DIGITAL ARTS & ENTERTAINMENT

Overview course Programming 1

1. Variables 1
2. Variables 2
3. Variables 3
4. Conditionals
5. Iterations
6. Functions 1
7. Functions 2
8. Arrays
9. Strings – Game
10. Classes 1 – Encapsulation
11. Classes 2 – Static const

howest
De Hogeschool West-Vlaanderen
Lid van de Associatie Universiteit Gent

Variables

DIGITAL ARTS & ENTERTAINMENT

Content

- Bit and Byte
- Variables
- Expressions
- Assignment operator, compound assignment
- binary and unary operators
- console input and output

howest
De Hogeschool West-Vlaanderen
Lid van de Associatie Universiteit Gent

DIGITAL ARTS & ENTERTAINMENT

bit: binary digit

0 or 1

False	True
No	Yes
Off	On
-	+

- basic unit of information
- 2 states, can only have one of them active
- physically implemented with a two-state device

howest
De Hogeschool West-Vlaanderen
Lid van de Associatie Universiteit Gent

Bits and Bytes

1 bit: 2 possible values:

binary	decimal
0	0
1	1

Bits and Bytes

2 bits: 4 possible combinations

binary	decimal
0 0	0
0 1	1
1 0	2
1 1	3

DIGITAL ARTS & ENTERTAINMENT

Bits and Bytes

3 bits: 8 possible combinations

binary	decimal
0 0 0	0
0 0 1	1
0 1 0	2
0 1 1	3
1 0 0	4
1 0 1	5
1 1 0	6
1 1 1	7

howest
De Hogeschool West-Vlaanderen
Lid van de Associatie Universiteit Gent

DIGITAL ARTS & ENTERTAINMENT

Bits and Bytes

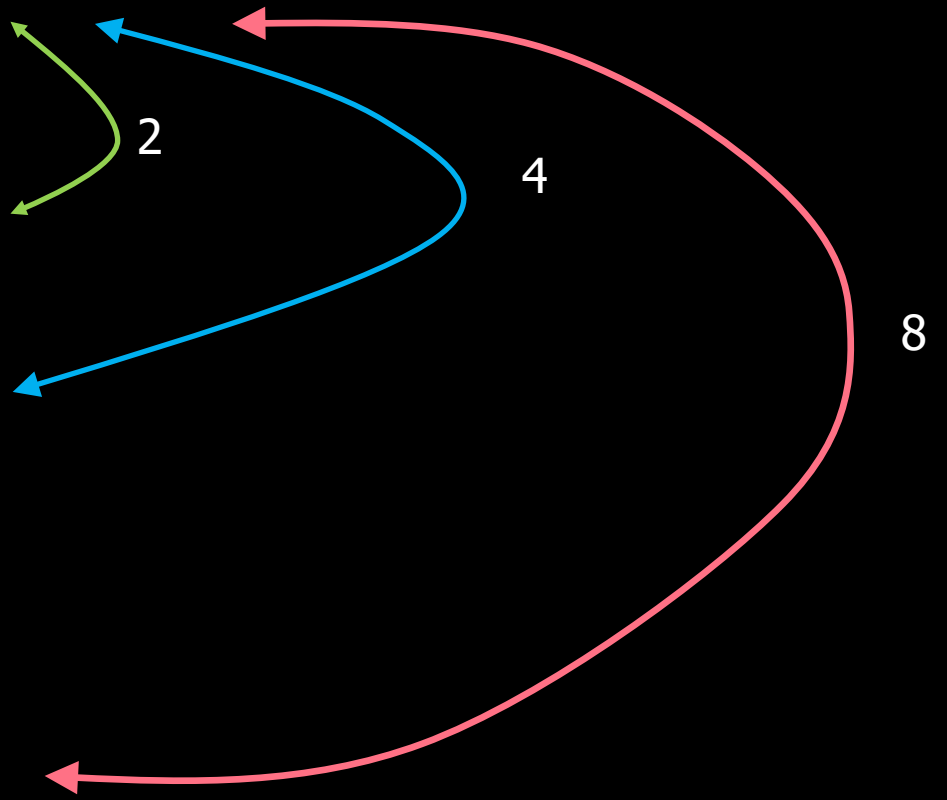
4 bits: 16 possible combinations

binary	decimal
0 0 0 0	0
0 0 0 1	1
0 0 1 0	2
0 0 1 1	3
0 1 0 0	4
0 1 0 1	5
0 1 1 0	6
0 1 1 1	7
1 0 0 0	8
1 0 0 1	9

