Structures

Ch 10.1

Image Clem Onojeghuo on Pexels

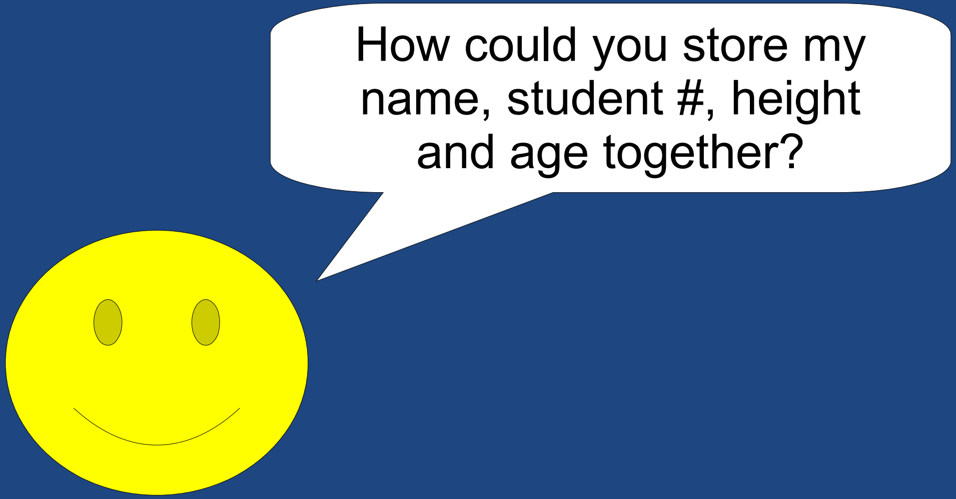
CMPT 130

Slides 15

© Dr. B. Fraser

Topics

1) How to manage complex data in a program.
2) How to use structures in a vector.


How could you store my name, student #, height and age together?

Motivation

- How could we store information about everyone in the class?

```
vector<string> names;
vector<int> studentNumbers;
vector<float> heights;
vector<float> gpa;
```

- These are called..
  Use same index in each vector for info on a single student.
- Hard to work with because to:
  - remove a student, we must change all four vectors.
  - sort the students, we must sort all vectors the same way.
  - pass a student to a function, we must pass four values.
  - add new information (eg. # credits taken) we must..

Solution: Structures

- Structure (“a struct”):
  a *complex* data type which..

..

- Example

```
struct student_t {
    string name;
    int studentNumber;
    float height;
    float gpa;
};
```

- Instantiate struct as local variable

```
student_t daStudent;
student_t newStudent = {"Steve A", 34500, 1.5, 3.2};
```

..

Allocates space on the stack as one block for the string, int, and two floats.

Accessing a struct

```
struct student_t {
        string name;
        int studentNumber;
        float height;
        float gpa;
};
```

- Use the dot operator (member operator) to access elements:

```
// Create a local variable for a student
student_t newStd;

// Fill in student's fields
newStd.name = readInName();
newStd.studentNumber = rand();
newStd.height = 1.75
newStd.gpa = 4.0;

// Display fields
cout << "name: "  << newStd.name
     << "ID: "    << newStd.studentNumber
     << "height: "<< newStd.height << endl;
```

Example

```
#include <iostream>
#include <iomanip>
using namespace std;


// Student structure
struct student_t {
    string name;
    int studentNumber;
    float height;
    float gpa;
};


void printStudent(student_t st)
{
    cout << fixed << setprecision(2);
    cout << setw(6) << st.studentNumber
         << setw(6) << st.height
         << setw(6) << st.gpa
         << " " << st.name
         << endl;
}
```

```
int main()
{
    student_t s1;

    cout << "Enter name: ";
    getline(cin, s1.name);

    cout << "Enter ID: ";
    cin >> s1.studentNumber;

    cout << "Enter GPA: ";
    cin >> s1.gpa;

    printStudent(s1);
}
```

Structures and Functions

- Can pass structures to functions
  - Pass by value: passes a..
    of the structure on the stack.
    void printStudent(student_t st);
    - expensive to copy large structs!
  - Pass by reference: function gets access to the original structure. Can use const reference too.
    void printStudent(const student_t &st);
- Can return a struct from a function:
  student_t readStudent();
  - It is..
    student_t newStd = readStudent();

Vector of Structs

- You can make a vector of structs:

```
vector<student_t> students;
```

- Access an element

```
students.at(3).name = "Billy";
cout << students.at(0).gpa << endl;
```

- Likely errors:

```
// What will this do:
students.name.at(3);

// Does this change the struct in the vector?
student_t temp = students.at(2);
temp.name = "Bobby";
```

Summary

- Structures are used to store related information.
  - Often used in vectors to store a program's data.
- Define with:

```
struct someStructName_t {
    int someValue1;
    float someValue2;
};
```

- Instantiate with:

```
someStructName_t myStructInstance;
vector<someStructName_t> myData;
```

- Use with:

```
cout << “val 1: “ << myStructInstance.someValue1 << endl;
cout << “vector: “ << myData.at(0).someValue1 << endl;
```