



Chapter 11 – Structured Data

Abstract data types (ADTs) are data types created by the programmer. ADTs have their own range (or domain) of data and their own set of operations that may be performed on them.



Abstraction

- An abstraction is a general model of something.



Data Types

- C++ has several primitive data types:


Table 11-1

bool	int	unsigned long int
char	long int	float
unsigned char	unsigned short int	double
short int	unsigned int	long double



Abstract Data Types

- A data type created by the programmer
 - The programmer decides what values are acceptable for the data type
 - The programmer decides what operations may be performed on the data type



11.2 Focus on Software Engineering: Combining Data into Structures

- C++ allows you to group several variables together into a single item known as a structure.

Table 11-2



Variable Declaration	Information Held
int EmpNumber;	Employee Number
char Name[25];	Employee's Name
float Hours;	Hours Worked
float PayRate;	Hourly Pay Rate
float GrossPay;	Gross Pay



Table 11-2 As a Structure:

```

struct PayRoll
{
    int EmpNumber;
    char Name[25];
    float Hours;
    float PayRate;
    float GrossPay;
};

```

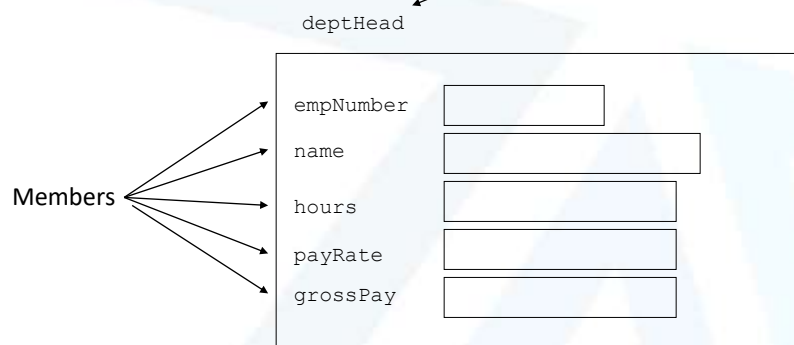
<https://manara.edu.sy/>

7



Figure 11-1

Structure Variable Name



<https://manara.edu.sy/>

8



Figure 11-2

deptHead

empNumber	<input type="text"/>
name	<input type="text"/>
hours	<input type="text"/>
payRate	<input type="text"/>
grossPay	<input type="text"/>

foreman

empNumber	<input type="text"/>
name	<input type="text"/>
hours	<input type="text"/>
payRate	<input type="text"/>
grossPay	<input type="text"/>

associate

empNumber	<input type="text"/>
name	<input type="text"/>
hours	<input type="text"/>
payRate	<input type="text"/>
grossPay	<input type="text"/>

<https://manara.edu.sy/>

9



Two steps to implementing structures:

- Create the structure declaration. This establishes the tag (or name) of the structure and a list of items that are members.
- Declare variables (or instances) of the structure and use them in the program to hold data.

<https://manara.edu.sy/>

10



11.3 Accessing Structure Members

- The dot operator (.) allows you to access structure members in a program



Program 11-1

// This program demonstrates the use of structures.

```
#include <iostream.h>

struct PayRoll
{
    int empNumber; // Employee number
    char name[25]; // Employee's name
    float hours; // Hours worked
    float payRate; // Hourly Payrate
    float grossPay; // Gross Pay
};
```



Program continues

```
void main(void)
{
    PayRoll employee;    // Employee is a PayRoll structure
    cout << "Enter the employee's number: ";
    cin >> employee.empNumber;
    cout << "Enter the employee's name: ";
    cin.ignore();        // To skip the remaining '\n' character
    cin.getline(employee.name, 25);
    cout << "How many hours did the employee work? ";
    cin >> employee.hours;
    cout << "What is the employee's hourly payrate? ";
    cin >> employee.payRate;
    employee.grossPay = employee.hours * employee.payRate;
    cout << "Here is the employee's payroll data:\n";
    cout << "Name: " << employee.name << endl;
}
```