Bits and Bytes

4 bits: 16 possible combinations

binary	decimal
0 0 0 0	0
0 0 0 1	1
0 0 1 0	2
0 0 1 1	3
0 1 0 0	4
0 1 0 1	5
0 1 1 0	6
0 1 1 1	7
1 0 0 0	8
1 0 0 1	9



Bits and Bytes

number of used bits vs possible represented values:

1 bit : $2^1 = 2$

2 bits: $2^2 = 4$

3 bits: $2^3 = 8$

4 bits: $2^4 = 16$

5 bits: $2^5 = 32$

6 bits: $2^6 = 64$

7 bits: $2^7 = 128$

8 bits: $2^8 = 256$

Bits and Bytes

A byte (Wikipedia):

- a unit of digital information.
- most commonly consists of eight bits.
- eight bits is a convenient power of two permitting the values 0 through 255 for one byte.

DIGITAL ARTS & ENTERTAINMENT

Bits and Bytes: from binary to decimal

highest significant bit

lowest significant bit

1	0	1	0	1	1	0	1

bit index	7	6	5	4	3	2	1	0
represented value	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
	128	64	32	16	8	4	2	1

1 x 128 + 0 x 64 + 1 x 32 + 0 x 16 + 1 x 8 + 1 x 4 + 0 x 2 + 1 x 1

128 + 32 + 8 + 4 + 1 = 173

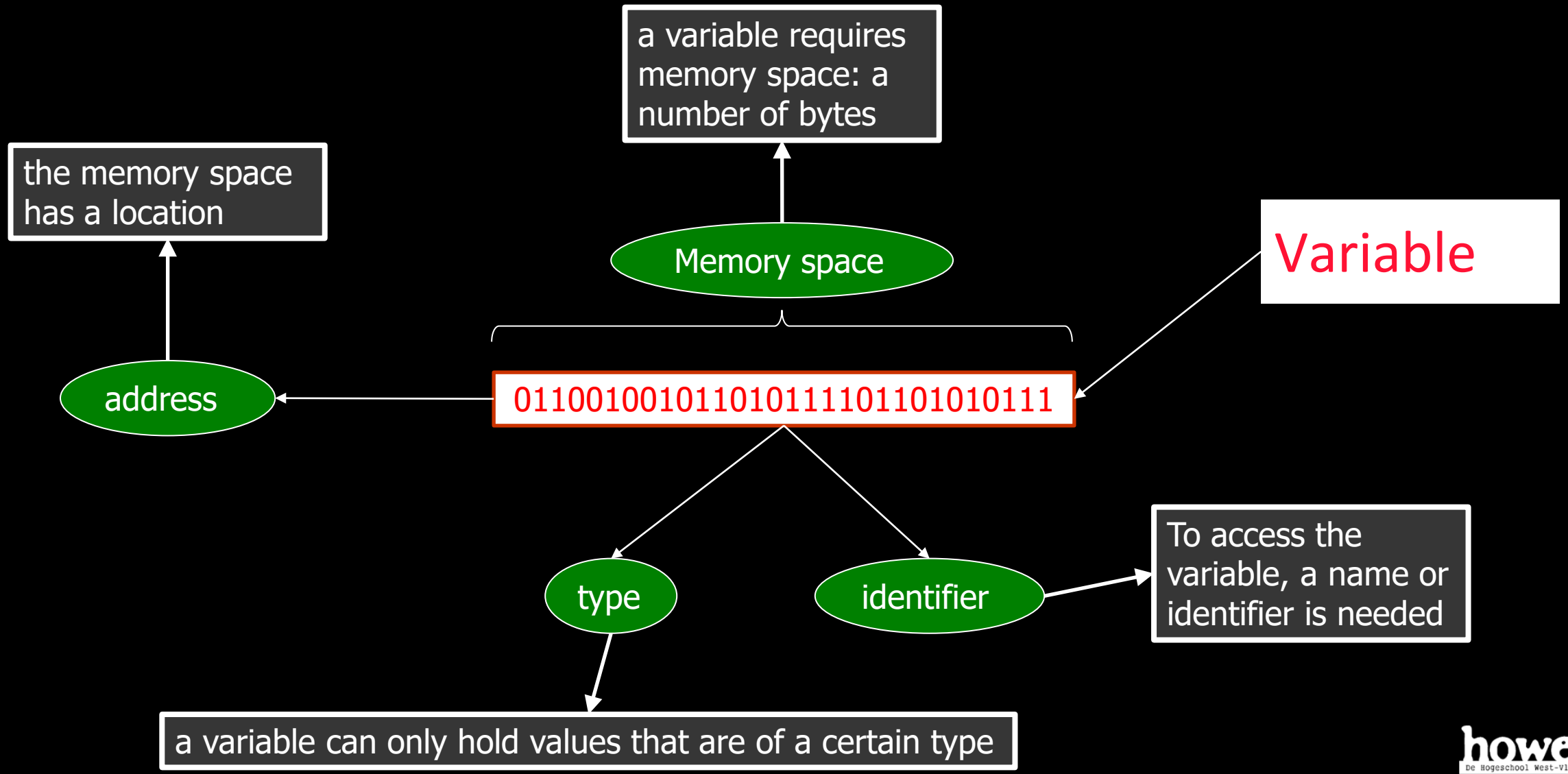
howest
De Hogeschool West-Vlaanderen
Lid van de Associatie Universiteit Gent

