VARIABLES AND PRIMITIVE DATA TYPES

C++ BASICS

prepared by:

Gyro A. Madrona
Electronics Engineer

TOPIC OUTLINE

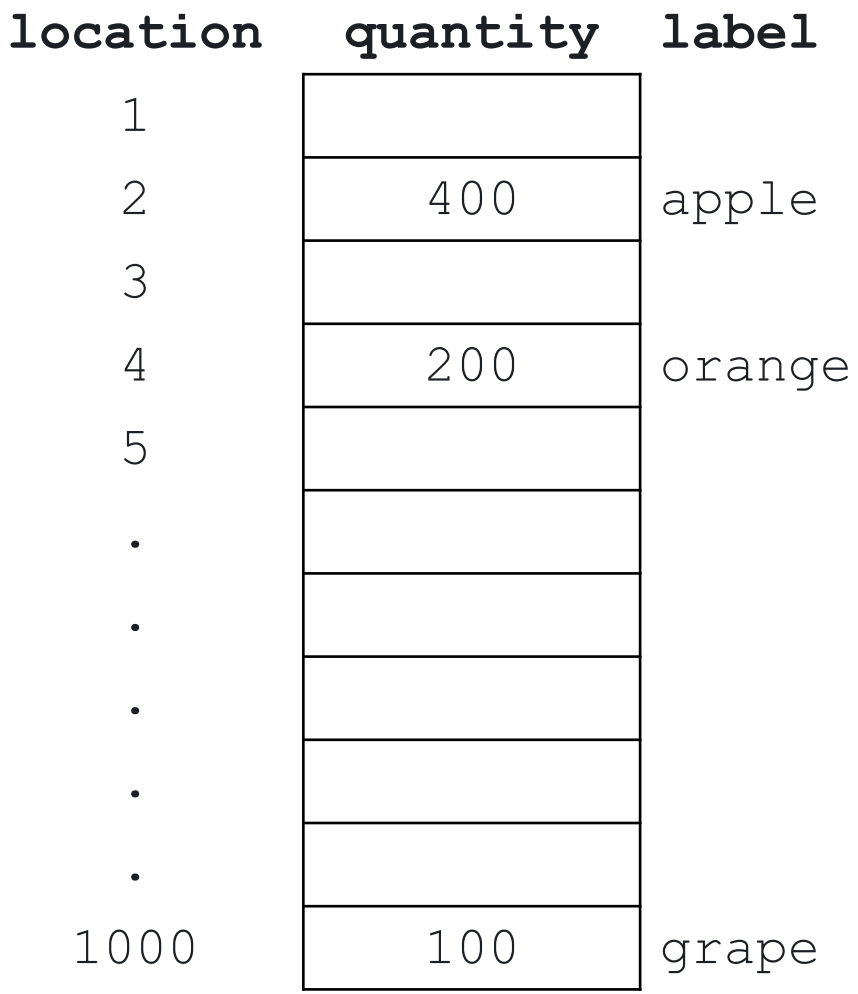
Variables

Primitive Data Types

VARIABLES

FRUIT CONTAINER ANALOGY

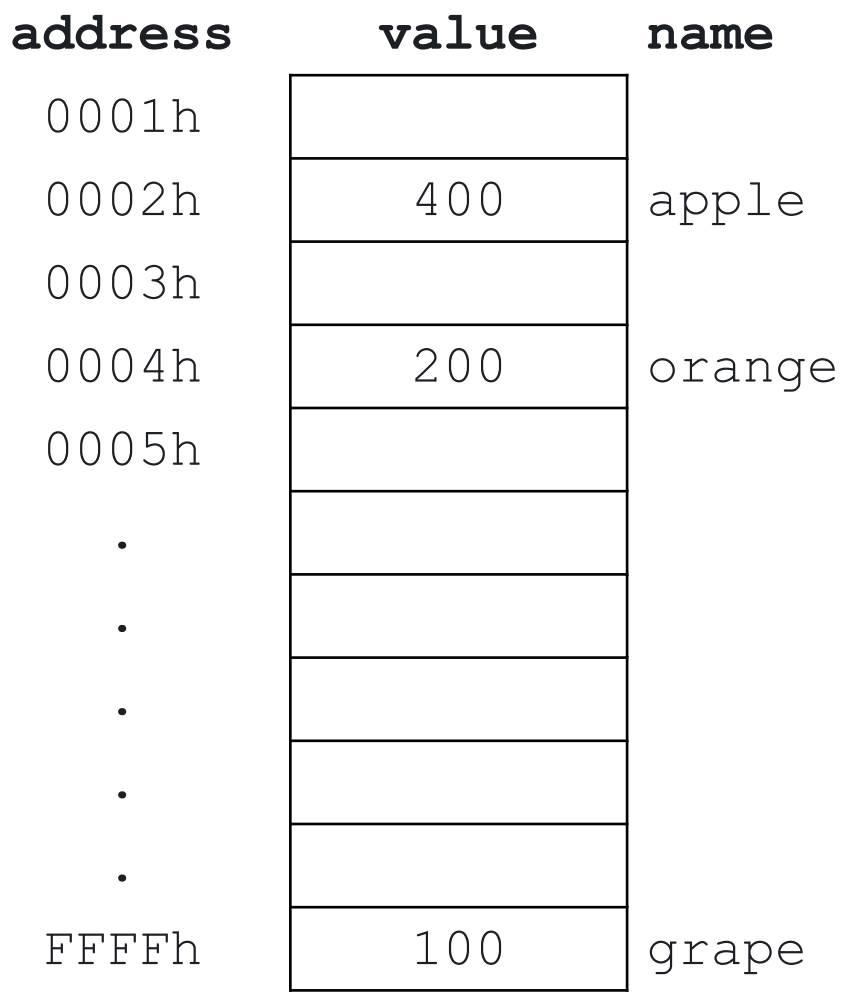
A **variable** is like a **storage container** that holds a specific type of fruit.



VARIABLE

A **variable** is a named storage location in memory that **holds a value** of a specific data type.

1. **Data type** – Defines what kind of data it can store (e.g., `int`, `double`, `char`).
2. **Name (identifier)** – a unique name assigned to the variable.
3. **Value** – The actual data stored in memory.



VARIABLE DECLARATION

Variable Declaration Syntax:

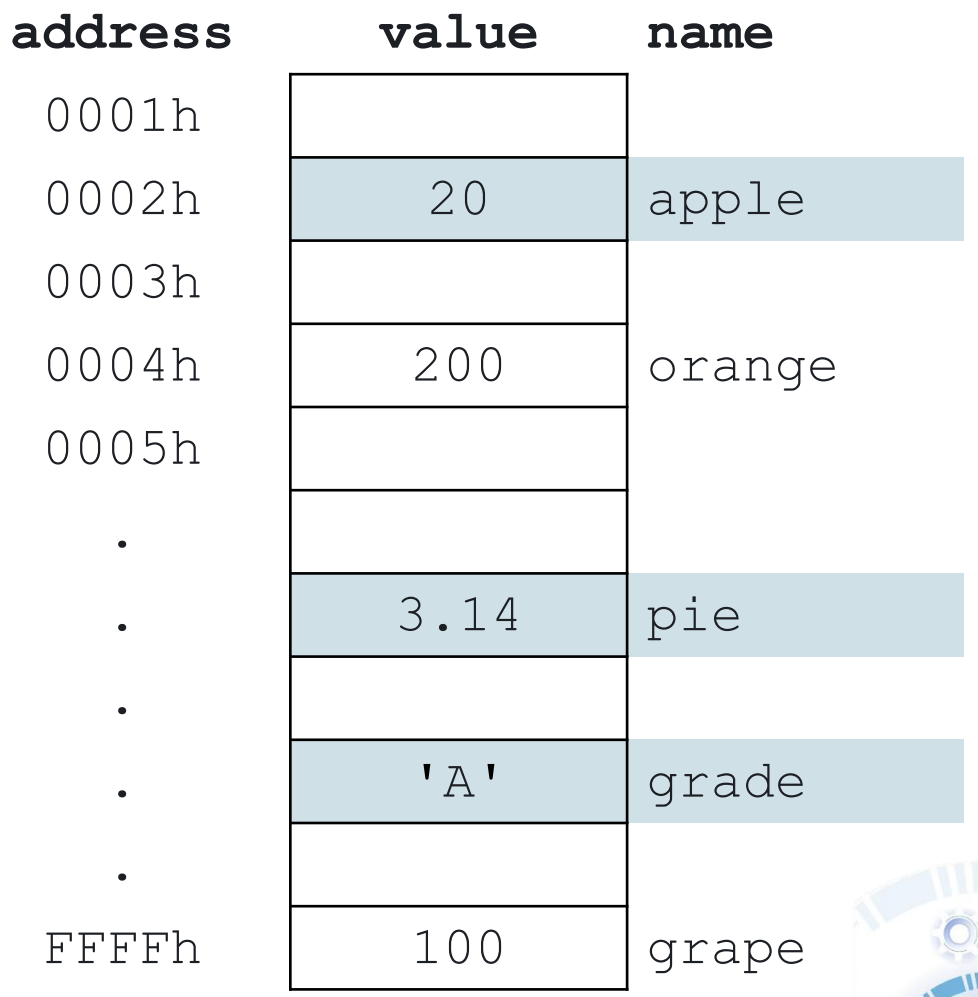
```
data_type name = value;
```

Example:

```
int apple = 20;

double pie = 3.14;

char grade = 'A';
```



IDENTIFIERS

A variable is identified by a **unique name**, called an **identifier**.

