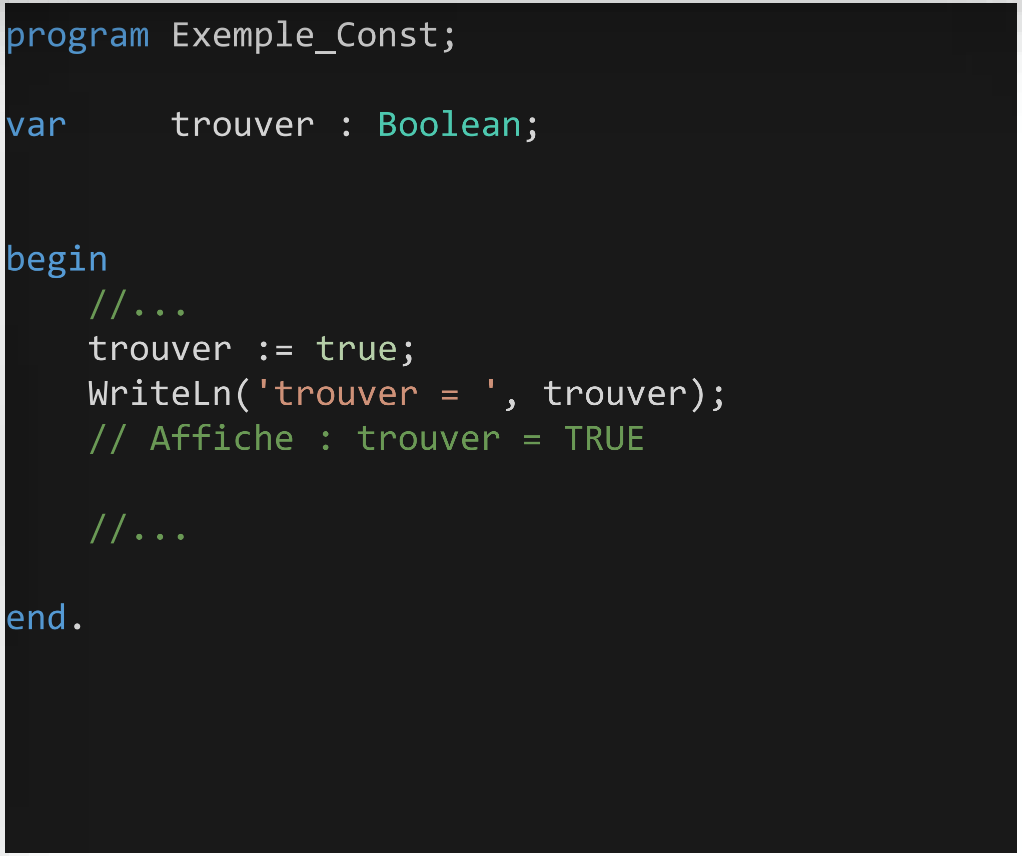
Variables, Constants and Types

PASCAL

Declaration

Boolean : Boolean

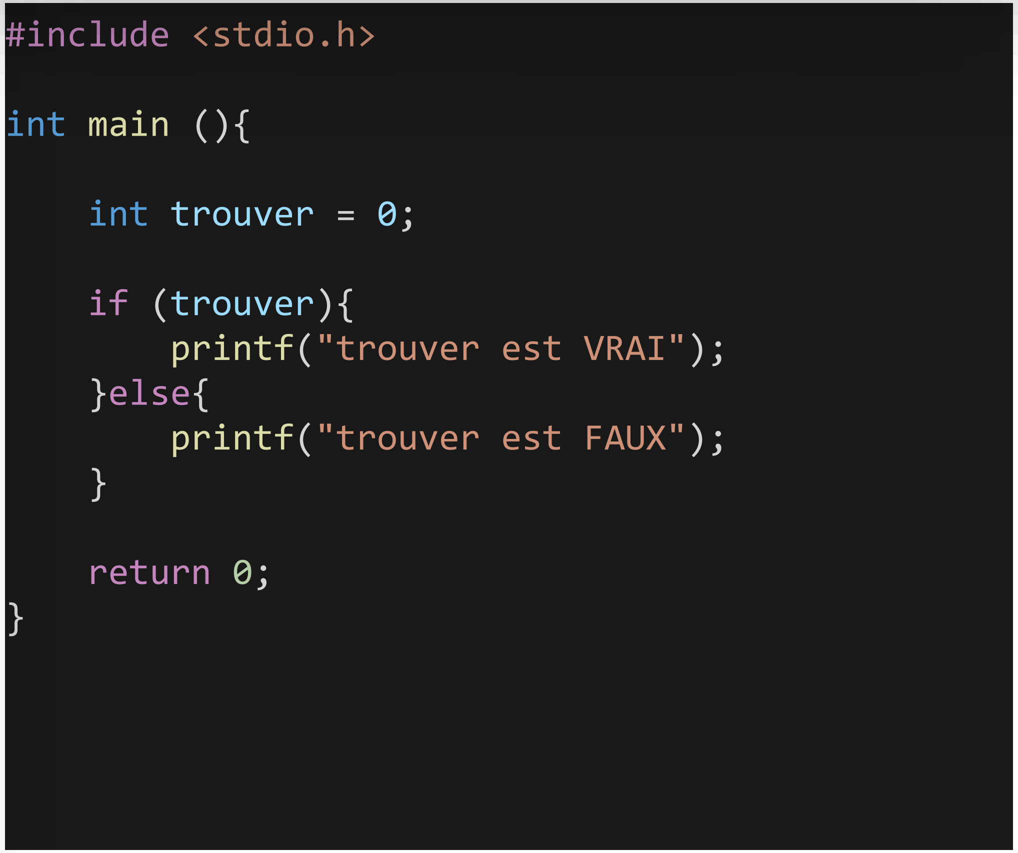
Examples



```pascal
program Exemple_Const;

var      trouver : Boolean;


begin
    //...
    trouver := true;
    WriteLn('trouver = ', trouver);
    // Affiche : trouver = TRUE

    //...

end.
```

C

Boolean : nécessite un bibliothèque « **stdbool.h** » sinon, toutes valeur **non nulle** correspond à « **true** » et **0** correspond à « **false** »

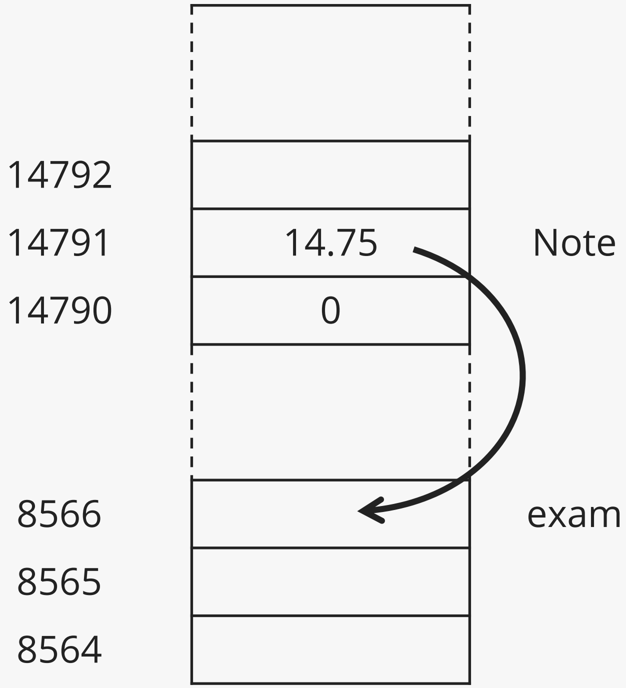


```c
#include <stdio.h>

int main (){

    int trouver = 0;

    if (trouver){
        printf("trouver est VRAI");
    }else{
        printf("trouver est FAUX");
    }

    return 0;
}
```

Basic Instructions

Assignment

- ✓ Its role is to ***assign*** (give, attribute) a **value** to a variable
- ✓ This value can be :
    - ➢ A constant
    - ➢ The value of another variable or constant
    - ➢ an expression



exam := Note

Syntax

variable := *expression*

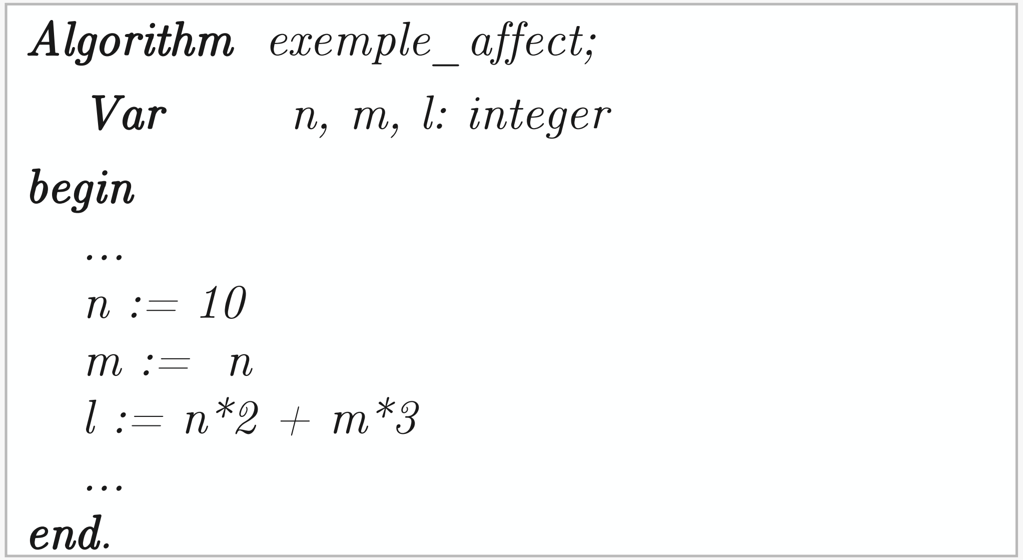
Assignment

✓ Double **rôles** de l'affectation Dual **roles** of assignment :

1) Calculate and evaluate the expression to the ***right*** of the symbol: ←

2) Assign and store the result in the variable to the ***left*** of the symbol ←

Examples



```
Algorithm  exemple_ affect;
   Var       n, m, l: integer
begin
   ...
   n := 10
   m :=  n
   l := n*2 + m*3
   ...
end.
```

