

Chapter 10 and 11

Well Not All of Them

Typedefs

- Typedef statements allow you to introduce a new name for an existing type.
- In my opinion, it is most useful for software engineering type reasons.
- It follows this pattern:
 - `typedef ExistingTypeName NewTypeName`
 - `typedef int Boolean;`
 - `Boolean dataOK;`

Enums

- An enumerated type is a user-defined data type whose domain is an ordered set of literal values expressed as identifiers.
- `enum Days { SUN, MON, TUE, WED, THU, FRI, SAT };`
- `enum Animals { RODENT, CAT, DOG, BIRD, REPTILE, HORSE, BOVINE };`
- `Animals inPatient;`
- `Animals outPatient;`

Allowed Operations

- `inPatient = Dog;`
- `inPatient = Animals(inPatient + 1);`
- `switch (inPatient)`
- You can also declare variables of this new type at the time of declaration.
- `enum Months { JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC } birthMonth, firstMonth, lastMonth;`

Structured Data Types

- A structured data type is one in which each value is a collection of components and whose organization is characterized by the method used to access individual components.
- page 548

Structs

- A record or a struct in C++, is a structured data type with a fixed number of components that are accessed by name. The components may be heterogeneous.
- A field or component is a piece of the record.

Example

```
struct TypeName
{
    DataType MemberName;
    DataType MemberName;
    ...
};
```

StudentRec

```
struct StudentRec
{
    string firstName;
    string lastName;
    float gpa;
    int programGrade;
};
```

Accessing a field

- To assign a value to one of the fields, you use the member selector operator ‘.’
- StudentRec Bill;
- Bill.firstName = “Fred”;
- Bill.lastName = “Williams”;

Aggregate Operations

- | | |
|--------------------------|----------------------------|
| ● Aggregate Operation | ● Allowed on Structs? |
| ● I/O | ● No |
| ● Assignment | ● Yes |
| ● Arithmetic | ● No |
| ● Comparison | ● No |
| ● Argument Passage | ● Yes, value and reference |
| ● Return from a function | ● Yes |

Assignment

- Just a quick word about assignment.
- The way it is copied is through member-wise assignment.
- Each field in the struct is simply matched up and copied.
- This becomes extremely important when you start dealing with “dynamic data”

Alternative Declaration

```
struct TypeName
{
    MemberList
} VariableList;
struct StudentRec
{
    ...
} Bill, Mary, Susie, Phil;
```

Unions

- Unions allow you to use the same memory space for several different types of data...
- not at the same time, but when you need to.

```
union WeightType  
{  
    long wtInTons;  
    int wtInPounds;  
    float wtInOunces;  
};
```

Data Abstraction

- Data abstraction is the separation of a data's logical properties from its implementation
- An **Abstract Data Type** is a data type whose properties are specified independently of any particular implementation.
- This leads us to Classes