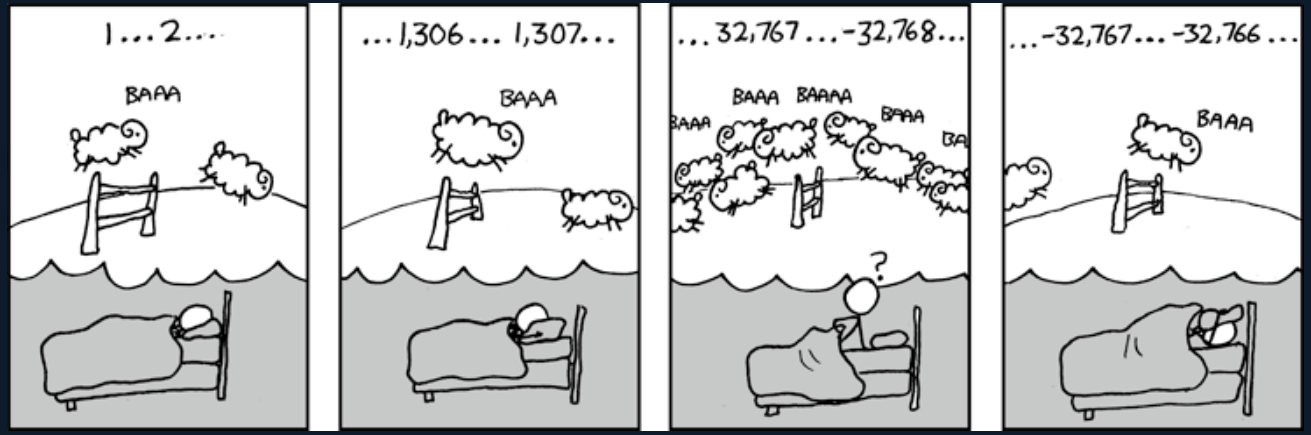
Variables



Canberra Institute of Technology

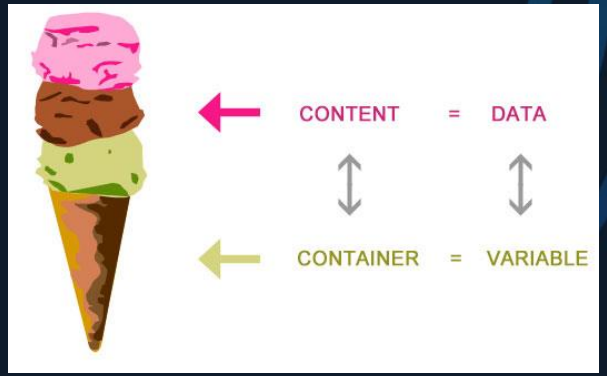
aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Contents

- Data Types
- Sizes
- Mathematical operators
- Naming variables
- Input & Output
- Variable scope

What are Variables?

Variables are storage space in memory (RAM) for the data we want to use in our programs.



- Variables form the building blocks of any program
- Variables have a name, (an alias for a physical memory location).
- Variables can be read from or written to.
- They are called variables because they represent information that can change

Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Primitive Variables

- C++ provides us with a set of variable types for us to use.
  - These are known as primitive data types.
    - They come in two flavours.
      - Integral (whole numbers)
      - Floating Point (fractional numbers)

Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Integral types

- Store a finite range
  - Exceeding this range will cause the value to overflow and wrap around.

- Have two flavours
  - Signed (they can be negative)
  - Unsigned (no negative values)

Integral Types

- Non-Numerical integral types
  - bool (true or false)
  - char – represents a character in the ascii character table
- Numerical Types
  - short (2 bytes)
  - int (4 bytes on x86, 8 bytes on x64)
  - char (1 byte) doesn't always need to represent a character

Declaring variables

- To create a variable in C++ we need to first write the data type followed by the name we want to give our variable.

```
char someLetter;
int aNumber;
bool trueFalse;
```

- We can declare multiple variables on the same line if they are the same type:

```
char a, b, c;
int aNumber, anotherNumber;
```

Setting variables

- We can set variables as they are created (initialisation):

```
int aNumber = 7;
int anotherNumber = 10;
char letter = 'A';

aNumber = -4;
```

- You should always provide default values to all variables when they are defined

Floating Point Types

- Floating point variables contain a fractional component.

float	A single precision variable, has an accuracy to 7 decimal places
double	A double precision variable, has accuracy to 15 decimal places

- In memory, floating point numbers are processed in a much more complex manner than whole numbers

Declaring and Using Floating Point Numbers

```
float floatValue = 13.245f;
float i = 13.245; // double is casted to float
double SomeDouble = 13.245;
```

What is that f?

- The f informs the compiler that the number is a float, otherwise it assumes it is a double (like in the third line)

Naming your Variables

Use intuitive variable names!

- Good, readable code is self documenting and easy to read
  - This comes from naming variables appropriately.

- Good variable names should be descriptive (nouns)
  - score, health, ammo, damage

- Bad variable names should be avoided.
  - Pants, awesome, cake, something, thingy

Variable Naming Conventions

- No spaces or punctuation allowed.
- Variable names can’t start with a number.

