Basic I/O using `cin` and `cout`

`cout`, `cin`, `cerr`, and `clog` are objects representing streams

`cout`

- standard output stream
- console

`cin`

- standard input stream
- keyboard

`<<`

- Insertion operator
- output streams

`>>`

- extraction operator
- input streams

`cout` and `<<`

- Insert the data into the cout stream

```
cout << data;
```

- Can be chained

```
cout << "data 1 is " << data1;
```

- Does not automatically add line breaks

```
cout << "data 1 is " << data1 << endl;
cout << "data 1 is " << data1 << "\n";
```

`cin` and >>

- Extract data from the cin stream based on data's type

```
cin >> data;
```

- Can be chained

```
cin >> data1 >> data2;
```

- Can fail if the entered data cannot be interpreted

  `data` could have an undetermined value

Section Overview

Variables and Constants

- Declaring variables
- C++ primitive types
  - integer
  - floating point
  - boolean
  - character
- `sizeof` operator

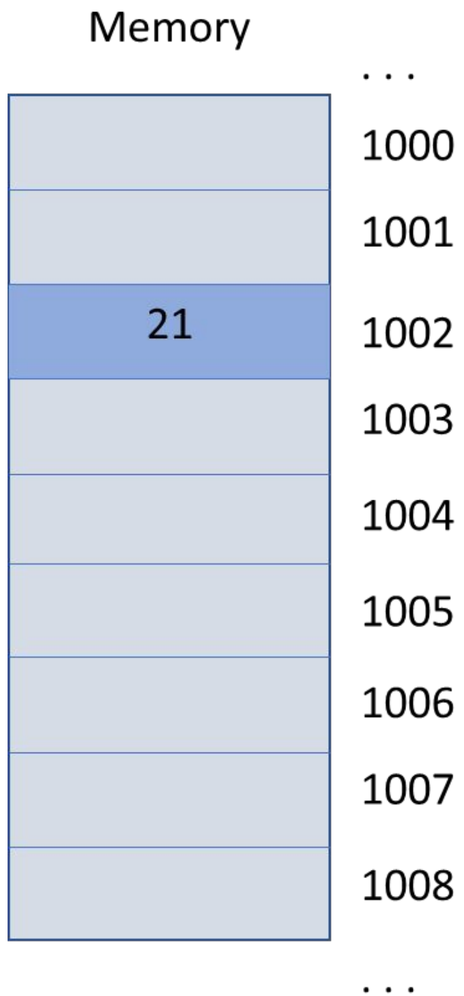
Section Overview

Variables and Constants

- What is a constant?
- Declaring constants
- Literal constants
- Constant expressions

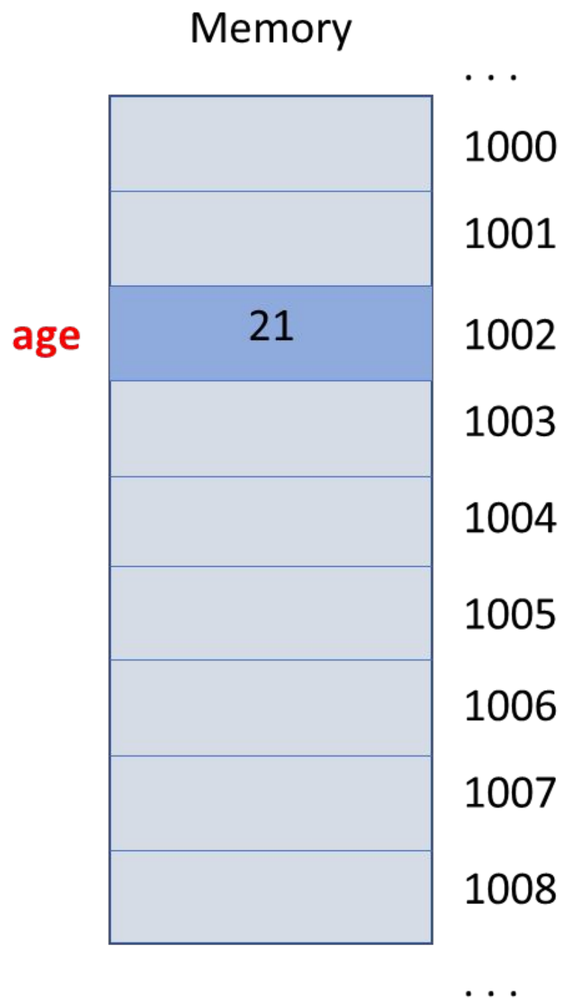
What is a variable?

move 21 to location 1002



What is a variable?

move 21 to age



What is a variable?

