GIT : dwarves.iut-fbleau.fr /git.iut/valarche / DEV-22-23

Rappel : Un langage de programmation comme intermédiaire entre nous et la machine

- une syntaxe + des outils qui transforment vers quelque chose de "lisible" par la machine
  - des mots réservés
  - des moyens de construire d'autres mots
  - des moyens de représenter des nombres
- une grammaire pour écrire des "phrases" correctes = programmes
- une sémantique pour qu'il y ait un lien entre ce qu'on écrit et ce qu'on pense que l'exécution va produire

variables

[nom, type, valeur, adresse]

boucle

while ( ) { }

for (; ;) { }

Déclaration d'une variable

Syntaxe

type nom [= valeur]

ex int t = 10

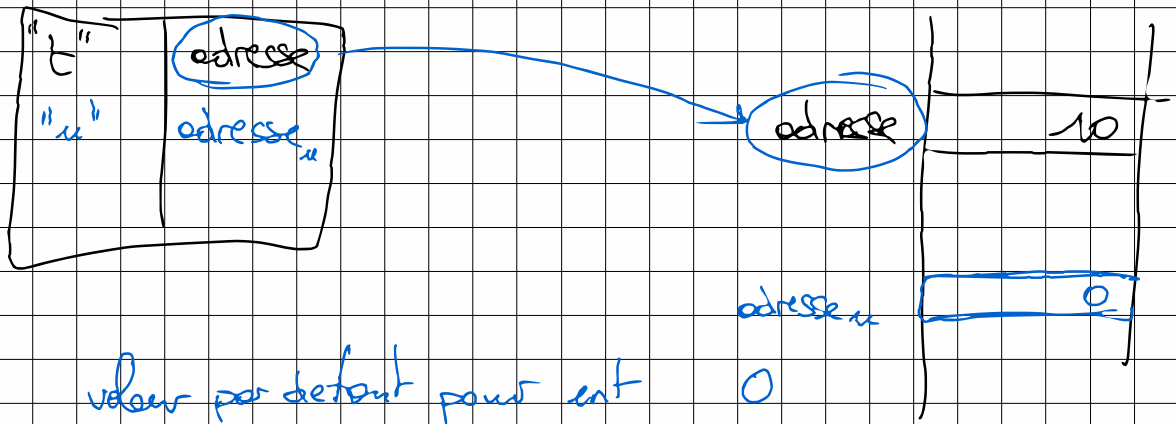
Sémantique

int u

("u", int, ?, ...)

("t", int, 10, ...)

création d'une "case" mémoire à une certaine adresse dans cette case la valeur est mise



valeur par défaut pour int 0

double

char

int

char c

Proposition : toujours mettre une valeur par défaut lors de la déclaration.