Program continues

```
    cout << "Number: " << employee.empNumber << endl;
    cout << "Hours worked: " << employee.hours << endl;
    cout << "Hourly Payrate: " << employee.payRate << endl;
    cout.precision(2);
    cout.setf(ios::fixed | ios::showpoint);
    cout << "Gross Pay: $" << employee.grossPay << endl;
}
```



Program Output with Example Input

Enter the employee's number: **489** [Enter]
 Enter the employee's name: **Jill Smith** [Enter]
 How many hours did the employee work? **40** [Enter]
 What is the employee's hourly payrate? **20** [Enter]
 Here is the employee's payroll data:
 Name: Jill Smith
 Number: 489
 Hours worked: 40
 Hourly Payrate: 20
 Gross Pay: \$800.00



Displaying a Structure

- The contents of a structure variable cannot be displayed by passing the entire variable to cout. For example, assuming *employee* is a PayRoll structure variable, the following statement will not work:

```
cout << employee << endl; //won't work!
```




Program 11-2

// This program uses a structure to hold geometric data about a circle.

```
#include <iostream.h>
#include <iomanip.h>
#include <math.h>      // For the pow function

struct Circle
{
    float radius;
    float diameter;
    float area;
};

const float pi = 3.14159;
```

<https://manara.edu.sy/>

17



Program continues

```
void main(void)
{
    Circle c;
    cout << "Enter the diameter of a circle: ";
    cin >> c.Diameter;
    c.Radius = C.Diameter / 2;
    c.Area = pi * pow(c.Radius, 2.0);
    cout << "The radius and area of the circle are:\n";
    cout.precision(2);
    cout.setf(ios::fixed | ios::showpoint);
    cout << "Radius: " << c.radius << endl;
    cout << "Area: " << c.area << endl;
}
```

<https://manara.edu.sy/>

18



Program Output with Example Input

Enter the diameter of a circle: **10 [Enter]**

The radius and area of the circle are:

Radius: 5

Area: 78.54



Strings as Structure Members

- When a character array is a structure member, use the same string manipulation techniques with it as you would with any other character array.



Program 11-3

// This program uses a structure to hold someone's first,
// middle, and last name.

```
#include <iostream.h>
#include <string.h>

struct Name
{
    char first[15];
    char middle[15];
    char last[15];
    char full[45];
};
```

<https://manara.edu.sy/>

21



Program continues

```
void main(void)
{
    Name person;
    cout << "Enter your first name: ";
    cin >> person.first;
    cout << "Enter your middle name: ";
    cin >> person.middle;
    cout << "Enter your last name: ";
    cin >> person.last;
    strcpy(person.full, person.first);
    strcat(person.full, " ");
    strcat(person.full, person.middle);
    strcat(person.full, " ");
    strcat(person.full, person.last);
    cout << "\nYour full name is " << person.full << endl;
}
```

<https://manara.edu.sy/>

22



Program Output with Example Input

Enter your first name: **Josephine** [Enter]

Enter your middle name: **Yvonne** [Enter]

Enter your last name: **Smith** [Enter]

Your full name is Josephine Yvonne Smith



11.4 Initializing a Structure

- The members of a structure variable may be initialized with starting values when the structure variable is declared.

```
struct GeoInfo
{
    char cityName[30];
    char state[3];
    long population;
    int distance;
};
GeoInfo location = {"Ashville", "NC", 50000, 28};
```



11.5 Arrays of Structures

- Arrays of structures can simplify some programming tasks.

```
struct BookInfo
{
    char title[50];
    char author[30];
    char publisher[25];
    float price;
};
BookInfo bookList[20];
```

<https://manara.edu.sy/>

25



Program 11-5

```
// This program stores, in an array of structures,
// the hours worked by 5 employees, and their hourly
// pay rates. (This is a modification of Program 7-11.)
```

```
#include <iostream.h>
```

```
struct PayInfo
{
    int hours;           // Hours Worked
    float payRate;      // Hourly Pay Rate
};
```