Bits and Bytes

Example:

- Color representation > see Windows Paint or Photoshop color picker:
  - a pixel can be represented using 3 (4) bytes:
    - red
    - green
    - blue
    - (alpha)
  - Resulting in 256 x 256 x 256 = 16 777 216 possible color values
- Colors in 3D API > OpenGl , DirectX
  - A color is represented by 4 floating point numbers with a range from 0 > 1
  - 4 x 4 bytes > 16 bytes

Variable


a variable requires memory space: a number of bytes
the memory space has a location
Memory space
Variable
address
01100100101101011101101010111
type
identifier
To access the variable, a name or identifier is needed
a variable can only hold values that are of a certain type

Variables and bytes

- a variable type uses 1, 2, 4 or 8 bytes

- 1 byte > 2^8 > 256 char
- 2 bytes > 2^{16} > 65536 char16_t
- 4 bytes > 2^{32} > 4 294 967 296 int, float
- etc. ...

Commonly used fundamental types

Category	Type	Bytes	Contents	range
integral	bool	1	represents a boolean value	true, false
	char	1	signed character representation	[-128 , 127]
	int	4	positive and negative numbers	[-2147483648, 2147483647]
floating	float	4	single precision floating point type	[1.8E-38, 3.4E+38]
	double	8	double precision floating point type	[2.2E-308, 1.8E+308]

Types of variables

- Many kinds of data:
  - Integral numbers
  - Decimal numbers
  - Boolean values
  - characters
- For every kind of data, there is a kind of variable: a *type*.

Types of variables

- The type of a variable determines what *kind of value* the variable can contain,

Types of variables

- The type of a variable determines what *kind of value* the variable can contain, and what *operations* can be performed on that value.

  E.g.. decimal number: can be added, divided, multiplied, etc.

Declaration and initialization of variables

- A variable has a name and a type, both are need to declare a variable.
- Before a variable can be used, it must be declared and should be initialized.
- In the declaration, first we decide on the type, then the variable name:

```
int number{ }; // uniform or brace initialization to default value
int number{ 0 }; // uniform or brace initialization
int number = 0; // copy initialization
```

These three declaration + initializations are equivalent. The first two are C++11 and later, the last is pre C++11.

char literal

- The char type is used to hold character values.
- Every character has a numeric representation: ASCII table.
  - e.g. The letter 'A' has the numeric value 65.
- To initialize a char with a letter there is no need to know its ascii value:
  - surround the letter with single quotes

```
char letter { 'A' }; // same as char letter { 65 };
```

When a letter is surrounded with single quotes, it is interpreted as its ascii numerical value.