- A variable is an abstraction for a memory location
- Allow programmers to use meaningful names and not memory addresses
- Variables have
    - Type – their category (integer, real number, string, Person, Account…)
    - Value – the contents (10, 3.14. “Frank”…)
- Variables **must** be declared before they are used
- A variables value may change

```
age = 21;        // Compiler error
```

```
int age;

age = 21;
```

Declaring and Initializing Variables

Declaring Variables

```
VariableType VariableName;
```

```
int age;
double rate;
string name;

Account franks_account;
Person james;
```

Declaring and Initializing Variables

Naming Variables

- Can contain letters, numbers, and underscores

- Must begin with a letter or underscore (_)
  - cannot begin with a number

- Cannot use C++ reserved keywords

- Cannot redeclare a name in the same scope
  - remember that C++ is case sensitive

Declaring and Initializing Variables

Naming Variables

Legal	Illegal
Age	int
age	$age
_age	2014_age
My_age	My age
your_age_in_2014	Age+1
INT	cout
Int	return

Declaring and Initializing Variables

Naming Variables – Style and Best Practices

- Be consistent with your naming conventions
  - myVariableName vs. my_variable_name
  - avoid beginning names with underscores

- Use meaningful names
  - not too long and not too short

- Never use variables before initializing them

- Declare variables close to when you need them in your code

Declaring and Initializing Variables

Initializing Variables

```
int age;  // uninitialized
```

```
int age = 21; // C-like initialization


int age (21); // Constructor initialization


int age {21}; // C++11 list initialization syntax
```

C++ Primitive Data Types

- Fundamental data types implemented directly by the C++ language

- Character types
- Integer types
    - signed and unsigned
- Floating-point types
- Boolean type

- Size and precision is often compiler-dependent
    - #include <climits>

C++ Primitive Data Types

Type sizes

- Expressed in bits
- The more bits the more values that can be represented
- The more bits the more storage required

Size (in bits)	Representable Values	
8	256	2^8
16	65,536	2^{16}
32	4,294,967,296	2^{32}
64	18,446,744,073,709,551,615	2^{64}

C++ Primitive Data Types

Character Types

- Used to represent single characters, ‘A’, ‘X’, ‘@’
- Wider types are used to represent wide character sets

Type Name	Size / Precision
char	Exactly one byte. At least 8 bits.
char16_t	At least 16 bits.
char32_t	At least 32 bits.
wchar_t	Can represent the largest available character set.

C++ Primitive Data Types

Integer Types

- Used to represent whole numbers
- Signed and unsigned versions

C++ Primitive Data Types

Integer Types

Type Name	Size / Precision
signed **short** *int*	At least 16 bits.
signed **int**	At least 16 bits.
signed **long** *int*	At least 32 bits.
signed **long long** *int*	At least 64 bits

Type Name	Size / Precision
unsigned short *int*	At least 16 bits.
unsigned *int*	At least 16 bits.
unsigned long *int*	At least 32 bits.
unsigned long long *int*	At least 64 bits

C++ Primitive Data Types

Floating-point Type

- Used to represent non-integer numbers
- Represented by mantissa and exponent (scientific notation)
- Precision is the number of digits in the mantissa
- Precision and size are compiler dependent

Type Name	Size / Typical Precision	Typical Range
float	/ 7 decimal digits	1.2×10^{-38} to 3.4×10^{38}
double	No less than float / 15 decimal digits	2.2×10^{-308} to 1.8×10^{308}
long double	No less than double / 19 decimal digits	3.3×10^{-4932} to 1.2×10^{4932}

C++ Primitive Data Types

Boolean Type

- Used to represent true and false
- Zero is false.
- Non-zero is true.

Type Name	Size / Precision
bool	Usually 8 bits true or false (C++ keywords)

Using the `sizeof` operator

- The `sizeof` operator
  - determines the size in bytes of a type or variable

- Examples:

```
sizeof(int)
sizeof(double)
```

```
sizeof(some_variable)
```

```
sizeof some_variable
```

Using the `sizeof` operator

`<climits>` and `<cfloat>`

- The `climits` and `cfloat` include files contain size and precision information about your implementation of C++

```
INT_MAX
INT_MIN
LONG_MIN
LONG_MAX
FLT_MIN
FLT_MAX
. . .
```

What is a constant?

- Like C++ variables
    - Have names
    - Occupy storage
    - Are usually typed

However, their value cannot change once declared!

Types of constants in C++

- Literal constants
- Declared constants
    - `const` keyword
- Constant expressions
    - `constexpr` keyword
- Enumerated constants
    - `enum` keyword
- Defined constants
    - `#define`

Types of constants in C++

Literal constants

- The most obvious kind of constant

  x = **12**;

  y = **1.56**;

  name = “**Frank**”;

  middle_initial = ‘**J**’;

Types of constants in C++

Literal constants

- Integer Literal Constants

  12 - an integer

  12U - an unsigned integer

  12L - a long integer

  12LL - a long long integer

Types of constants in C++

Literal constants

- Floating-point Literal Constants

  12.1 - a double

  12.1F - a float

  12.1L - a long double

Types of constants in C++

Literal constants

- Character Literal Constants (escape codes)
  - \n - newline
  - \r - return
  - \t - tab
  - \b - backspace
  - \’ - single quote
  - \” - double quote
  - \\ - backslash

```
cout << “Hello\tthere\nmy friend\n”;
Hello   there
my friend
```

Types of constants in C++

Declared constants

- Constants declared using the `const` keyword

```
const double pi {3.1415926};

const int months_in_year {12};

pi = 2.5;  // Compiler error
```

Types of constants in C++

Defined constants

- Constants declared using the `const` keyword

```
#define pi 3.1415926
```

Don’t use defined constants in Modern C++