Basic Instructions

Assignment

Declaration

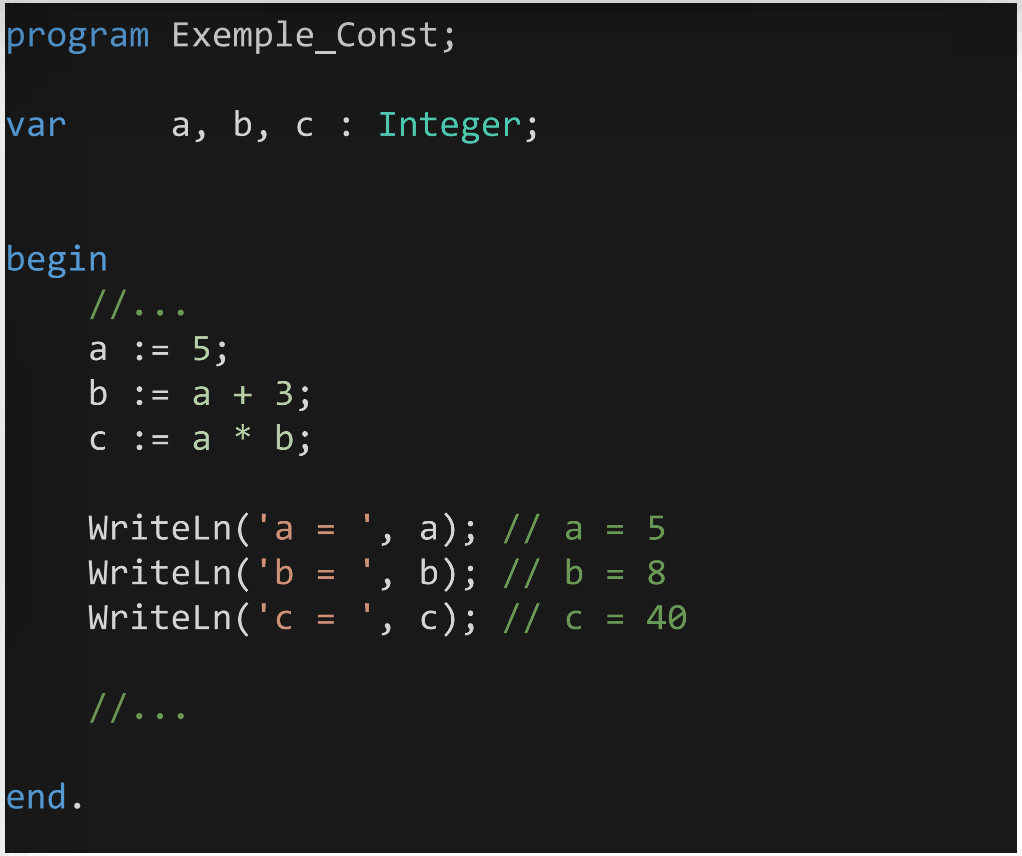
PASCAL

variable := expression

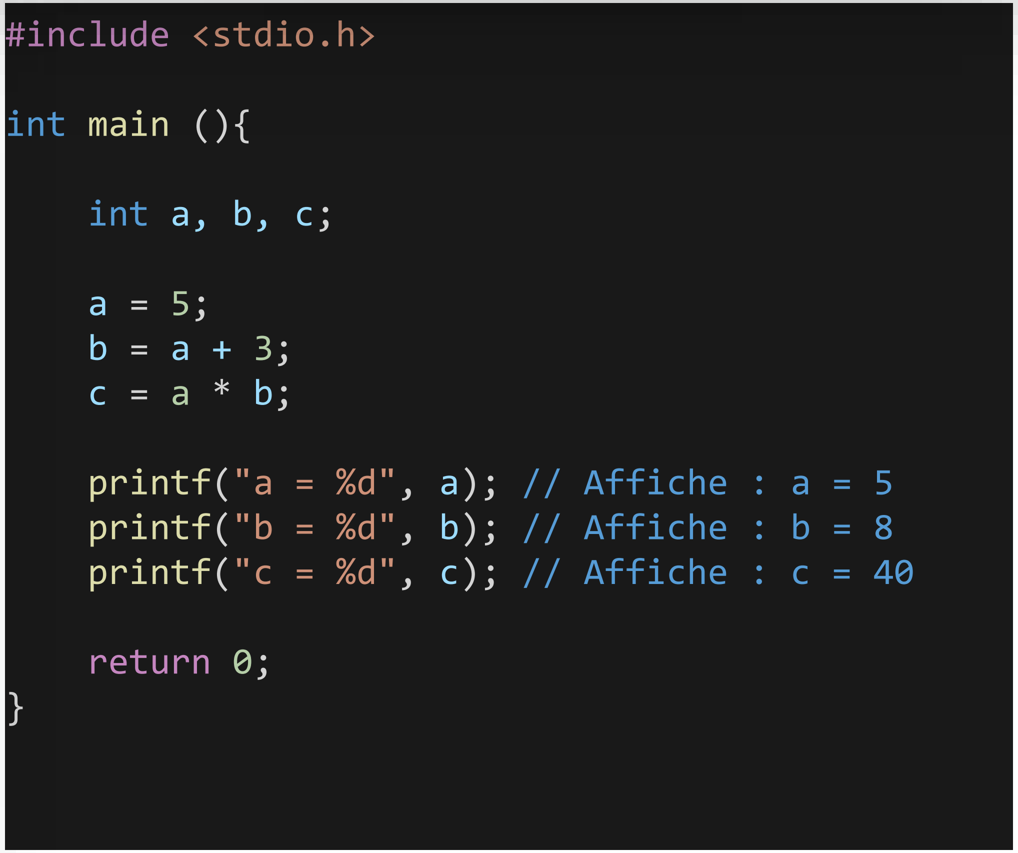
C

variable = expression

Examples



```
program Exemple_Const;

var     a, b, c : Integer;


begin
    //...
    a := 5;
    b := a + 3;
    c := a * b;

    WriteLn('a = ', a); // a = 5
    WriteLn('b = ', b); // b = 8
    WriteLn('c = ', c); // c = 40

    //...

end.
```



```
#include <stdio.h>

int main (){

    int a, b, c;

    a = 5;
    b = a + 3;
    c = a * b;

    printf("a = %d", a); // Affiche : a = 5
    printf("b = %d", b); // Affiche : b = 8
    printf("c = %d", c); // Affiche : c = 40

    return 0;
}
```

Expressions

- ✓ An ***expression*** is a set of values (***operands***), linked by ***operators***, and equivalent to a single value.
- ✓ An ***operator*** is a sign that connects two values to produce a result

Syntax

expression= *operand* ***operator*** *operand*

Expressions : Arithmetic

- ✓ ***Operands :*** **integer**, **real**.
- ✓ ***Operator*** :

  + : addition

  - : soustraction

  ***** : multiplication

  / : division

  DIV: quotient in Euclidean division

  MOD: remainder in Euclidean division (modulo)

Expressions : Arithmetic

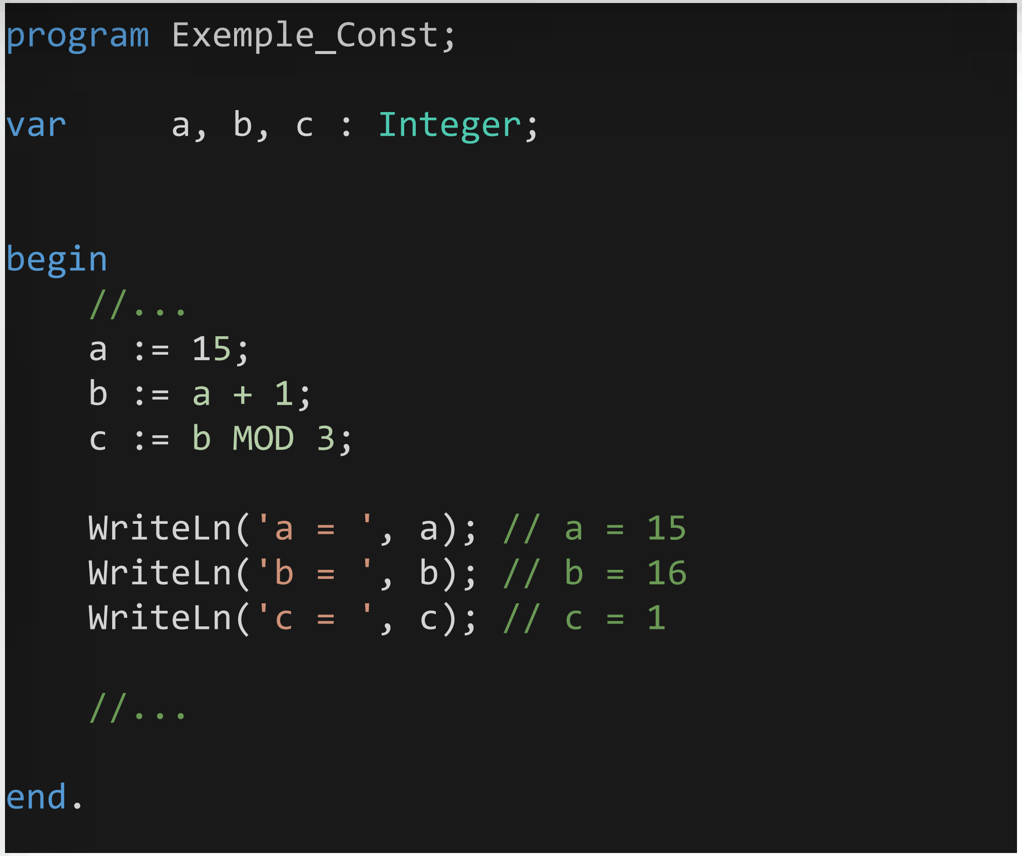
Basic Instructions

PASCAL

Déclaration

Opérateurs : **+**, **-** , ***** , **/** , **DIV** (Division entière), **MOD** (Modulo)