Using the single quotes notation make you code more readable.

DIGITAL ARTS & ENTERTAINMENT

ascii table

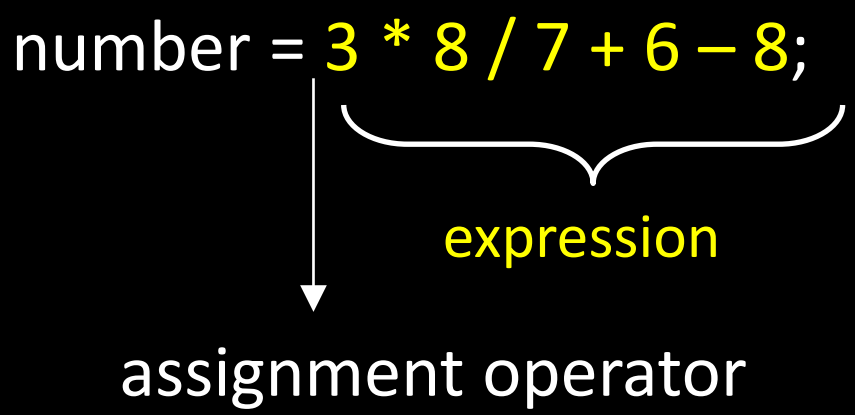
Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040	&#32;	Space	64	40	100	&#64;	@	96	60	140	&#96;	`
1	1	001	SOH (start of heading)	33	21	041	&#33;	!	65	41	101	&#65;	A	97	61	141	&#97;	a
2	2	002	STX (start of text)	34	22	042	&#34;	"	66	42	102	&#66;	B	98	62	142	&#98;	b
3	3	003	ETX (end of text)	35	23	043	&#35;	#	67	43	103	&#67;	C	99	63	143	&#99;	c
4	4	004	EOT (end of transmission)	36	24	044	&#36;	$	68	44	104	&#68;	D	100	64	144	&#100;	d
5	5	005	ENQ (enquiry)	37	25	045	&#37;	%	69	45	105	&#69;	E	101	65	145	&#101;	e
6	6	006	ACK (acknowledge)	38	26	046	&#38;	&	70	46	106	&#70;	F	102	66	146	&#102;	f
7	7	007	BEL (bell)	39	27	047	&#39;	'	71	47	107	&#71;	G	103	67	147	&#103;	g
8	8	010	BS (backspace)	40	28	050	&#40;	(	72	48	110	&#72;	H	104	68	150	&#104;	h
9	9	011	TAB (horizontal tab)	41	29	051	&#41;	)	73	49	111	&#73;	I	105	69	151	&#105;	i
10	A	012	LF (NL line feed, new line)	42	2A	052	&#42;	*	74	4A	112	&#74;	J	106	6A	152	&#106;	j
11	B	013	VT (vertical tab)	43	2B	053	&#43;	+	75	4B	113	&#75;	K	107	6B	153	&#107;	k
12	C	014	FF (NP form feed, new page)	44	2C	054	&#44;	,	76	4C	114	&#76;	L	108	6C	154	&#108;	l
13	D	015	CR (carriage return)	45	2D	055	&#45;	-	77	4D	115	&#77;	M	109	6D	155	&#109;	m
14	E	016	SO (shift out)	46	2E	056	&#46;	.	78	4E	116	&#78;	N	110	6E	156	&#110;	n
15	F	017	SI (shift in)	47	2F	057	&#47;	/	79	4F	117	&#79;	O	111	6F	157	&#111;	o
16	10	020	DLE (data link escape)	48	30	060	&#48;	0	80	50	120	&#80;	P	112	70	160	&#112;	p
17	11	021	DC1 (device control 1)	49	31	061	&#49;	1	81	51	121	&#81;	Q	113	71	161	&#113;	q
18	12	022	DC2 (device control 2)	50	32	062	&#50;	2	82	52	122	&#82;	R	114	72	162	&#114;	r
19	13	023	DC3 (device control 3)	51	33	063	&#51;	3	83	53	123	&#83;	S	115	73	163	&#115;	s
20	14	024	DC4 (device control 4)	52	34	064	&#52;	4	84	54	124	&#84;	T	116	74	164	&#116;	t
21	15	025	NAK (negative acknowledge)	53	35	065	&#53;	5	85	55	125	&#85;	U	117	75	165	&#117;	u
22	16	026	SYN (synchronous idle)	54	36	066	&#54;	6	86	56	126	&#86;	V	118	76	166	&#118;	v
23	17	027	ETB (end of trans. block)	55	37	067	&#55;	7	87	57	127	&#87;	W	119	77	167	&#119;	w
24	18	030	CAN (cancel)	56	38	070	&#56;	8	88	58	130	&#88;	X	120	78	170	&#120;	x
25	19	031	EM (end of medium)	57	39	071	&#57;	9	89	59	131	&#89;	Y	121	79	171	&#121;	y
26	1A	032	SUB (substitute)	58	3A	072	&#58;	:	90	5A	132	&#90;	Z	122	7A	172	&#122;	z
27	1B	033	ESC (escape)	59	3B	073	&#59;	;	91	5B	133	&#91;	[	123	7B	173	&#123;	{
28	1C	034	FS (file separator)	60	3C	074	&#60;	<	92	5C	134	&#92;	\	124	7C	174	&#124;	|
29	1D	035	GS (group separator)	61	3D	075	&#61;	=	93	5D	135	&#93;	]	125	7D	175	&#125;	}
30	1E	036	RS (record separator)	62	3E	076	&#62;	>	94	5E	136	&#94;	^	126	7E	176	&#126;	~
31	1F	037	US (unit separator)	63	3F	077	&#63;	?	95	5F	137	&#95;	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