- To make variables with multiple words in them, use a naming convention called Camel Case.
  - The letter of each word is capitalised, *except for the first*.
    playerHealth, ammoRemaining

Reserved words

- Some words cannot be variable names because they are key words in C++:

```
and       auto      break     case      catch
class     continue  default   delete    else
explicit  export    extern    false     for
friend    mutable   namespace new       not
operator  private   protected public    register
return    signed    static    switch    template
this      throw     true      try       typename
union     unsigned  using     virtual   volatile
```

Mathematical operations

- We can apply operations to our variables just like we can in maths.

- The mathematical operators available to us are:
  - Addition (+)
  - Subtraction (-)
  - Multiplication (*)
  - Division (/)
  - Modulus(%)
  - Increment (++)
  - Decrement (--).

```
int x = 1 + 2;
int y = 5 - x;
int z = x * y;
```

Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Modulus

The modulus operator divides one number by the other and returns the remainder.

For example:

- 5 % 2 will return 1, since 5 divided by 2 is 2 with 1 remainder.

Increment and decrement

These operators will add one or subtract one from a variable.

```
int num1 = 5;
num1++;
std::cout << num1;
```

The placement of the ++ (or --) is important. Placing it after the variable is called post-increment. This means the variable is incremented *after* it has been used within a statement.

```
int num1 = 5;
int num2 = num1++;
```

Placing the ++ (or --) before the variable is called pre-increment. This means the variable is incremented before it is used.

```
int num1 = 5;
int num2 = ++num1;
```

Operator precedence

This works the same way that it does in mathematics

- Multiplication and division take precedence over addition and subtraction

- And we can use parentheses (brackets) to force some operations to occur before others.

```
int x = (5 + 3) * 10;
```

Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Integer division

What will be the value of answer after the following line of code runs?

```
float answer = 1 / 2;
```

The result we get is 0.

An integer divided by another integer will give us an integer. So, the decimal part of the answer is cut off.

Type conversions

The problem on the previous slide can be solved by ensureing the numbers are floating point i.e. (1.0 or 2.0f)

So if we divide an integer by a float – what data type do we get?

Test the following:

```
float answer0 = 1.0f / 2;
float answer1 = 1 / 2.0f;
float answer2 = 1.0f / 2.0f;
```

We can force a variable to be a different type, but we will learn about this later.

Outputting variables

- We can output variables to the console:

```
int number = 5;
std::cout << number;
```

- We can output multiple variables by placing the << operator before each variable:

```
int number = 5;
char letter = 'A';
std::cout << number << letter;
```

Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Input

We can now collect input from the user by storing their data in variables.

To do this, use cin and the stream extraction operator >>

```
int number = 5;
std::cin >> number;
```

Input

We can also input multiple values with the same cin statement:

```
int num1, num2;
std::cin >> num1 >> num2;
```

The >> operator skips whitespace characters – spaces, tabs and newlines.

```
char letter1, letter2;
std::cin >> letter1 >> letter2;
```

What will be stored in letter1 and letter2 if the following data is input?

AB

Size of data types

- Different data types take up different amounts of memory

Name	Description	Size*	Range*
`char`	Character or small integer.	1byte	signed: -128 to 127 unsigned: 0 to 255
`short int (short)`	Short Integer.	2bytes	signed: -32768 to 32767 unsigned: 0 to 65535
`int`	Integer.	4bytes	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
`long int (long)`	Long integer.	4bytes	signed: -2147483648 to 2147483647 unsigned: 0 to 4294967295
`bool`	Boolean value. It can take one of two values: true or false.	1byte	true or false
`float`	Floating point number.	4bytes	+/- 3.4e +/- 38 (~7 digits)
`double`	Double precision floating point number.	8bytes	+/- 1.7e +/- 308 (~15 digits)
`long double`	Long double precision floating point number.	8bytes	+/- 1.7e +/- 308 (~15 digits)
`wchar_t`	Wide character.	2 *or* 4 bytes	1 wide character

Variable overflow and underflow

When a variable is assigned a value outside of it's range then the variable overflows or underflows.

When a variable overflows, its contents wrap around to that data type's lowest value.

```
short num = 32767;
std::cout << num << std::endl;
num = num + 1;
std::cout << num << std::endl;
```

The output for this will be:

32767
-32768

Variable scope

- Once a variable is created it does not live forever
- At some point it ceases to exist and the memory it used is released.
- A variable comes into existence once it is declared and is destroyed by the next closing curly brace }

```
int main()
{
	int number = 5;
	{
		int x = 10;
	}
	number = x;
	return 0;
}
```

Global variables

- Global variables are variables that are defined outside of any curly braces in your file.
- These variables exist from the point of definition until the program closes.
- Global variables are automatically initialised to zero.
- However you should avoid the use of global variables!
  They when programs scail up, the become difficult to track and maintain. Global variables lead to poor program structure.

Summary

- Variables are the building blocks of programs – we will expand upon the topics we learned today over the next few weeks.

- The skills you learned today are things that will be used every day throughout your programming career, so it’s a good idea to understand them solidly now!

Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

References

- Gaddis, T, 2012, *Starting Out with C++: From Control Structures Through Objects*, 7th edition, Pearson Education

- Dhaval, 2013, *Fundamental data types*, Programming Language
  - *http://c-programing4u.blogspot.com.au/2013/04/fundamental-data-types.html*

Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX