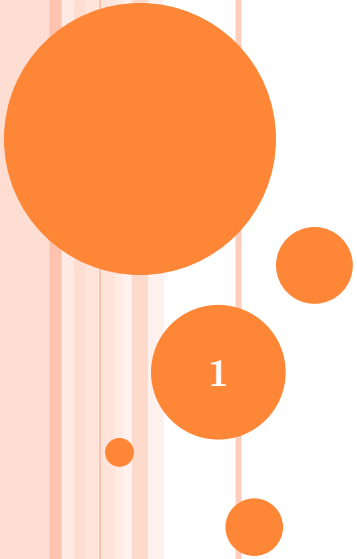

# VECTORS

## CS A150 - C++ Programming 1

# VECTORS

- A **vector** is a collection of data items of the **same type**.
- A list of items that "**grows and shrinks**".
- Formed from the **Standard Template Library** (**STL**).
  - Using a **template class** → can have different types (int, double, etc.).

# Vector Basics

- A vector has a ***base type***.
- Syntax:

```
vector < Base_Type >
```

  - Indicates a **template class**
  - Any type can be "plugged in" to ***Base_Type***
- Example declaration:

```
vector<int> v;
```

  - `v` is a vector of type `int`

# Declaration

- Each element can be **accessed *separately***.
- Here we define a **vector** of salaries

```
vector<double> salaries;
```

- Type of the **vector** is a **double**
- The variable **salaries** stores a ***sequence*** of **doubles**.

# Assigning Values

- The **first time** you **insert** values in a **vector** you need to use the function **push_back**

```
v.push_back(32);
```

- If you need to modify or output the value assigned, you can use the **[]** operator

```
salaries[4] = 35000;
```

- The **first** index is 0.
- Index 4 is the fifth index in the vector.

# Syntax

```
#include <vector>
...
vector<int> v;          //declaration

v.push_back(32);        //initialization
v.push_back(25);

cout << v[0] << endl;   //can use to read or change,
                        //    but NOT to initialize

cout << "Size: " << v.size() << endl;
```

# USING THE `size` FUNCTION

- The function **size** returns an **unsigned integer** that stores the size of the vector.

- **Efficiency tip:** If you need to call the function more than once use a **variable** to store the size.

```
unsigned int numOfElements = v.size();

for (unsigned int i = 0; i < numOfElements; ++i)
    cout << v[i] << " ";
```

# Using the `const` Modifier

- When passing **vectors** as **parameters** you should
  - Pass by ***reference*** (**&**)
    - Use the **`const`** modifier,
      if the function does ***not*** change the vector.

# Example

- File: vectors_1

# Deleting Data from the Vector

- Having a **vector v** of **integers** that contains the following items:

```
16, 25, 39, 48, 51, 64, 79
```

  - To remove 39 → index 2:

```
v.erase(v.begin() + 2);
```

  - The vector will ***shrink***:

```
16, 25, 48, 51, 64, 79
```

# Deleting Data from the Vector(cont.)

- **erase( )**
  - Removes a single element or a range of elements from the vector.
  - **Example 1:** Deleting element at index 3

```
v.erase(v.begin() + 3);
```

  - **Example 2:** Deleting elements from index 2 (**included**) to index 5 (**not** included)

```
v.erase(v.begin() + 2, v.begin() + 5);
```

# Other Useful Functions (cont.)

- **insert( )**
  - Inserts an element ***before*** the element at position

- **Example:** Given a vector that contains 1 2 3 4 5,
  insert 999 at index 3.

```
v.insert(v.begin( ) + 3, 999);
```

  - Will result in → 1 2 3 999 4 5

# Other Useful Functions (cont.)

- **clear( )**
  - Removes all elements from the vector and resets its size to 0
  - **Parameters:** none
  - **Return value:** none
- **empty( )**
  - Returns whether the vector is empty (size 0)
  - **Parameters:** none
  - **Return value:** **true** if the vector is empty; **false** otherwise

- Check cplusplus.com for more functions

# Example

- File: vectors_2

# Size, Capacity, and Max Size

- At any point in time a vector has a **capacity**, which corresponds to how much memory is allocated to contain elements.
- The **size** denotes the number of elements that have been inserted in the vector.
- The **max_size** is the number of elements that the vector can hold.

# Efficiency Issues

- **Vectors** grow ***automatically***; that is, by default their capacity is **doubled** as needed.
- If **efficiency** is an issue, you should ***explicitly*** increase the capacity of the vector.

```
//to set the capacity to at least 32 elements
v.reserve(32);


//to set the capacity  to at least 10 more
//elements than the number of elements
//currently in the vector
v.reserve(v.size() + 10);
```

# Resizing a Vector

- You can change the **size** of a vector by using the function **resize()**:

```
v.resize(24);
```

  - If the **initial size** of the vector is
    - ***greater*** than 24, then all but the first 24 elements are lost.
    - ***smaller*** than 24, then the ***additional*** elements will be **zeros** by default, OR
      - Can set an element to be inserted, for example 100:

```
v.resize(24,100);
```

**Note:** Only the ***new*** elements will be 100.

# Example

- File: vectors_3

# Vectors (end