Exemples



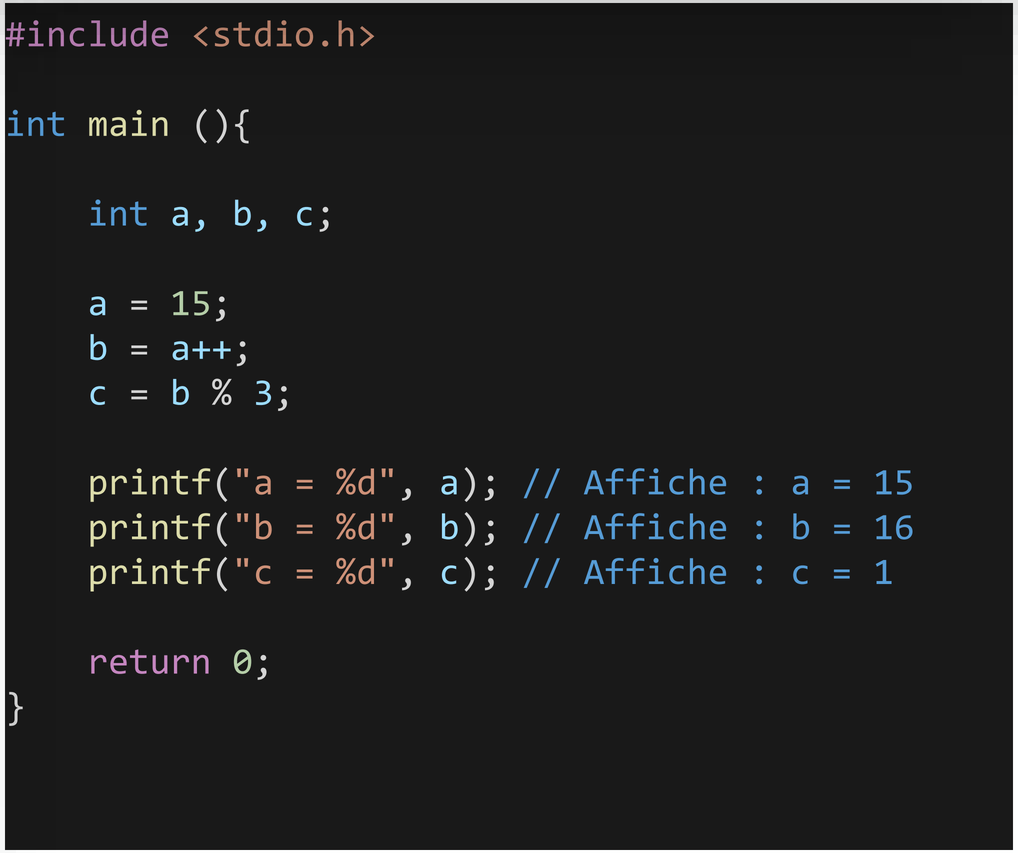
```
program Exemple_Const;

var     a, b, c : Integer;


begin
    //...
    a := 15;
    b := a + 1;
    c := b MOD 3;

    WriteLn('a = ', a); // a = 15
    WriteLn('b = ', b); // b = 16
    WriteLn('c = ', c); // c = 1

    //...

end.
```

C

Déclaration

Opérateurs : **+**, **-** , ***** , **/** (Division entière) , **%** (Modulo), **++** (Incrément),**--** (Décrément)

Exemples



```
#include <stdio.h>

int main (){

    int a, b, c;

    a = 15;
    b = a++;
    c = b % 3;

    printf("a = %d", a); // Affiche : a = 15
    printf("b = %d", b); // Affiche : b = 16
    printf("c = %d", c); // Affiche : c = 1

    return 0;
}
```

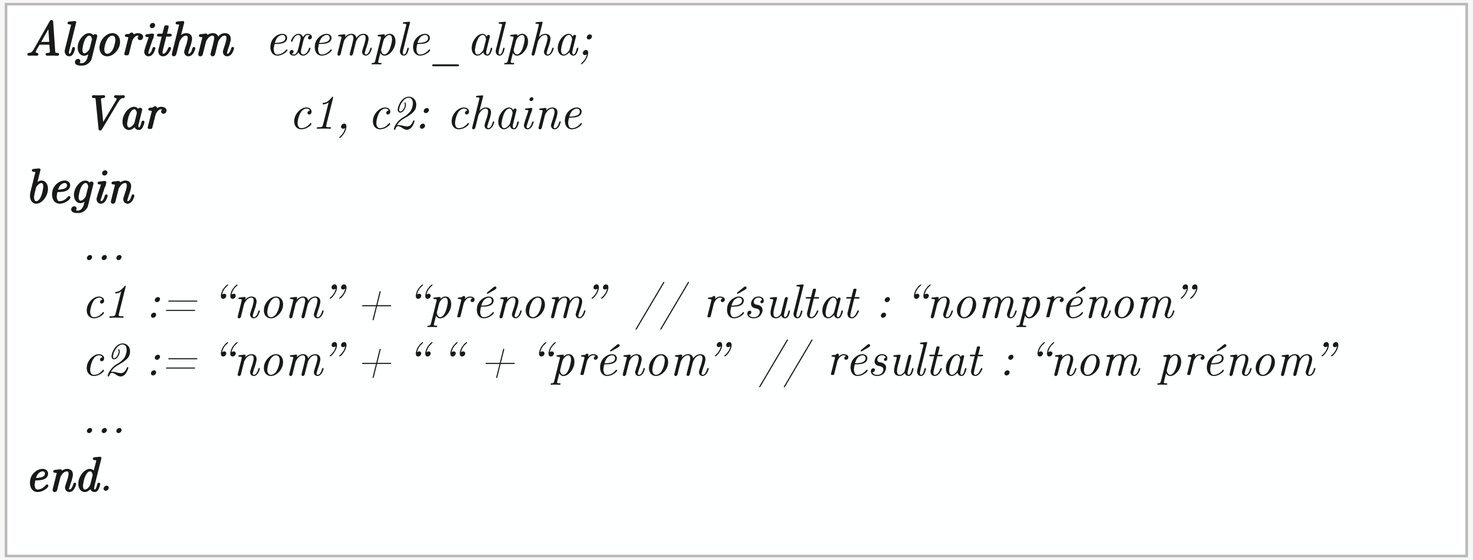
Basic Instructions

Expressions : Alphanumeric

- ✓ ***Operands :*** **caracters**, **strings**.
- ✓ ***Operator*** :

