COMP26020 Programming Languages and Paradigms -- Part 1

Variables, Types, Printing to the Console

Variables

Variables

```c
#include <stdio.h>

int main() {
    int a;                          // declare local variable a of type int (signed integer)

    a = 12;                         // set the value of a to 12
    printf("a's value: %d\n", a);   // print the value of a

    return 0;
}
```

04-c-variables-types-printf/variables-1.c

- Variable: **name**, **type**, **value**
- Must be **declared** before being used
  - Declare with `<variable type> <variable name>;`

Variables

```c
#include <stdio.h>

int main() {
    int a;                          // declare local variable a of type int (signed integer)

    a = 12;                         // set the value of a to 12
    printf("a's value: %d\n", a);   // print the value of a

    return 0;
}
```

04-c-variables-types-printf/variables-1.c

- Variable: **name**, **type**, **value**
- Must be **declared** before being used
  - Declare with `<variable type> <variable name>;`

```c
int _x;   // variable names should start with a letter or underscore
int sum;
int final_balance2; // it can contain a combination of letters/numbers/underscores
int a_long_variable_name_is_also_possible_but_not_very_practical;
```

Variables

```c
#include <stdio.h>

int main() {
    int a;                          // declare local variable a of type int (signed integer)

    a = 12;                         // set the value of a to 12
    printf("a's value: %d\n", a);   // print the value of a

    return 0;
}
```

04-c-variables-types-printf/variables-1.c

- Variable: **name**, **type**, **value**
- Must be **declared** before being used
  - Declare with `<variable type> <variable name>;`

```c
int a; int b; int c;
int d = 12;  // declare and set
int x, y = 10, z = 11; // declare x, y and z,
                       // set y and z

a = 12;      // set a to 12
```

```c
b = 20;       // set b to 20
c = 10 + 10; // set c to 20
a = b;        // a = 20
d++;          // d = d + 1
y *= 2        // y = y * 2;
//
```

04-c-variables-types-printf/variables-2.c

Types

Types

Types are used by the compiler for 2 things:

- The type define the **amount of space in memory** allocated for a variable by the compiler
- At build time, the compiler **checks the validity of operations** on the variable based on its type

Types

Types are used by the compiler for 2 things:

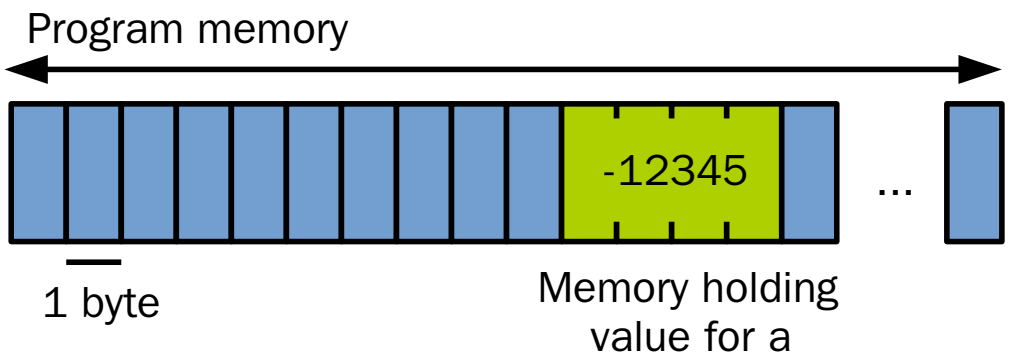
- The type define the **amount of space in memory** allocated for a variable by the compiler
- At build time, the compiler **checks the validity of operations** on the variable based on its type

3 basic types categories: integers, floating-point numbers, characters

Types

```
int a;          // signed integer takes 4 bytes in memory on the Intel x86-64 architecture
a = -12345;
```

At runtime:



Types

Characters and floats:

```
char my_char_variable;
float my_float_variable;

my_char_variable = 'x';   // use single quotes for characters
my_float_variable = 12.4;
```

Types

Arithmetics mixing integers and floats:

```c
int int1 = 2, int2 = 4;
float float1 = 2.8;

/* when mixed with floats in arithmetics, integers are promoted to floats */
float float_res = int1 + float1;              // float_res:  4.8
float float_res2 = int1 * (float1 + int2);    // float_res2: 13.6

/* when stored in an integer variable, floats are _truncated_ */
int int_res = int1 * (float1 + int2);         // int_res: 13

printf("float result:   %f\n", float_res);
printf("float result2:   %f\n", float_res2);
printf("integer result: %d\n", int_res);
```

