

CHAPTER 2 READ W1

HELLO

6

/* My first C program

* Date: ...

* Author: Bernd Hamann

* Input: None

* Output: "Hello."

* Algo: Prgm. prints "Hello."

* using the "genlib.h" library

*/

COMMENTS

NO SPACE HERE
#include <stdio.h>
#include "genlib.h"

LIBRARIES

main()

{ printf("Hello, \n");
}

function
call

argument
(a string)

MAIN PRGM.

ADD 2 INTS.

/* Adding 2 ints.

* Inp: ...

* Outp: ...

* Algo: ...

*/

#include <stdio.h>

#include "genlib.h"

#include "simpio.h"

main()

{ int no1, no2, sum; /* Declaration */

printf("Input 1st number = ");

no1 = GetInteger();

↑
"assignment"

W1

7

```
printf("Input 2nd number = ");
no2 = GetInteger();
sum = no1 + no2;
```

↑ compute this value
 ↑ assign this value
 sum will have assigned value

"PLACE-HOLDERS"

```
printf("The sum of %d and %d is %d\n",
      no1, no2, sum);
```

• DATA TYPES

"Certain types of operations/operators apply to specific data types."

E.g. "*" :

3 * 4 ✓

"Claire" * "Phil"

???

TYPE	DECLARATION	EX.
- decimal integer	<u>int</u>	... -2, -1, 0, 1, 2, ..., MAX-INT
- floating-pt. no.	<u>double</u>	{?} $2^{15}-1$ ↑ Ex: 0.001 9.99 7.88E-9 (= 7.88 × 10 ⁻⁹)

- String (in "genlib.h")

= string of characters and digits

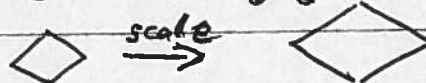
"Eric", "E1ric", "alpha12BETH"

- GENERAL: DEFINE OWN DATA TYPES & OPERATIONS

e.g.: - Vector addition:

$a_1 \xrightarrow{1b} c = a_1 \oplus b$

- scaling a polygon



W 1

8

/* Add 2 real no's: */

```
... include ...  
main()  
{ double no1, no2, sum;  
  printf ("1st no = ");  
  no1 = GetReal();  
  printf ("2nd no = ");  
  no2 = GetReal();  
  sum = no1 + no2;  
  printf ("The sum of %g and %g is %g \n",  
          no1, no2, sum);  
}
```

/* STRING EX. */

```
... include ...  
main()  
{ string name;
```

```
  printf ("Type your name = ");
```

INPUT

```
  name = GetLine();
```

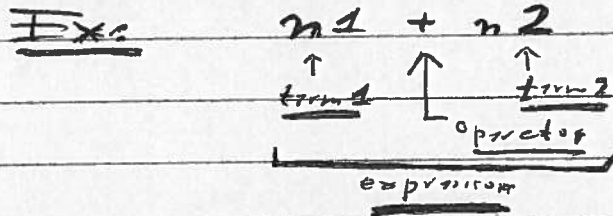
```
  printf ("Hello, %s. \n", name);
```

OUTPUT

```
}
```

• Expressions

"An expression consists of terms & operators."



term = constant

3.1416

- variable

π

- function call

$\sin(x)$

- expr. in parentheses

$(x+y)$

→ Constants:

- INT : 10, 1, 1000000

- Float : 0.08, 1000.00, 2.99E-9

- STRING : "hello"

→ Variables ("place-holders"):

→ name type value

↑ initially UNDEFINED !!!



- starts with letter or "_"
e.g.: no 1, _no1, No1

- continues with letters, digits or "_"

- must NOT be a key word like
if, for, while etc.

• ASSIGNMENT

Var = expr;

- declaration: int n1; state: $n1$?

→ initialization: n1 = 10; state: $n1$ 10

• OPERATORS / OPERANDS

- binary +, -, /, *

expression
 $(2 * x) + (3 * y)$
 ↑ ↑
 sub-expressions

- unary = e.g., -5, -x

- INT op Float: int n1;
double n2;

$n1 + n2$ ⇒ double!

- INT DIVISION & REMAINDER operators

$$0 \% 4 \rightarrow 0$$

$$1 \% 4 \rightarrow 1$$

$$2 \% 4 \rightarrow 2$$

$$4 \% 4 \rightarrow 0$$

" modulo "

$$9 / 4 \rightarrow 2$$

$$9 \% 4 \rightarrow 1$$

$$\Rightarrow 9 = 2 * 4 + 1$$

Precedence

- L
E
S
S
P
R
I
O
R
I
T
Y
- 0) Parantheses ()
 - 1) Unary op's -x ← TYPE-CASTING
 - 2) Multiplication *, /
 - 3) Addition +, -
 - 4) "From left to right"
- Eval. from left to right when equal preced.

Ex: → $(2 * x) + (3 * 4)$
 $\hat{=}$ $2 * x + 3 * 4$

→ $10 - 5 - 2 = (10 - 5) - 2 = \underline{\underline{3}}$
 ~~$10 - (5 - 2) = 7$~~

→ $avg = (n1 + n2) / 2.0;$

→ $8 * \underbrace{(7 - 6 + 5)}_6 \% \underbrace{(4 + 3 / 2)}_5 - 1$

$\underbrace{\hspace{10em}}_{48}$
 $\underbrace{\hspace{15em}}_3$
 $\underbrace{\hspace{20em}}_{\underline{\underline{2}}}$

• Type Conversion

EX: • $1 + 2.3$

$\xrightarrow{\text{conv.}} 1.0 + 2.3$

INTs are

'type-cast' to
DOUBLES

• int n;

n = 1.9999;

→ n will have value 1
"truncation"

• TYPE-CASTING

→ "Force a certain type"

→ Precedence higher than multiplication

EX:

double quotient;
int num, denom;

quotient = (double) num /
 (double) denom;

OR ... = (double) num / denom;

OR ... = num / (double) denom;

ex: $5/3 \Rightarrow$ value 1 ✓
 $5/3.0 \Rightarrow$ value 1.666666... ✓