  + : concatenation

Examples



```
Algorithm  exemple_alpha;
    Var        c1, c2: chaine
begin
    ...
    c1 := “nom” + “prénom”  // résultat : “nomprénom”
    c2 := “nom” + “ “ + “prénom”  // résultat : “nom prénom”
    ...
end.
```

Basic Instructions

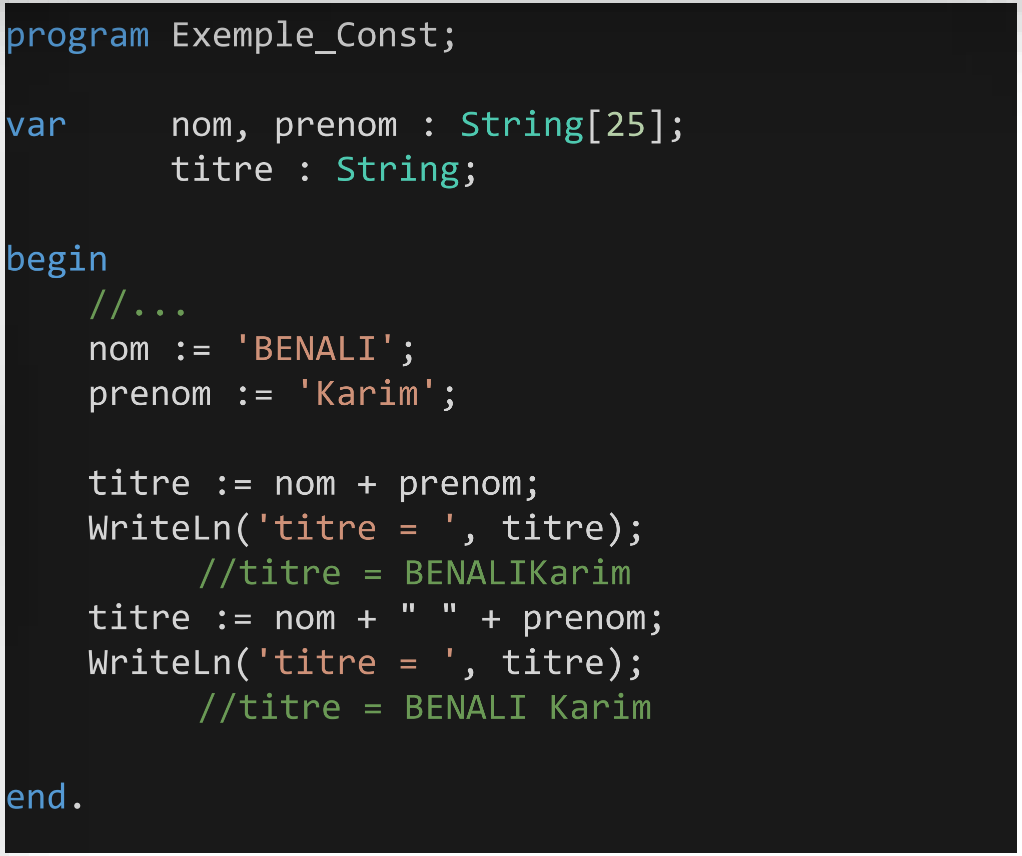
Expressions : Alphanumeric

Déclaration

Exemples

PASCAL

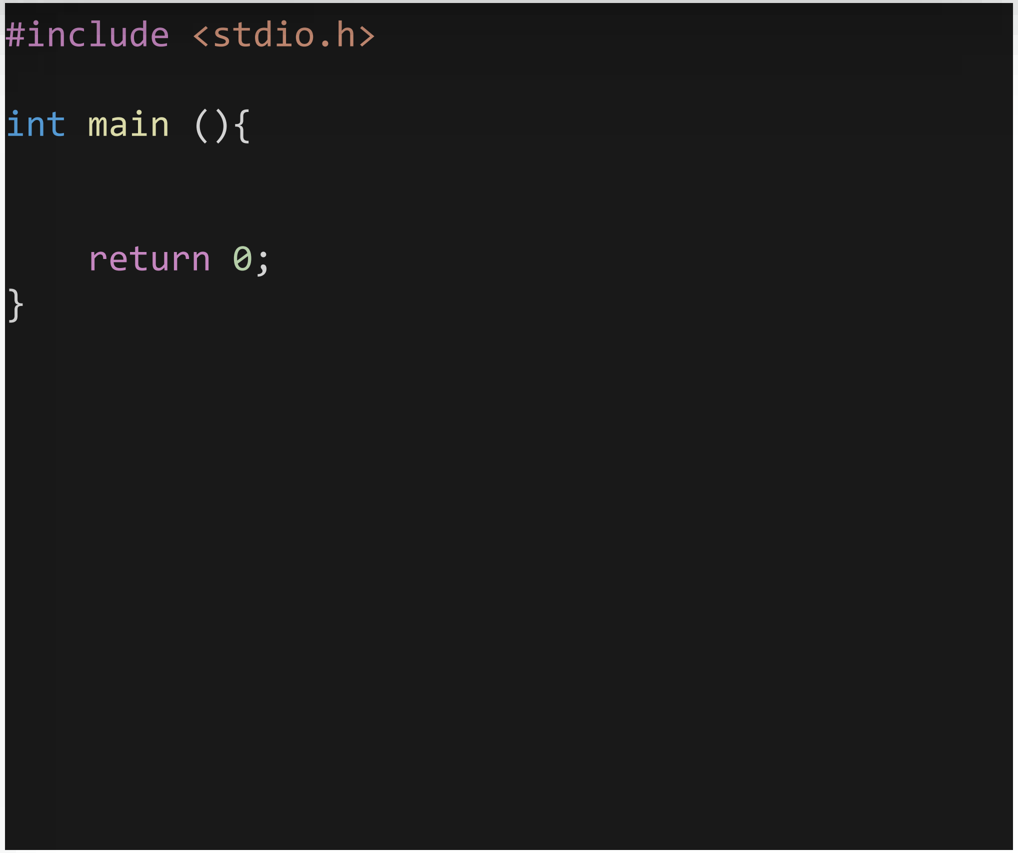
Opérateurs : **+**



```
program Exemple_Const;

var      nom, prenom : String[25];
         titre : String;

begin
    //...
    nom := 'BENALI';
    prenom := 'Karim';

    titre := nom + prenom;
    WriteLn('titre = ', titre);
         //titre = BENALIKarim
    titre := nom + " " + prenom;
    WriteLn('titre = ', titre);
         //titre = BENALI Karim

end.
```