04-c-variables-types-printf/types.c

More Types, Qualifiers

- Types: `char`, `int`, `float` (single precision), `double` (double precision)
- Integer qualifiers: `signed` (positive and negative integers), `unsigned` (only positive integers), `short` (smaller), `long` (larger)
- Determines the **storage size in memory**

```
short int a;               // signed, at least 16 bits: [-32,767,          +32,767]
int b;                     // signed, at least 16 bits: [-32,767,          +32,767]
unsigned int c;            // unsigned:                 [0,                +65,535]
long int d;                // at least 32 bits:         [-2,147,483,647,   +2,147,483,647]
unsigned long int e;       // unsigned:                 [0,                +4,294,967,295]
long long int f;           // at least 64 bits:         [-9x10^18,         +9x10^18]
long long unsigned int g;  // unsigned:                 [0,                +18x10^18]
float h;                   // storage size unspecified, generally 32 bits
double i;                  // storage size unspecified, generally 64 bits
```

- Exact implementation is **architecture dependent**
- Complete list here: https://en.wikipedia.org/wiki/C_data_types

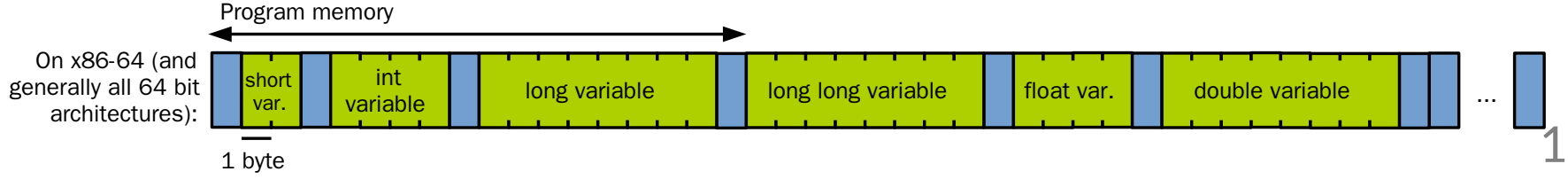
Types

- The function `sizeof` can be used to get the exact size of a type on a given machine

```c
int so_short = sizeof(short int);
int so_int = sizeof(int);
int so_uint = sizeof(unsigned int);
int so_long = sizeof(long int);
int so_longlong = sizeof(long long int);
int so_float = sizeof(float);
int so_double = sizeof(double);

printf("size of short:         %d bytes\n", so_short);
printf("size of int:           %d bytes\n", so_int);
printf("size of unsigned int:  %d bytes\n", so_uint);
printf("size of long int:      %d bytes\n", so_long);
printf("size of long long int: %d bytes\n", so_longlong);
printf("size of float:         %d bytes\n", so_float);
printf("size of double:        %d bytes\n", so_double);
```

04-c-variables-types-printf/sizeof.c



Printing to the Console: printf

Printing to the Console

- `printf` accepts **one or more arguments**:
  - Format string (mandatory)
  - Optionally a list of variables which values should be printed, replacing markers (e.g. `%d` in the format string)

```c
int int_var = -1;
unsigned int uint_var = 12;
long int lint_var = 10;
float float_var = 2.5;
double double_var = 2.5;
char char_var = 'a';
char string_var[] = "hello";



//
```

```c
printf("Integer: %d\n", int_var);
printf("Unsigned integer: %u\n", uint_var);
printf("Long integer: %ld\n", lint_var);
printf("Float: %f\n", float_var);
printf("Double: %lf\n", double_var);
printf("Characters: %c\n", char_var);
printf("String: %s\n", string_var);

printf("Several variables: %d, %lf, %s\n",
    int_var, double_var, string_var);
//                  04-c-variables-types-printf/printf.c
```

- Format string accepts escaped characters such as `\n` for line break
- See here for a list of all markers/modifiers

Printing to the Console (2)

- Make sure to use the right marker!
  - In most situations the compiler won't tell you if you make a mistake and incorrect values will be displayed at runtime

```c
int my_variable = -42;
printf("-42 printed with %%d: %d\n", my_variable);
printf("-42 printed with %%u: %u\n", my_variable);  // -42 interpreted as an unsigned integer

unsigned long long int ull = 5467513055454315;
printf("5467513055454315 printed with %%llu: %llu\n", ull);
printf("5467513055454315 printed with %%u: %u\n", ull);// interprets only 4 bytes vs. 8 for llu
printf("5467513055454315 printed with %%d: %d\n", ull);// first 4 bytes, as unsigned
//
```

04-c-variables-types-printf/printf-bugs.c

Summary

- Variables
- Types
- `printf`

Feedback form: https://bit.ly/2VslNaB