'a' 'A' 0 0.0

= affectation assignment

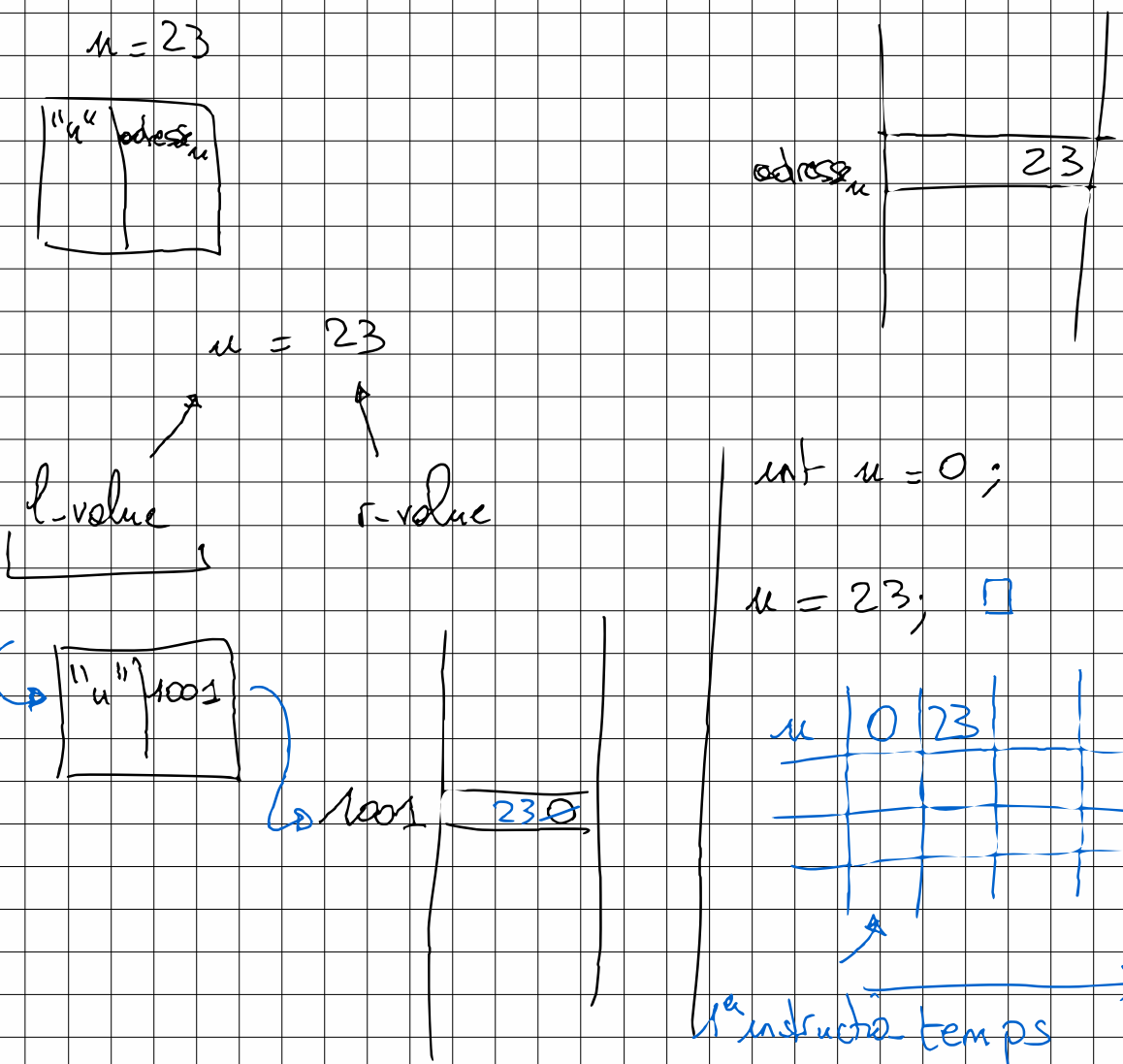
:= <- u <- 23

u = 23

Sémantique opérationnelle de l'affectation

modification de la valeur de la variable qui porte le nom

u u = 23 ("u", int, 23, ...)


x = 23
"x"
adresse x
adresse x
23
x = 23
l-value
r-value
"x"
1001
1001
230
int x = 0 ;
x = 23;
x
0
23
1ère instruction
temps

```
{ int t = 10; ①
  int u; ②
  u = t+1; ③
  u = 2*u; ④
  t = u+1; ⑤
  return EXIT_SUCCESS; }
```

10+1 = 11

t	10	10	10	10	23
u		?	11	22	22
	①	②	③	④	⑤

t+1) expression

aller chercher la valeur de t
à rajouter 1

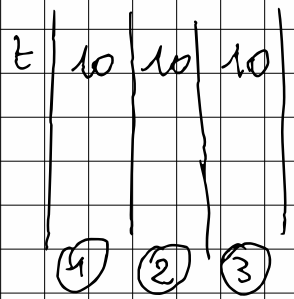
u = t+1

u = 2*u

t = u+1

```
{ int t = 10; ①

  t = 10; ②
  t = 10; ③
  return ___; }
```



② $t = 10$

```
{ int t=0;   (1)
  int u=4;   (2)
  t=1;       (3)
  while (u>=0) {
(4)   u=u-1;
(5)   t=2*t;
  }
  return EXIT_SUCCESS;
}
```

t	0	0	1	1	2	2	4	4	8	8
u		4	4	3	3	2	2	1	1	0
		(1)	(2)	(3)	(4)	(5)	(4)	(5)	(4)	(5)	(4)

t	8	16	16	32
u	0	0	-1	-1
	(5)	(4)	(5)	

si la condition est vraie while (condition) {
j'exécute ce qu'il y a
entre { } et je recommence } bloc
sinon je vais après le }