C

Opérateurs :



```
#include <stdio.h>

int main (){


    return 0;
}
```

Expressions : logic

- ✓ *Operands :* **boolean (true, false)**.
- ✓ *Operator* :

  NON : negation

  AND : logic « AND »

  OR : logic « OR »

Basic Instructions

Expressions : logic

Déclaration

PASCAL

Opérateurs : **AND**, **OR**, **NOT**

C

Opérateurs : **&&** (AND), **||** (OR), **!** (NOT)

Exemples

```
program Exemple_Const;

var     nom, prenom : String[25];
        titre : String;

begin
    //...

end.
```

```
#include <stdio.h>

int main (){


    return 0;
}
```

Expressions : relational

- ✓ ***Operands :*** **integer**, **real, character**, **string**.
- ✓ ***Operator*** :

< : less than

> : greater than

= : equal

<= : less than or equal

>= : greater than or equal

<> : different

Expressions : relational

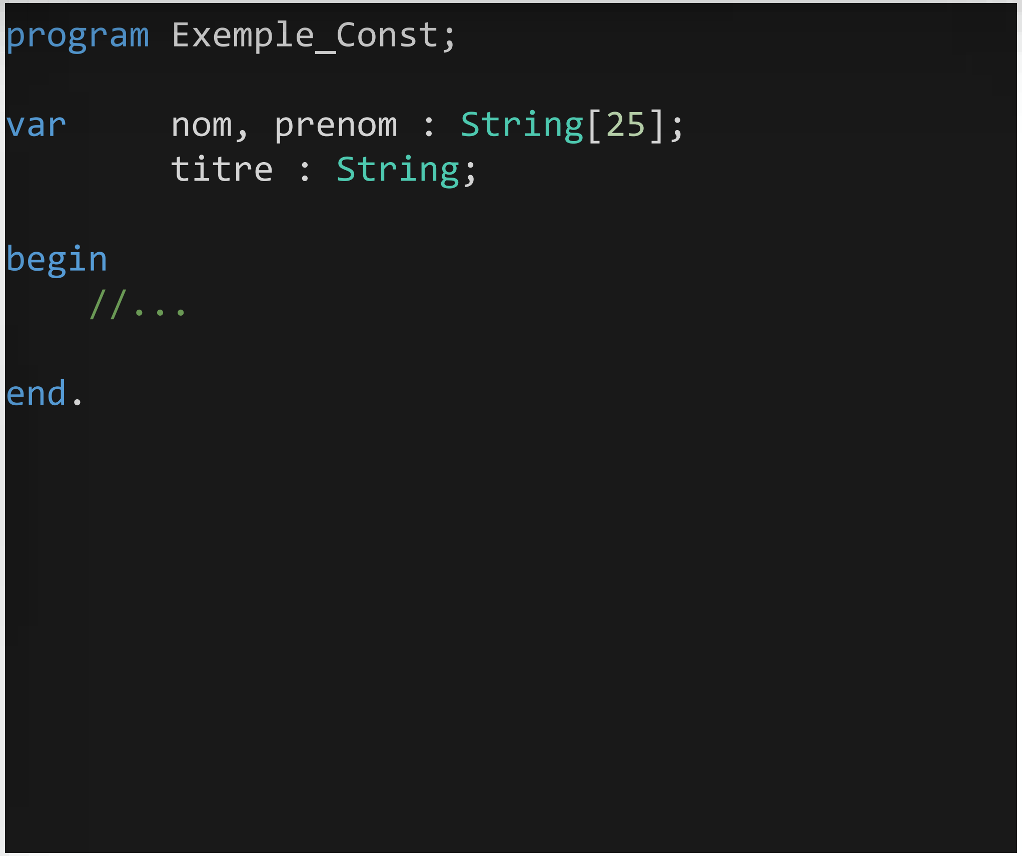
Basic Instructions

Déclaration

Exemples

PASCAL

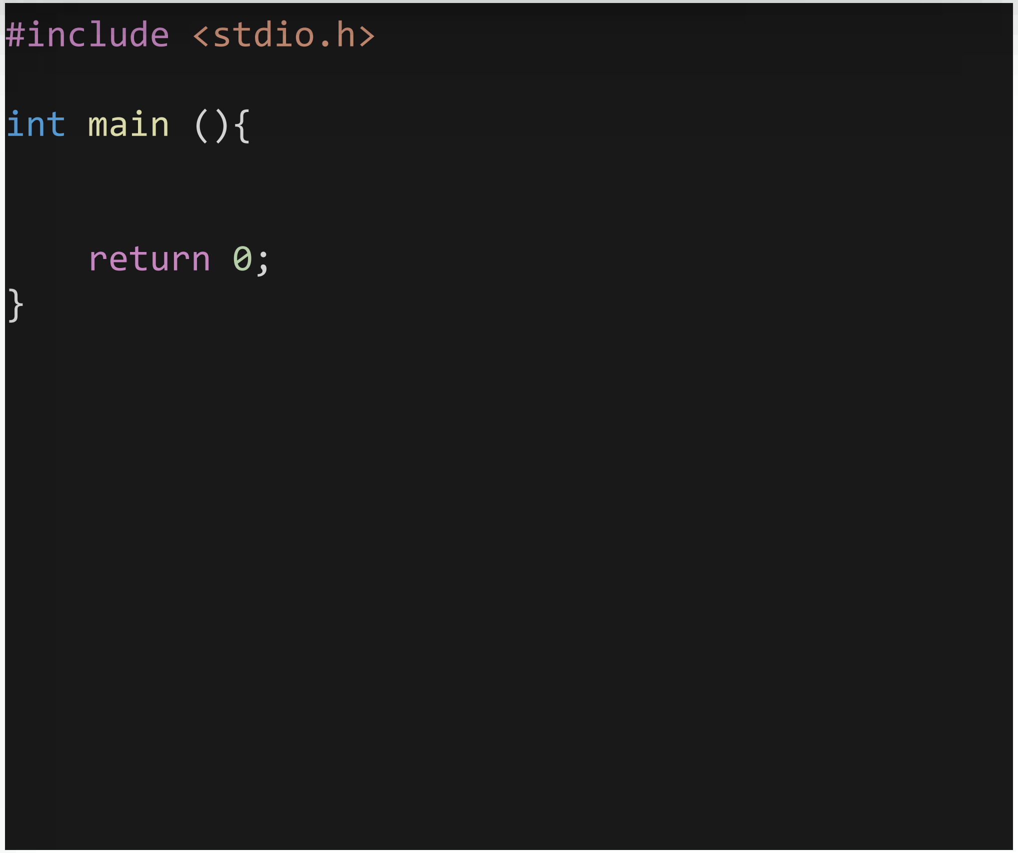
Opérateurs : <, <=, >, >=, =, <>



```
program Exemple_Const;

var     nom, prenom : String[25];
        titre : String;

begin
    //...

end.
```

