MY CODE DOESN'T WORK
LETS CHANGE NOTHING AND RUN IT AGAIN
quickmeme.com

Structs

CS2308
Gentry Atkinson

Primitive Datatypes

- C++ provides us with a list of data types built into the language:
  - Int, char, bool, float, double
  - string was added as a primitive in (I think) 2003
- Each primitive has some memory usage associated with it, and is interpreted some way in output.

User Defined Datatypes

- We are not limited to <u>only</u> using built-in datatypes in C++, we can make our own.
- Several keywords in C++ allow us to create new datatypes:
  - Typedef, union, enum, and struct
- A <u>struct</u> is a collection of primitive datatypes that can each be referenced individually.

Creating structs

- The struct keyword is used outside of any function definition.
- {Curly brackets} are used to enclose a list of named primitive variables.
- A semicolon; must be used to terminate the definition.
- Remember, no memory is allocated by a struct definition, instead a new datatype is created.

Example 1

```
struct Cat{
    string name;
    string breed;
    int age;
};

int main(int argc, char** argv){
    Cat c;
    return 0;
} //try to predict the output
```

Referencing struct Members

- We use the . dot operator to reference the individual members of a <u>struct</u>.
- structs can include arrays as members:
  - my_struct.a[0] = 1;
- We can also create arrays of structs:
  - all_cats[0].name = “Tibalt”;

Example 2

```
struct Student{
    string name;
    float grades[10];
};


int main(int argc, char** argv){
    Student roster[20];
    roster[0].name =  "Bruce";
    roster[0].grades[0] = 90.0;
    cout << "Student " << roster[0].name << " earned a "
        << roster[0].grades[0] << endl;
    return 0;
} //try to predict the output
```

Initialization List

•Just like an array, a struct can be initialized with a list of values in {curly brackets}.

•The values in the list must be in the same order as the struct members.

•If there are fewer values in the list than in the struct, the remaining members are set to 0;

Example 3

```
struct Foo{
    int a;
    int b;
    int c;
};

int main(int argc, char** argv){
    Foo a = {1, 2, 3};
    Foo b = {4, 5};
    cout << "struct a = " << a.a << a.b << a.c << endl;
    cout << "struct b = " << b.a << b.b << b.c << endl;
} //try to predict the output
```

Constructors

•structs can include a special function that runs whenever a variable of the struct's type is created, called a "constructor".

•The constructor must have the same name as the struct.

•One struct can have several over-loaded constructors.

Example 4

```
struct Rectangle{
    float length, width, area;
    Rectangle(float l, float w){
        length = l;
        width = w;
        area = length*width;
    }
};

int main(int argc, char** argv){
    Rectangle a(1,2);
    cout << "Rectangle a has area " << a.area << endl;
} //try to predict the output
```

Questions or Comments?