Programming with C I

Fangtian Zhong
CSCI 112

Gianforte School of Computing
Norm Asbjornson College of Engineering
E-mail: fangtian.zhong@montana.edu

2024.08.30

Let’s write a C program

That stores an int, double, and char variable, and prints them all out.

Placeholders in format string

Placeholder	Variable Type	Function Use
%c	char	printf/scanf
%d	int	printf/scanf
%f	double	printf
%lf	double	scanf

The scanf Function

- Copies data from the standard input device (usually the keyboard) into a variable.

```
scanf( “%lf” , &miles);
scanf( “%c%c%c” , &letter_1, &letter_2, &letter_3);
```

- Must pass address of variable to store using the addressof operator (&)

The return Statement

- Last line in the main function.
- Transfers control from your program to the operating system.
- The value 0 indicates that your program executed without an error.

```
return (0);
```

Arithmetic Operators

Arithmetic Operator	Meaning	Example
+	addition	5 + 2 is 7 5.0 + 2.0 is 7.0
–	subtraction	5 – 2 is 3 5.0 – 2.0 is 3.0
*	multiplication	5 * 2 is 10 5.0 * 2.0 is 10.0
/	division	5.0 / 2.0 is 2.5 5 / 2 is 2
%	remainder	5 % 2 is 1

Type casting

- converting an expression to a different type by writing the desired type in parentheses in front of the expression

```
int x = 5;
double y = (double) x;
```

Rules for Evaluating Expressions

- **Parentheses rule**
  - all expression must be evaluated separately
  - nested parentheses evaluated from the inside out
  - innermost expression evaluated first
- **Operator precedence rule**
  - unary +, - first (setting sign)
  - *, /, % next
  - binary +, - last
- **Note prefix and postfix increment/decrement!**
  - ++a and --a are executed before value is used
  - a++ and a-- are executed after value is used

Rules for Evaluating Expressions

Right Associativity

- Unary operators in the same subexpression and at the same precedence level are evaluated right to left.

Left Associativity

- Binary operators in the same subexpression and at the same precedence lever are evaluated left to right.

Figure Evaluation Tree for area = PI * radius * radius;

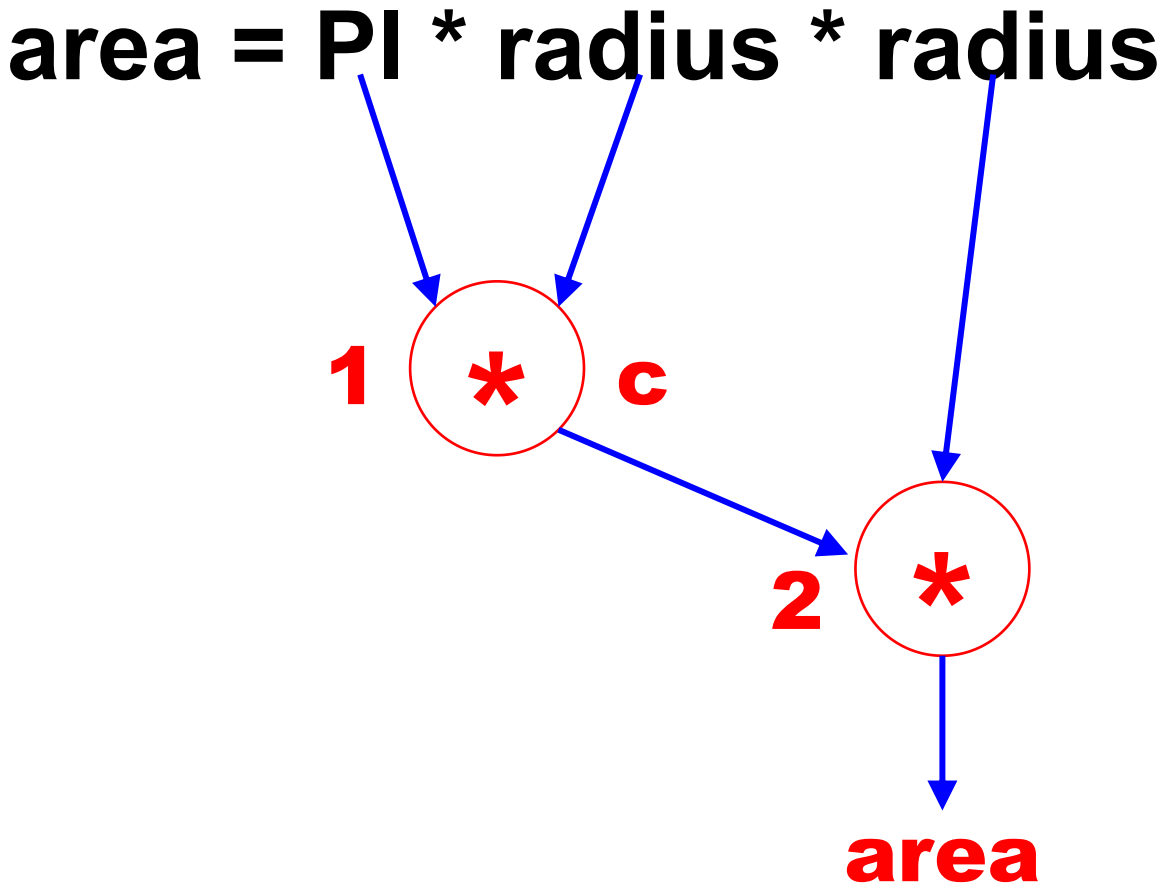


Figure Step-by-Step Expression Evaluation

area = PI * radius * radius

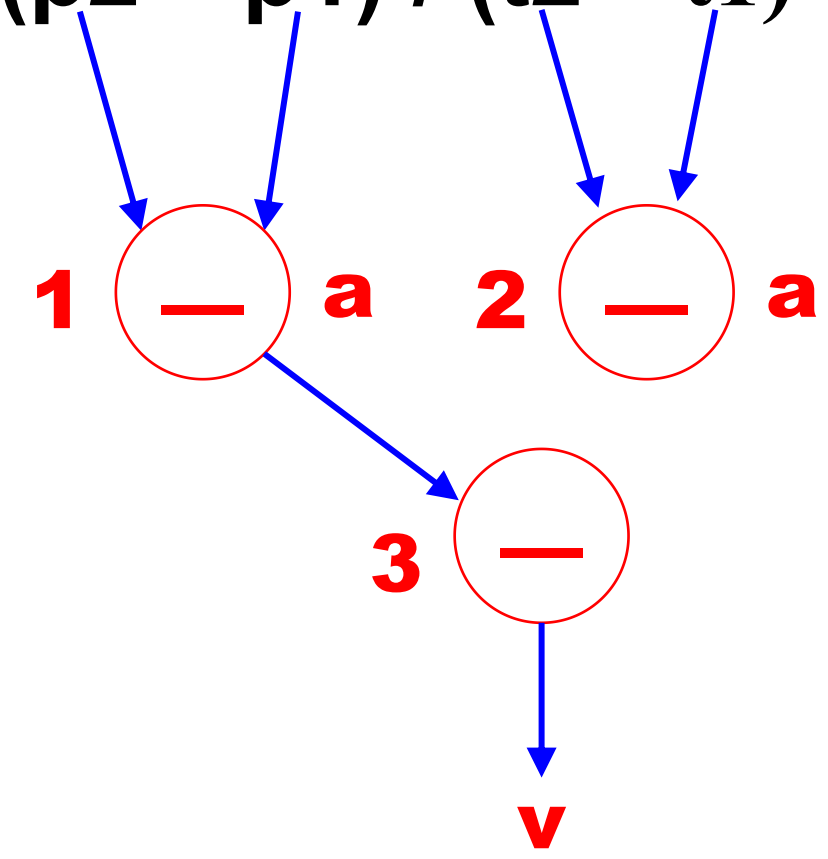
3.14159 2.0 2.0

6.28318

12.56636

Figure Evaluation Tree and Evaluation for v = (p2 - p1) / (t2 - t1);

v = (p2 – p1) / (t2 – t1)