<https://manara.edu.sy/>

26

Program continues



```
void main(void)
{
    PayInfo workers[5]; // Array of 5 structures
    cout << "Enter the hours worked by 5 employees and their\n";
    cout << "hourly rates.\n";
    for (int index = 0; index < 5; index++)
    {
        cout << "Hours worked by employee #" << (index + 1);
        cout << ": ";
        cin >> workers[index].hours;
        cout << "Hourly pay rate for employee #";
        cout << (index + 1) << ": ";
        cin >> workers[index].payRate;
    }
}
```

<https://manara.edu.sy/>

27

Program continues



```
cout << "Here is the gross pay for each employee:\n";
cout.precision(2);
cout.setf(ios::fixed | ios::showpoint);
for (index = 0; index < 5; index++)
{
    float gross;
    gross = workers[index].hours * workers[index].payRate;
    cout << "Employee #" << (index + 1);
    cout << ": $" << gross << endl;
}
}
```

<https://manara.edu.sy/>

28



Program Output with Example Input

Enter the hours worked by 5 employees and their hourly rates.

Hours worked by employee #1: **10 [Enter]**

Hourly pay rate for employee #1: **9.75 [Enter]**

Hours worked by employee #2: **15 [Enter]**

Hourly pay rate for employee #2: **8.62 [Enter]**

Hours worked by employee #3: **20 [Enter]**

Hourly pay rate for employee #3: **10.50 [Enter]**

Hours worked by employee #4: **40 [Enter]**

Hourly pay rate for employee #4: **18.75 [Enter]**

Hours worked by employee #5: **40 [Enter]**

Hourly pay rate for employee #5: **15.65 [Enter]**

Here is the gross pay for each employee:

Employee #1: \$97.50

Employee #2: \$129.30

Employee #3: \$210.00

Employee #4: \$750.00

Employee #5: \$626.00

<https://manara.edu.sy/>

29



Initializing a Structure Array

```
PayInfo workers[5] = {{ 10, 9.75},
                    {15, 8.62},
                    {20, 10.50},
                    {40, 18.75},
                    {40, 15.65}};
```

<https://manara.edu.sy/>

30



11.6 Focus on Software Engineering: Nested Structures

It's possible for a structure variable to be a member of another structure variable.

```
struct Costs
{
    float wholesale;
    float retail;
};
struct Item
{
    char partNum[10]
    char description[25];
    Costs pricing;
};
```

<https://manara.edu.sy/>

31



Program 11-6

```
// This program shows a structure with two nested structure members.
#include <iostream.h>
struct Date
{
    int month;
    int day;
    int year;
};
struct Place
{
    char address[50];
    char city[20];
    char state[15];
    char zip[11];
};
```

<https://manara.edu.sy/>

32



Program continues

```
struct EmpInfo
{
    char name[50];
    int empNumber;
    Date birthDate;
    Place residence;
};

void main(void)
{
    EmpInfo manager;
    // Ask for the manager's name and employee number
    cout << "Enter the manager's name: ";
    cin.getline(manager.name, 50);
    cout << "Enter the manager's employee number: ";
    cin >> manager.empNumber;
```

<https://manara.edu.sy/>

33



Program continues

```
// Get the manager's birth date
cout << "Now enter the manager's date-of-birth.\n";
cout << "Month (up to 2 digits): ";
cin >> manager.birthDate.month;
cout << "Day (up to 2 digits): ";
cin >> manager.birthDate.day;
cout << "Year (2 digits): ";
cin >> manager.birthDate.year;
cin.get(); // Eat the remaining newline character
// Get the manager's residence information
cout << "Enter the manager's street address: ";
cin.getline(manager.residence.address, 50);
cout << "City: ";
cin.getline(manager.residence.city, 20);
cout << "State: ";
cin.getline(manager.residence.state, 15);
```

<https://manara.edu.sy/>

34



Program continues

```

cout << "Zip Code: ";
cin.getline(manager.residence.zip, 11);
// Display the information just entered
cout << "\nHere is the manager's information:\n";
cout << manager.name << endl;
cout << "Employee number " << manager.empNumber << endl;
cout << "Date of Birth: ";
cout << manager.birthDate.month << "-";
cout << manager.birthDate.day << "-";
cout << manager.birthDate.year << endl;
cout << "Place of residence:\n";
cout << manager.residence.address << endl;
cout << manager.residence.city << ", ";
cout << manager.residence.state << " ";
cout << manager.residence.zip << endl;
}

