Constants Lesson Outline

What is a Constant?

In mathematics, a ***constant*** is a value that cannot change.

In programming, a ***constant*** is like a variable, except that its value cannot change.

The Difference Between a Variable and a Constant

The **difference** between a variable and a constant:

- a **variable**'s value can **vary**, but
- a **constant**'s value is **constant**.

Categories of Constants: Literal & Named

There are two categories of constants:

- ***literal constants***, whose values are expressed **literally**;
- ***named constants***, which have **names**.

Literal Constants

A ***literal constant*** is a constant whose value is expressed **literally**:

- `int` literal constants
  EXAMPLES: `5`, `0`, `-127`, `+403298`, `-385092809`
- `float` literal constants
  EXAMPLES: `5.2`, `0.0`, `-127.5`, `+403298.2348`, `-3.85092809e+08`
- `char` literal constants
  EXAMPLES: `'A', '7', '?'`
- character string literal constants
  EXAMPLES: `"A"`, `"Henry"`, `"What's it to ya?"`

Literal Constant Example Program

```
% cat tax1997_literal.c
#include <stdio.h>
int main ()
{ /* main */
    float income, tax;

    printf("I'm going to calculate the federal income\n");
    printf("  tax on your 1997 income.\n");
    printf("What was your 1997 income in dollars?\n");
    scanf("%f", &income);
    tax = (income - (4150.0 + 2650.0)) * 0.15;
    printf("The 1997 federal income tax on $%2.2f\n", income);
    printf("  was $%2.2f.\n", tax);
} /* main */
% gcc -o tax1997_literal tax1997_literal.c
% tax1997_literal
I'm going to calculate the federal income
  tax on your 1997 income.
What was your 1997 income in dollars?
20000
The 1997 federal income tax on $20000.00
  was $1980.00.
```

Named Constants

A ***named constant*** is a constant that has a name.

A named constant is **exactly like a variable, except** that its value is set at compile time (by initializing it) and **CANNOT** change at runtime.

A named constant is **exactly like a literal constant, except** that it **HAS A NAME**.

In a named constant declaration, we indicate that it's a constant via the `const` ***attribute***, and we **MUST** initialize it:

```
const float pi = 3.1415926;
```

Name Constant Example Program

```
% cat circlecalc.c
#include <stdio.h>

int main ()
{ /* main */
    const float pi              = 3.1415926;
    const float diameter_factor = 2.0;
    float radius, circumference, area;

    printf("I'm going to calculate a circle's\n");
    printf(" circumference and area.\n");
    printf("What's the radius of the circle?\n");
    scanf("%f", &radius);
    circumference = pi * radius * diameter_factor;
    area = pi * radius * radius;
    printf("The circumference is %f\n", circumference);
    printf(" and the area is %f.\n", area);
} /* main */
% gcc -o circlecalc circlecalc.c
% circlecalc
I'm going to calculate a circle's
 circumference and area.
What's the radius of the circle?
5
The circumference is 31.415924
 and the area is 78.539810.
```

The Value of a Named Constant Can't Be Changed

```
% cat constassign.c
#include <stdio.h>

int main ()
{ /* main */
    const float pi = 3.1415926;

    pi = 3.0;
} /* main */
% gcc -o constassign constassign.c
constassign.c: In function 'main':
constassign.c:7: error: assignment of read-only variable 'pi'
```

Why Literal Constants Are BAD BAD BAD

When you embed numeric literal constants in the body of your program, you make it **much harder** to ***maintain*** and ***upgrade*** your program.

1997 Tax Program with Numeric Literal Constants

```
% cat tax1997_literal.c
#include <stdio.h>

int main ()
{ /* main */
    float income, tax;

    printf("I'm going to calculate the federal income\n");
    printf("  tax on your 1997 income.\n");
    printf("What was your 1997 income in dollars?\n");
    scanf("%f", &income);
    tax = (income - (4150.0 + 2650.0)) * 0.15;
    printf("The 1997 federal income tax on $%2.2f\n", income);
    printf("  was $%2.2f.\n", tax);
} /* main */
% gcc -o tax1997_literal tax1997_literal.c
% tax1997_literal
I'm going to calculate the federal income
  tax on your 1997 income.
What was your 1997 income in dollars?
20000
The 1997 federal income tax on $20000.00
  was $1980.00.
```

1999 Tax Program with Numeric Literal Constants

```
% cat tax1999_literal.c
#include <stdio.h>

int main ()
{ /* main */
    float income, tax;

    printf("I'm going to calculate the federal income\n");
    printf("  tax on your 1999 income.\n");
    printf("What was your 1999 income in dollars?\n");
    scanf("%f", &income);
    tax = (income - (4300.0 + 2750.0)) * 0.15;
    printf("The 1999 federal income tax on $%2.2f\n", income);
    printf("  was $%2.2f.\n", tax);
} /* main */
% gcc -o tax1999_literal tax1999_literal.c
% tax1999_literal
I'm going to calculate the federal income
  tax on your 1999 income.
What was your 1999 income in dollars?
20000
The 1999 federal income tax on $20000.00
  was $1942.50.
```

Why Named Constants Are Good

When you use named constants in the body of your program instead of literal constants, you **isolate** the constant values in the declaration section, making them **trivial** to find and to change.

1997 Tax Program with Named Constants

```
% cat tax1997_named.c
#include <stdio.h>

int main ()
{ /* main */
    const float standard_deduction = 4150.0;
    const float single_exemption = 2650.0;
    const float tax_rate = 0.15;
    const int tax_year = 1997;
    float income, tax;

    printf("I'm going to calculate the federal income tax\n");
    printf("  on your %d income.\n", tax_year);
    printf("What was your %d income in dollars?\n", tax_year);
    scanf("%f", &income);
    tax = (income - (standard_deduction + single_exemption)) * tax_rate;
    printf("The %d federal income tax on $%2.2f\n", tax_year, income);
    printf("  was $%2.2f.\n", tax);
} /* main */
% gcc -o tax1997_named tax1997_named.c
% tax1997_named
I'm going to calculate the federal income tax
  on your 1997 income.
What was your 1997 income in dollars?
20000
The 1997 federal income tax on $20000.00
  was $1980.00.
```

1999 Tax Program with Named Constants

```
% cat tax1999_named.c
#include <stdio.h>

int main ()
{ /* main */
    const float standard_deduction = 4300.0;
    const float single_exemption = 2750.0;
    const float tax_rate = 0.15;
    const int tax_year = 1999;
    float income, tax;

    printf("I'm going to calculate the federal income tax\n");
    printf("  on your %d income.\n", tax_year);
    printf("What was your %d income in dollars?\n", tax_year);
    scanf("%f", &income);
    tax = (income - (standard_deduction + single_exemption)) * tax_rate;
    printf("The %d federal income tax on $%2.2f\n", tax_year, income);
    printf("  was $%2.2f.\n", tax);
} /* main */
% gcc -o tax1999_named tax1999_named.c
% tax1999_named
I'm going to calculate the federal income tax
  on your 1999 income.
What was your 1999 income in dollars?
20000
The 1999 federal income tax on $20000.00
  was $1942.50.
```