p1	p2	t1	t2
4.5	9.0	0.0	60.0

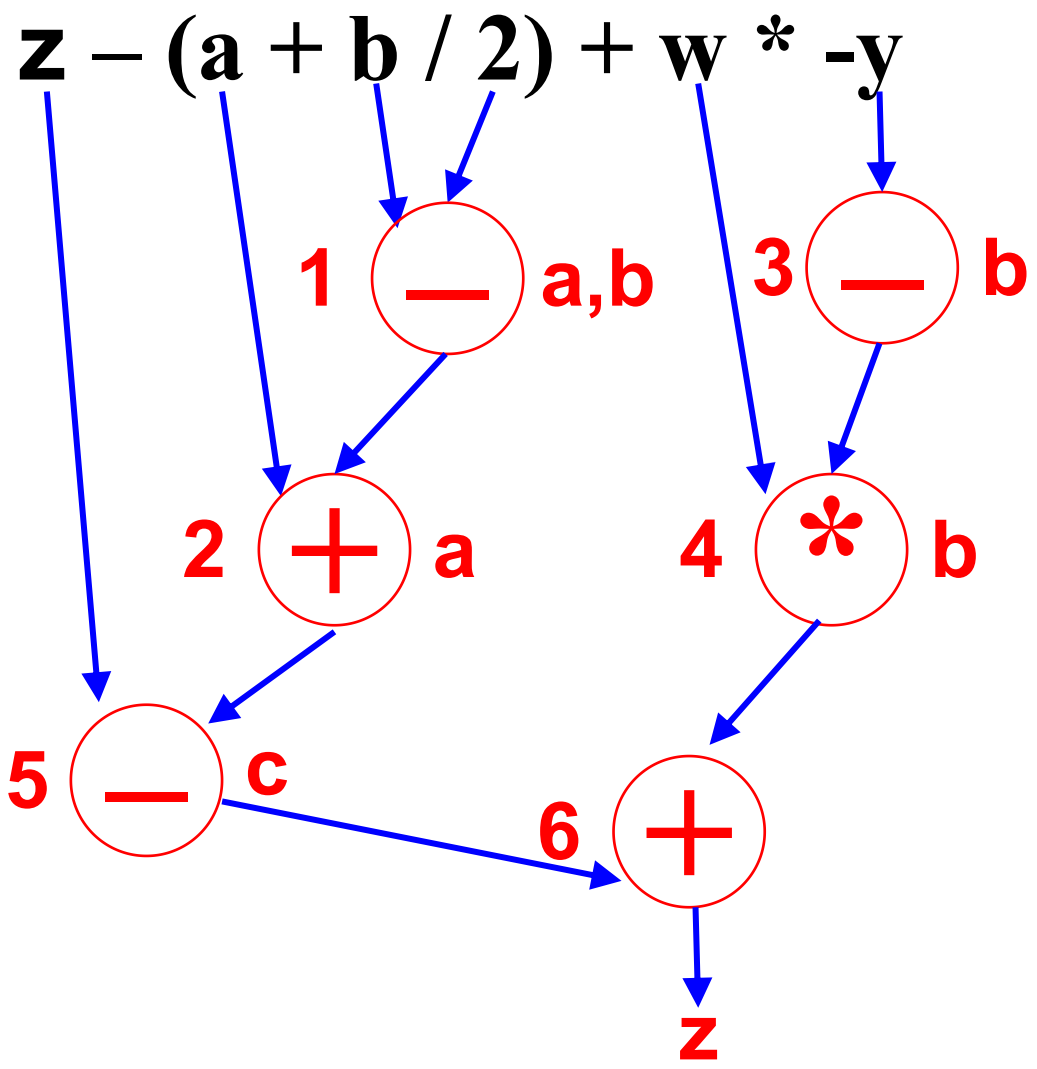
v = (p2 – p1) / (t2 – t1)

9.0 4.5 60.0 0.0

4.5 60.0

0.075

Figure Evaluation Tree and Evaluation for z - (a + b / 2) + w * -y



z	a	b	w	y
8	3	9	2	-5

z – (a + b / 2) + w * -y

8 3 9 2 -5

4 5

7 10

1

11

Common Programming Errors

- **debugging**
  - removing errors from a program
- **syntax error**
  - a violation of the C grammar rules
  - detected during program translation (compilation)
- **run-time error**
  - an attempt to perform an invalid operation
  - detected during program execution
- **logic error**
  - an error caused by following an incorrect algorithm

Figure A Program with a Run-Time Error

```
#include <stdio.h>

int
main (void>
{
        int    first, second;
        doubt temp, ans;

        printf("Enter two integers> ");
        scanf("%d%d", &first, &second);
        temp = second / first;
        ans = first / temp;
        printf("The result is %.3f\n", ans);

        return (0);
}

Enter two integers> 14 3
Arithmetic fault, divide by zero at line 272 of routline main
```

Figure A Program That Produces Incorrect Results Due to & Omission

```
#include <stdio.h>

int main (void>
{
        int    first, second; sum;

        printf("Enter two integers> ");
        scanf("%d%d", first, second); /* ERROR | | should be &first, &second */
        sum = first + second;
        printf("%d + %d = %d\n", first, second, sum);

        return (0);
}

Enter two integers> 14 3
5971289 + 5971297 = 11942586
```

Wrap Up

- Every C program has preprocessor directives and a main function.
- The main function contains variable declarations and executable statements.
- C’s data types enable the compiler to determine how to store a value in memory and what operations can be performed on that value.

THE END

Fangtian Zhong
CSCI 112

Gianforte School of Computing
Norm Asbjornson College of Engineering
E-mail: fangtian.zhong@montana.edu

2024.08.30