howest
De Hogeschool West-Vlaanderen
Lid van de Associatie Universiteit Gent

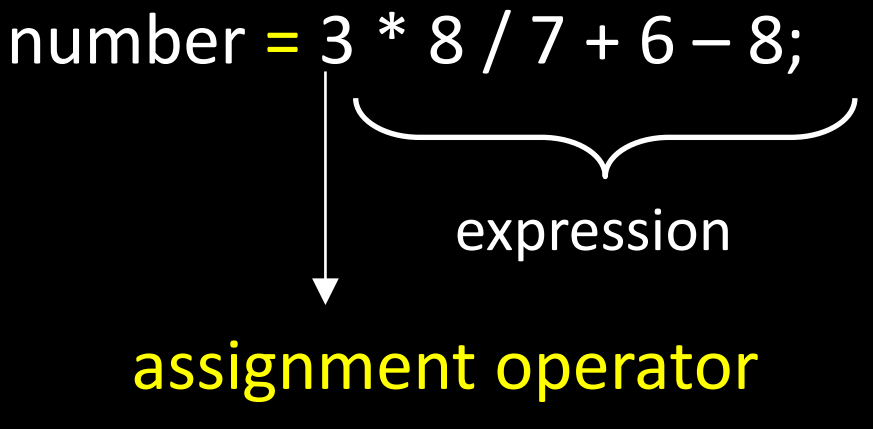
Expression

- Operator: do operations on variables. Examples: + - * / %
- Operand: the variables on which the operators are performed
- Definition: An *expression* is "a sequence of operators and operands that specifies a computation" (cppreference.com)
- Example: 3 * 8 / 7 + 6 – 8