- It can contain letters, digits, and underscores.
- It cannot have "space".
- It cannot start with a digit.
- It cannot be a reserved keyword (**`int`**, **`return`**, **`class`**).
- It is case-sensitive (e.g., **`age`** and **`Age`** are different variables)

Valid Identifiers:

```
// contains only letters
int age;

// starts with an underscore
double _salary;

// contains letters and a digit
char grade1;

// uses an underscore instead of space
float total_price;
```

IDENTIFIERS

A variable is identified by a **unique name**, called an **identifier**.

- It can contain letters, digits, and underscores.
- It cannot have "space".
- It cannot start with a digit.
- It cannot be a reserved keyword (**`int`**, **`return`**, **`class`**).
- It is case-sensitive (e.g., **`age`** and **`Age`** are different variables)

Invalid Identifiers:

```
// starts with a digit
int 1stRank;

// uses a reserved keyword
char class;

// contains a space
float total price;
```

C++ KEYWORDS

asm	double	new	switch
auto	else	operator	template
break	enum	private	this
case	extern	protected	throw
catch	float	public	try
char	for	register	typedef
class	friend	return	union
const	goto	short	unsigned
continue	if	signed	virtual
default	inline	sizeof	void
delete	int	static	volatile
do	long	struct	while

Refer to C++ documentation for the complete list.

PRIMITIVE DATA TYPES

PRIMITIVE DATA TYPES

Primitive data types are **built-in** or predefined data types and can be used directly by the user to declare variables.

INTEGER

Example:

```
int age = 25;
int age = 1.25; // invalid
```

Integer **(int)** is used to store **integer values** (whole numbers).

Size: 4 bytes

CHAR

Character **(char)** is used to store a **single character** (ASCII value).

Size: 1 byte

Example:

```
char grade = 'A';

char grade = 'AB'; // invalid

char grade = '2';
```

BOOLEAN

Example:

```
bool is_active = true;

bool is_active = "true"; // invalid
```

Boolean **(bool)** is used to store Boolean values (**true** or **false**).

Size: 1 byte

FLOATING POINT

Floating point `(float)` is used to store **single-precision** floating-point numbers (decimal values).

Size: 4 bytes

Example:

```
float pi = 3.14f;

float pi = "3.14f"; // invalid

float pi = 3;
```

DOUBLE FLOATING POINT

Double floating point **(double)** is used to store **double-precision** floating-point numbers (decimal values with higher precision than **float**).

Size: 8 bytes

Example:

```
double s = 123.456;

double s = "123.456"; // invalid

double s = 123;
```

VOID

Valueless **(void)** represents the absence of a type. Commonly used as a return type for functions that **do not return a value**.

Size: no storage allocated

Example:

```
void message(){
    cout << "Hello, World!";
}
```

```
// invalid
void message(){
    cout << "Hello, World!";
    return 0;
}
```

WIDE CHARACTER

Example:

```
wchar_t unit = L'Ω';
```

Wide character **(wchar_t)** is used to store **wide characters** (typically for Unicode or larger character sets).

Size: 2 bytes (Windows) or 4 bytes (Linux)

EXERCISE

Determine the output of this code snippet:

```
int voltage = 2.5;

cout << voltage;
```

output:

```
2
```

Output Explanation:

Since voltage is an **`int`**, the decimal part (**`0.5`**) is truncated, and voltage will hold the value **`2`**.

EXERCISE

Determine the output of this code snippet:

```
double current_1 = 10.7;

int current_2 = current_1;

cout << current_2;
```

output:

```
10
```

Output Explanation:

The **`double`** value **`10.7`** is assigned to an **`int`** variable **`current_2`**. Hence, the decimal part **`(0.7)`** is truncated.

EXERCISE

Determine the output of this code snippet:

```
char letter = 'A';

int ascii_value = letter;

cout << ascii_value;
```

output:

```
65
```

Output Explanation:

The char value **'A'** is implicitly converted to its ASCII integer value (**65**) and assigned to the int variable **ascii_value**.

EXERCISE

Determine the output of this code snippet:

```
int number = 2147483647;

cout << number + 1;
```

output:

```
-2147483648
```

Output Explanation:

2,147,483,647 is the maximum value for a 32-bit signed integer. Adding **1** to **number** causes an integer overflow.

EXERCISE

Determine the output of this code snippet:

```
bool status = True;

cout << status;
```

output:

```
compilation error
```

Output Explanation:

The boolean literals are **`true`** and **`false`** (all lowercase), not True or False.

EXERCISE

Determine the output of this code snippet:

```
bool status = true;

int value = status;

cout << value;
```

output:

```
1
```

Output Explanation:

The **`bool`** value **`true`** is implicitly converted to an **`int`**. In C++, **`true`** is represented as **`1`** and **`false`** is represented as **`0`**.

LABORATORY