

IS12 - Introduction to Programming

Lecture 8:

Data Types and Expressions in C

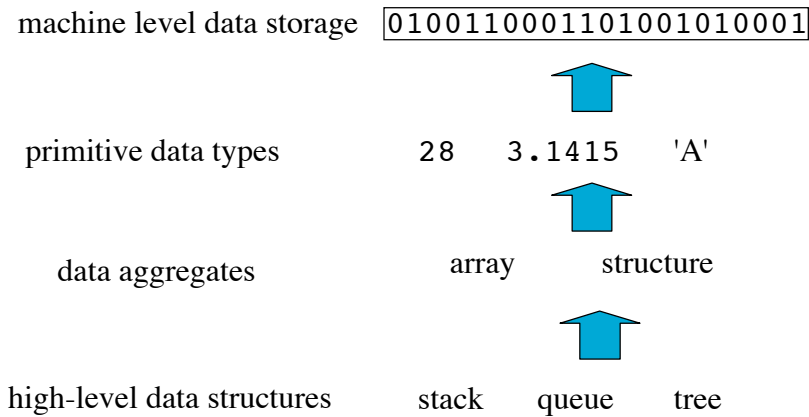
Peter Brusilovsky

<http://www2.sis.pitt.edu/~peterb/0012-072/>

Overview

- Primitive Data Types: integer, float, character
- printf() with formatting string
- Arithmetic operations
- Type conversion
- Order of calculation

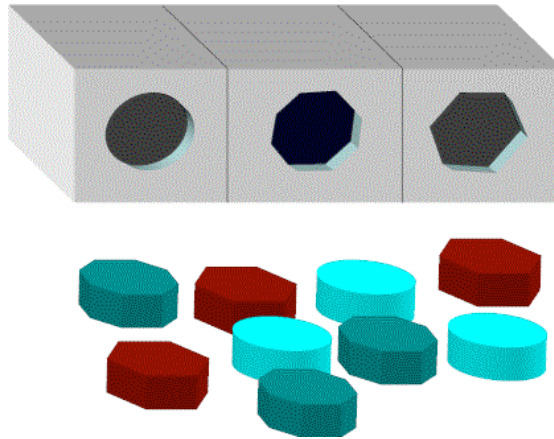
From Data to Data Structures



On each level...

- We do not want to be concerned with the way to represent objects of this level via objects of lower level - and we don't have to.
- We want to be concerned with the semantics of data on this level.
 - What is it?
 - What we can do with it?

Primitive Data Types



Primitive Data Types

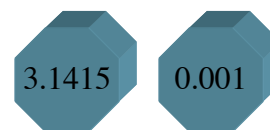
■ Integer data

1, 10, -999, 1000



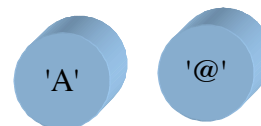
■ Floating point data

3.1415, -0.001, 2.0



■ Characters

'A', 'B', '_', '@'



Primitive data can be printed

```
/* Example: Printing primitive data with printf */  
#include <stdio.h>  
  
main()  
{  
    printf("Hello, World!\n");  
    printf("Here are integers: %d %d\n", 10, 99);  
    printf("Here are floats: %f %f\n", 3.1415, 0.001);  
    printf("Here are characters: %c %c\n", 'A', '@');  
}
```

How printf works

- Simple form - prints a string
`printf("Some String\n");`
– Note special symbols `\t` and `\n`
- Regular form
`printf("..%d..%f..%c..\n",
Expr1, Expr2, Expr3);`
– % specifications should match **expressions**



Expressions?

- What is an expression?
- Anything that can have a *value*
- In C almost anything is an expression
- Expressions can have values of different type
- A literal constant is an expression:
4 3.1415 99 'A' 0.0001



More control over printing

```
#include <stdio.h>

void main()
{
    printf("Hello, World!\n");
    printf("Integer: %5d\n", 10);
    printf("Float: %4.2f\n", 3.1415);
    printf("Characters: %c\n", 'A');
    printf("Mix: %08d %8.5f %c\n", 99, 0.0001, '@');
}
```



Summary of print control

- `%d` prints an integer value
- `%6d` prints an integer at least 6 char wide (leading blanks may be printed)
- `%f` prints a floating point value
- `%6f` prints a floating point at least 6 char wide
- `%.2f` prints a floating point value with exactly 2 digits after decimal point
- `%6.2f` prints a floating point at least 6 char wide, including 2 digits after decimal point



Arithmetic operations

- **Integer:** `+` `-` `*` `/` `%`
- **Float:** `+` `-` `*` `/`
- **Example:**

```
#include <stdio.h>
main() {
    printf("Let's calculate!\n");
    printf("1234 + 4321 = %5d\n", 1234 + 4321);
    printf("2 * 3.1415 = %7.1f\n", 2 * 3.1415);
    printf("5/2 = %d\n", 5 / 2); /* int division */
    printf("5.0 / 2 = %f\n", 5.0 / 2);
                                /* type conversion */
    printf("5 %% 2 = %d\n", 5 % 2); /* remainder */
}
```



Type conversion

- Automatic type conversion
 - If operands are of the same type, the result will be of this type too
 - If operands are of different types, they will be converted to the “broader” type (i.e., int to float) before the calculation
- Later we will learn about *assignment conversion* and *casting*



Multi-step calculations

```
/* Temperature Converter */  
  
#include <stdio.h>  
  
void main()  
{  
    printf("100 Fahrenheit = %.2f Celsius\n",  
        (5.0 / 9.0 ) * (100 - 32));  
}
```

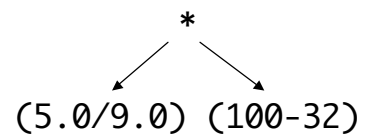


Example with brackets

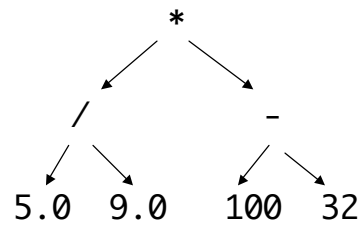
$$(5.0/9.0)*(100-32)$$



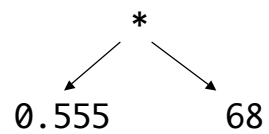
Example with brackets



Example with brackets



Example with brackets





Example with brackets

37.77



Order of calculation

- Operations have priorities (* and / have higher priority than + and -)
- Within operators of the same priority the order is defined by their *associativity*
- Use brackets when you need to change the default order of calculations or when you are not sure
- Learn precedence/associativity table! (K&R2, p.53; D&D, p. 44; Perry, insert)

Precedence of operators

■ * / % → third priority

■ + - → fourth priority

■ $3 + 12 * 6 \equiv 3 + (12 * 6)$

■ $1.2 * 2 + 3 \equiv (1.2 * 2) + 3$

■ $3 - 99 \% 5 \equiv 3 - (99 \% 5)$

Associativity of operators

■ * / % → left to right

■ + - → left to right

■ $3 * 12 * 6 \equiv (3 * 12) * 6$

■ $1.2 * 2 / 3 \equiv (1.2 * 2) / 3$

■ $3 / 9 \% 5 * 2 \equiv ((3 / 9) \% 5) * 2$