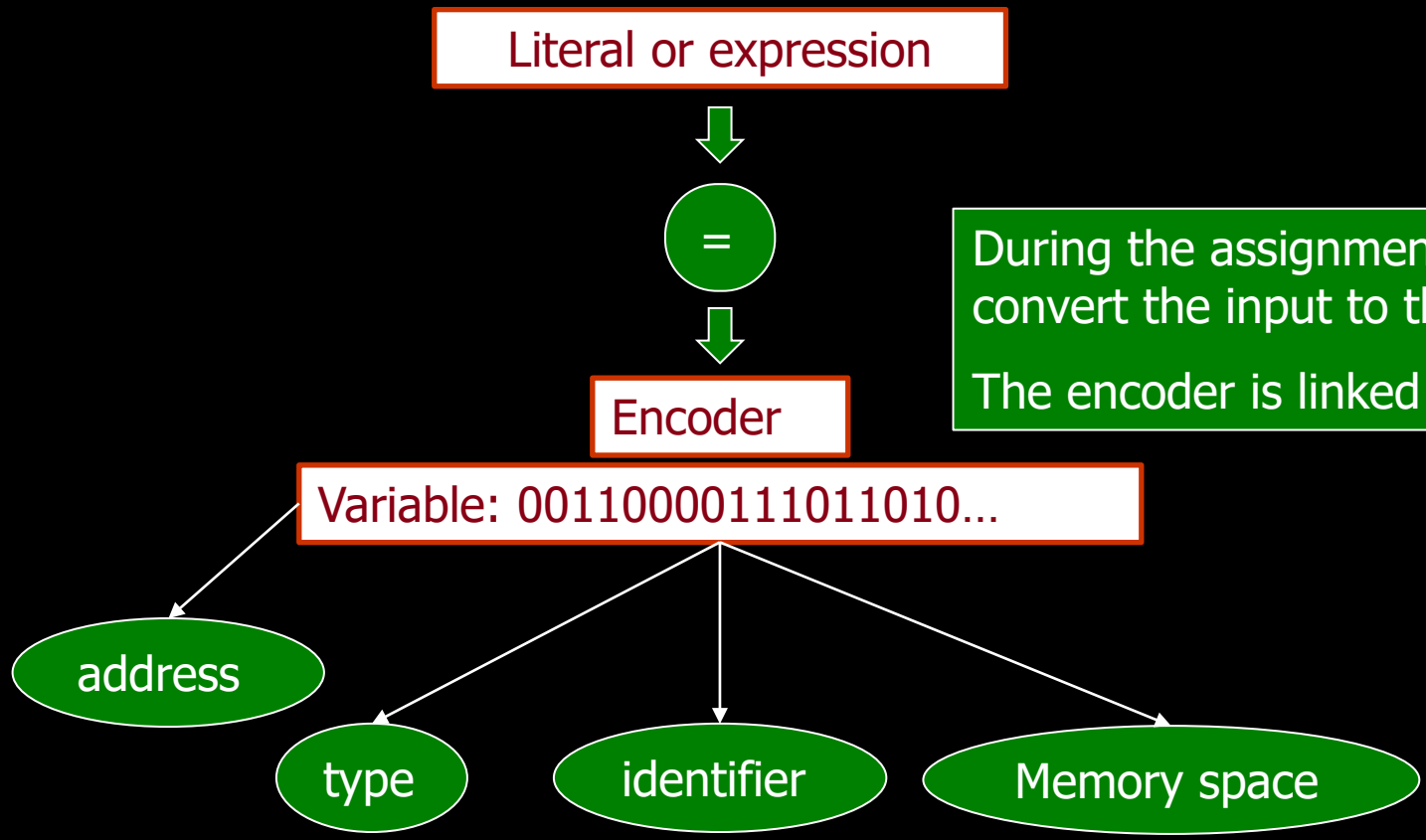


Assignment operator: "="

- Converts and writes a value in a variable
  - Converts: example: from decimal to binary
  - Writes: the different bits of the variable get a value 0 or 1
  - NOT the math “equals”
- Example:
  - first the expression is processed
  - next, the resulting value is converted to an integer binary number and stored in the variable



