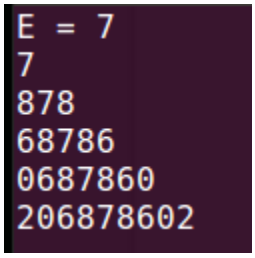
Solution avec un affichage non pyramidal




```
(*
* TASK : Bac TP 2015 (pyramide)
* LANG : Free Pascal
* Example: Affichage non pyramidal
*                pour e = 7
*)
```

```
procedure saisir(var e: byte);
begin
        repeat
                write('E = ');
                read(e);
        until e in [1..9];
end;
```

```
function calculer_c(ligne: string): integer;
var
        c, x, i, e: integer;
begin
        c := length(ligne);
        for i := 1 to length(ligne) do begin
                val(ligne[i], x, e);
                c := c + x;
        end;

        calculer_c := c mod 10;
end;
```

```
function est_divisible_7(ligne: string): boolean;
var
	i, signe, som, x, e, j: integer;
	tranche, tranche_inv: string;
begin
	som := 0;
	signe := 1;
	i := 1;
	while i <= length(ligne) do begin
		tranche := copy(ligne, i, 3);

		tranche_inv := '';
		for j := length(tranche)
downto 1 do
			tranche_inv :=
tranche_inv + tranche[j];

		val(tranche_inv, x, e);
		som := som + x * signe;
		signe := -signe;
		i +=3;

	end;
	est_divisible_7 := som mod 7 = 0;
end;
```

```
procedure afficher(e: byte);
var
	ligne, cs: string;
	c: integer;
begin
	str(e, ligne);
	writeln(ligne);
	while not (est_divisible_7(ligne)) do begin
		c := calculer_c(ligne);
		str(c, cs);
		ligne := cs + ligne + cs;
		writeln(ligne);
	end;

end;
```

```
//Programme principal
var
	e: byte;
begin
	saisir(e);
	afficher(e);

end.
```

Solution avec un affichage non pyramidal

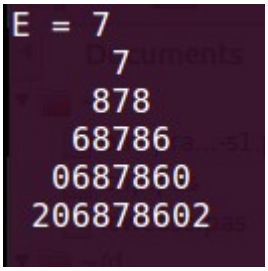


fig. #1

Pour pouvoir réaliser l'affichage en *fig. #1*, il faut savoir le nombre des lignes générées et donc les stocker dans un tableau.

```
(*
* TASK : Bac TP 2015 (pyramide)
* LANG : Free Pascal
* Exemple: Affichage pyramidal
*)
```

```
type
        tab = array[1..1000] of string;

procedure saisir(var e: byte);
begin
        repeat
                write('E = ');
                read(e);
        until e in [1..9];
end;
```

```
function calculer_c(ligne: string): integer;
var
        c, x, i, e: integer;
begin
        c := length(ligne);
        for i := 1 to length(ligne) do begin
                val(ligne[i], x, e);
                c := c + x;
        end;

        calculer_c := c mod 10;
end;
```

```
function est_divisible_7(ligne: string): boolean;
var
	i, signe, som, x, e, j: integer;
	tranche, tranche_inv: string;
begin
	som := 0;
	signe := 1;
	i := 1;
	while i <= length(ligne) do begin
		tranche := copy(ligne, i, 3);

		tranche_inv := '';
		for j := length(tranche)
downto 1 do
			tranche_inv :=
tranche_inv + tranche[j];

		val(tranche_inv, x, e);
		som := som + x * signe;
		signe := -signe;

		i +=3;
	end;
	est_divisible_7 := som mod 7 = 0;
end;
```

```
procedure stocker(var t: tab; var taille: integer; e: byte);
var
	ligne, cs: string;
	c: integer;
begin
	str(e, ligne);
	taille := 1;
	t[taille] := ligne;
	while not (est_divisible_7(ligne)) do begin
		c := calculer_c(ligne);
		str(c, cs);
		ligne := cs + ligne + cs;
		taille += 1;
		t[taille] := ligne;
	end;

end;
```

```
procedure afficher_pyramide(t: tab ; taille: integer);
var
	i, j: integer;
begin
	for i := 1 to taille do begin

		for j := taille downto i do
			write(' ');

		writeln(t[i]);
	end;
end;
```

```
//Programme principal
var
	e: byte;
	taille: integer;
	t: tab;
begin
	saisir(e);
	stocker(t, taille, e);
	afficher_pyramide(t, taille);
end.
```