C

Opérateurs : <, <=, >, >=, ==, !=



```
#include <stdio.h>

int main (){


    return 0;
}
```

Inputs / Outputs : Reading

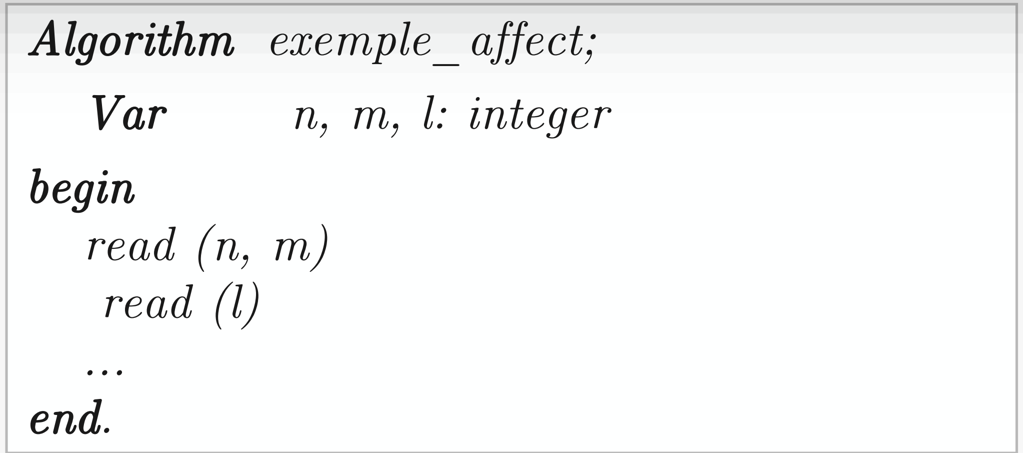
✓ Allows you to ***provide*** values from outside

- Values are entered using the ***keyboard***
- The P1, P2, …Pn are variables

Syntax

> ***read*** *(P1, P2, …, Pn)*

Examples



```
Algorithm exemple_affect;
    Var      n, m, l: integer
begin
    read (n, m)
     read (l)
    …
end.
```