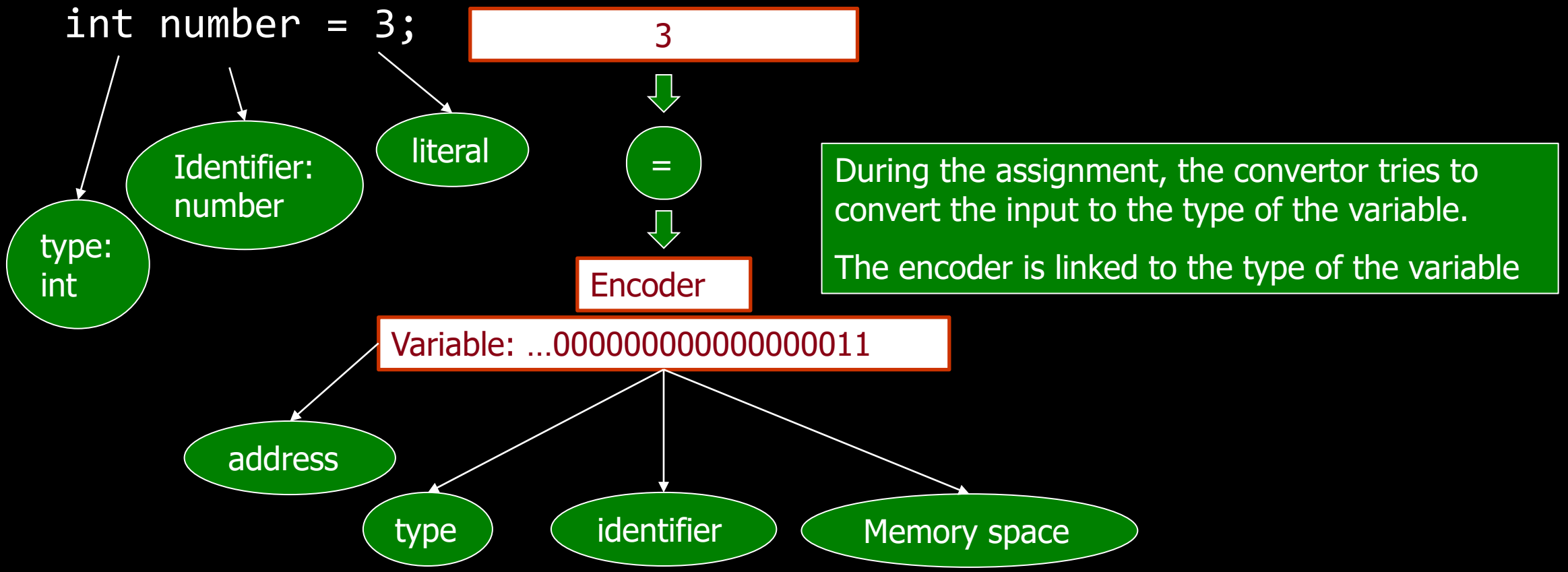
Assignment operator: "="



During the assignment, the convertor tries to convert the input to the type of the variable.

The encoder is linked to the type of the variable

Assignment operator: "="



During the assignment, the convertor tries to convert the input to the type of the variable.

The encoder is linked to the type of the variable

Operators (+ - * /)

- Precedence and Order of Evaluation: (cppreference)
  - Similar to mathematics:
  - First * / %
  - Then + -
  - Round brackets or parentheses ( ) can be used to group calculations. Grouped operations have priority.

The divide operator /

- Division with integer types : division in which the fractional part (remainder) is discarded.
  - `8/4  -> 2`
  - `8/3  ->  2.6666 -> 2`
  - `8/2  ->  4`
  - `16/9 -> 1.77777 -> 1`

The divide operator /

- Division with floating types : division in which the fractional part remains.
  - `8.0/4.0   -> 2`
  - `8.0/3.0  ->  2.6666`
  - `16.0/9.0 -> 1.77777`

Binary operators:

One operand on each side of the operator

```
int a{10}, b{20}, c{};
c = a + b;
c = a / b;
a = b * c;
```

Compound assignment operators:

Provide a shorter syntax for assigning the result of an arithmetic operator. They perform the operation on the two operands before assigning the result to the first operand.

```
int a{10}, b{20}, c{};
a += b;  // same as a = a + b;
a /= b;  // same as a = a / b;
b *= c;  // same as b = b * c;
```

Unary operators:

Are operators that act upon a single operand to produce a new value.

```
int a{10}, b{20}, c{};
++a;  // same as a = a + 1;
--a;  // same as a = a - 1;
```

Demo

- Declaration and initialization of variables
- Assignment operator
- Expressions (+, -, *, /)
- Reading and writing from and to the console

DIGITAL ARTS & ENTERTAINMENT

References

- http://www.cplusplus.com/doc/tutorial/variables/
- http://www.learncpp.com/cpp-tutorial/11-structure-of-a-program/

howest
De Hogeschool West-Vlaanderen
Lid van de Associatie Universiteit Gent