Inputs/outputs: Reading

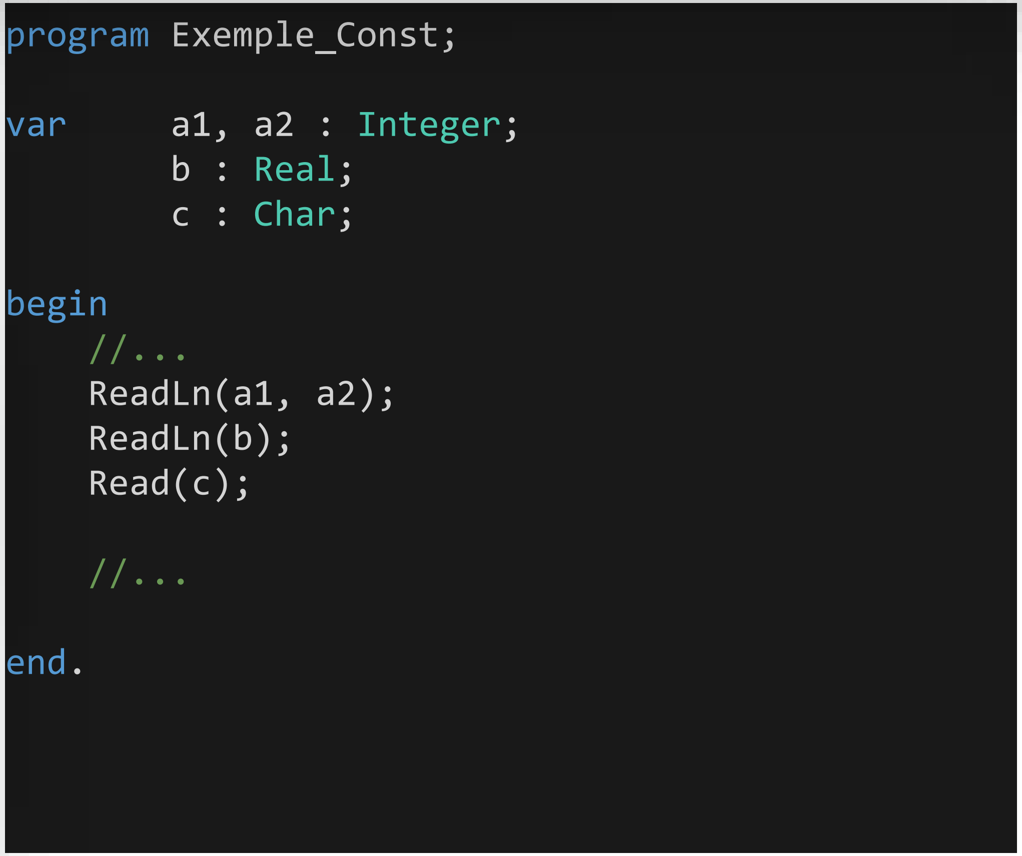
Basic Instructions

Déclaration

Exemples

PASCAL

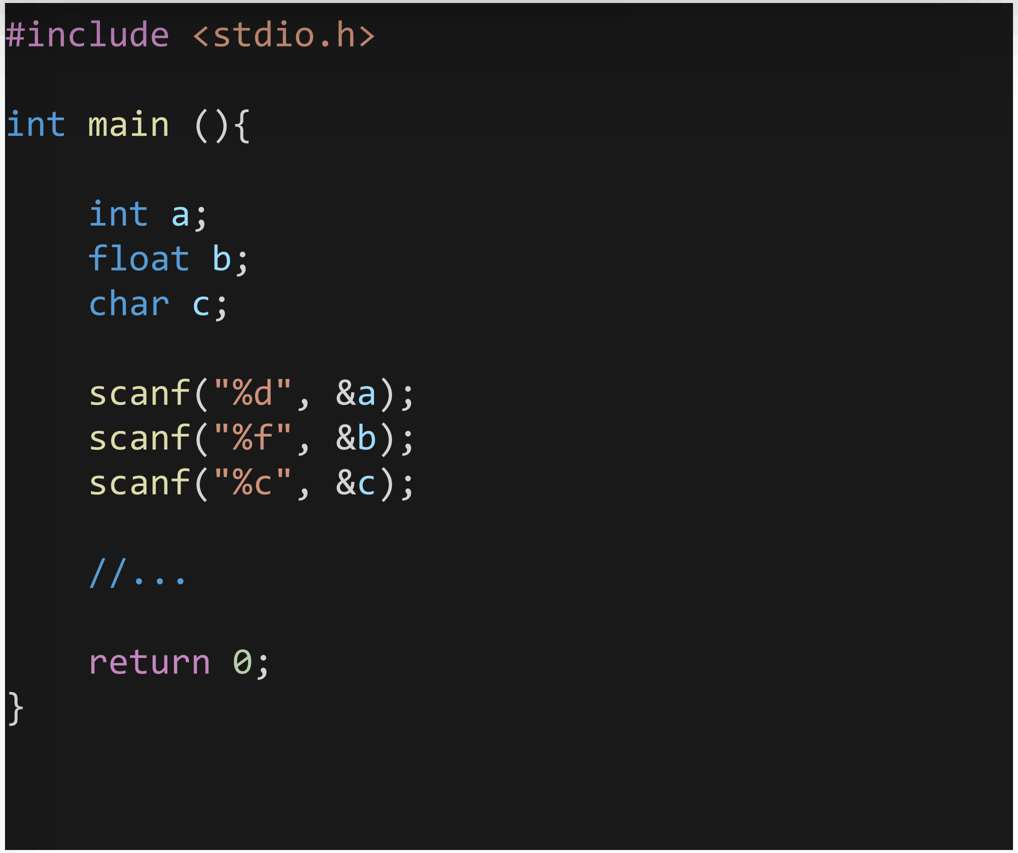
Syntaxe: **Read**, **ReadLn**



```pascal
program Exemple_Const;

var     a1, a2 : Integer;
        b : Real;
        c : Char;

begin
    //...
    ReadLn(a1, a2);
    ReadLn(b);
    Read(c);

    //...

end.
```

C

Syntaxe: **scanf** (<stdio.h)



```c
#include <stdio.h>

int main (){

    int a;
    float b;
    char c;

    scanf("%d", &a);
    scanf("%f", &b);
    scanf("%c", &c);

    //...

    return 0;
}
```

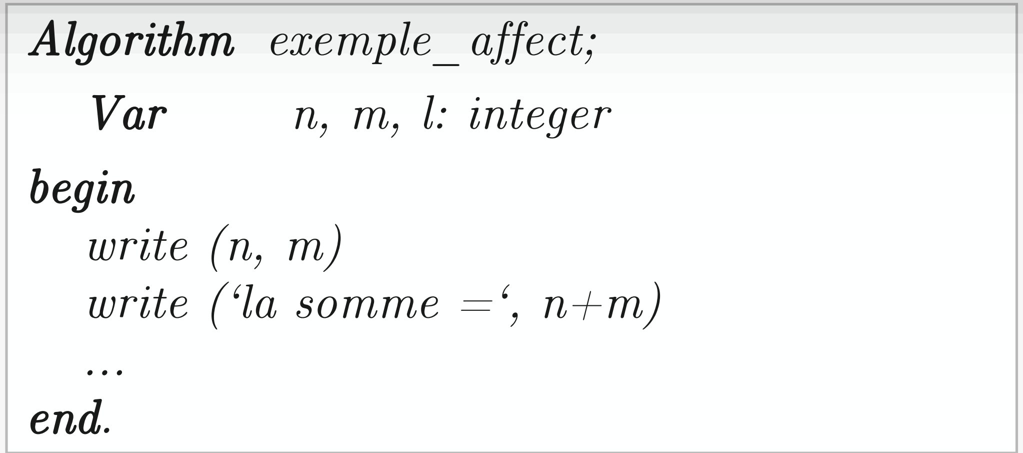
Inputs / Outputs : Ecriture

- ✓ Allows you to ***display*** the results of an algorithm
  - ➢ The values are displayed on the ***screen***
  - ➢ E1, E2, …En can be: variables, strings or expressions

Syntax

> ***Write*** *(E1, E2, …, En)*

Examples



```
Algorithm  exemple_affect;
   Var       n, m, l: integer
begin
   write (n, m)
   write ('la somme =', n+m)
   ...
end.
```

Basic Instructions

Inputs/outputs: Writing

Déclaration

PASCAL

Syntaxe: **Write**, **WriteLn**

C

Syntaxe: **printf** (<stdio.h)

Exemples

```pascal
program Exemple_Const;

var     a1, a2 : Integer;
        b : Real;
        c : Char;

begin
    //...

    WriteLn('Hello, World');
    WriteLn(a1, a2);
    WriteLn(b);
    WriteLn(c);
    WriteLn('Résultat est ', a1+a2);

    //...

end.
```

```c
#include <stdio.h>

int main (){

    int a;
    float b;
    char c;

    //...
    printf("Hello, World");
    printf("%d", a);
    printf("%f", b);
    printf("%c", c);
    printf("Résultat est %d", a);
    //...

    return 0;
}
```

Ministry of Higher Education and Scientific Research
Djilali BOUNAAMA University - Khemis Miliana(UDBKM)
Faculty of Science and Technology
Department of Mathematics and Computer Science

Chapter 2

Simple Sequential Algorithm

MI-L1-UEF121 : Algorithms and Data Structures I

Noureddine AZZOUZA

n.azzouza@univ-dbkm.dz