```

<https://manara.edu.sy/>

35



Program Output with Example Input:

```

Enter the manager's name: John Smith [Enter]
Enter the manager's employee number: 789 [Enter]
Now enter the manager's date-of-birth
Month (up to 2 digits): 10 [Enter]
Day (up to 2 digits): 14 [Enter]
Year (2 digits): 65 [Enter]
Enter the manager's street address: 190 Disk Drive [Enter]
City: Redmond [Enter]
State: WA [Enter]
Zip Code: 98052 [Enter]
Here is the manager's information:
John Smith
Employee number 789
Date of birth: 10-14-65
Place of residence:
190 Disk Drive
Redmond, WA 98052

```

<https://manara.edu.sy/>

36



11.7 Structures as Function Arguments

Structure variables may be passed as arguments to functions.



Figure 11-3

```
showRect (box) ;  
  
void showRect(Rectangle r)  
{  
    cout << r.length << endl;  
    cout << r.width << endl;  
    cout << r.area << endl;  
}
```

Program 11-7



// This program demonstrates a function that accepts
// a structure variable as its argument.

```
#include <iostream.h>

struct InvItem
{
    int partNum;           // Part number
    char description[50]; // Item description
    int onHand;           // Units on hand
    float price;         // Unit price
};

void ShowItem(InvItem); // Function prototype
```

<https://manara.edu.sy/>

39

Program continues



```
void main(void)
{
    InvItem Part = {171, "Industrial Widget", 25, 150.0};
    ShowItem(part);
}

// Definition of function ShowItem. This function accepts
// an argument of the InvItem structure type. The contents
// of the structure is displayed.
void ShowItem(InvItem piece)
{
    cout.precision(2);
    cout.setf(ios::fixed | ios::showpoint);
    cout << "Part Number: " << piece.partNum << endl;
    cout << "Description: " << piece.description << endl;
    cout << "Units On Hand: " << piece.onHand << endl;
    cout << "Price: $" << piece.price << endl;
}
```

<https://manara.edu.sy/>

40

Program Output



Part Number: 171
 Description: Industrial Widget
 Units On Hand: 25
 Price: \$150.00

Program 11-8



// This program has a function that uses a structure reference variable
 // as its parameter.

```
#include <iostream.h>
struct InvItem
{
    int partNum;           // Part number
    char description[50]; // Item description
    int onHand;           // Units on hand
    float price;         // Unit price
};

// Function Prototypes
void GetItem(InvItem&);
void ShowItem(InvItem);
```



Program continues

```

void main(void)
{
    InvItem part;
    GetItem(part);
    ShowItem(part);
}
// Definition of function GetItem. This function uses a structure reference
// variable as its parameter. It asks the user for information to store in the
// structure.
void GetItem(InvItem &piece)
{
    cout << "Enter the part number: ";
    cin >> piece.partNum;
    cout << "Enter the part description: ";
    cin.get(); // Eat the remaining newline
    cin.getline(piece.description, 50);
}

```

<https://manara.edu.sy/>

43



Program continues

```

    cout << "Enter the quantity on hand: ";
    cin >> piece.onHand;
    cout << "Enter the unit price: ";
    cin >> piece.price;
}
// Definition of function ShowItem. This function accepts an argument of
// the InvItem structure type. The contents of the structure is displayed.
void ShowItem(InvItem piece)
{
    cout.precision(2);
    cout.setf(ios::fixed | ios::showpoint);
    cout << "Part Number: " << piece.partNum << endl;
    cout << "Description: " << piece.description << endl;
    cout << "Units On Hand: " << piece.onHand << endl;
    cout << "Price: $" << piece.price << endl;
}

```

<https://manara.edu.sy/>

44



Program Output

Enter the part number: **800** [Enter]
 Enter the part description: **Screwdriver** [Enter]
 Enter the quantity on hand: **135** [Enter]
 Enter the unit price: **1.25** [Enter]
 Part Number: **800**
 Description: **Screwdriver**
 Units On Hand: **135**
 Price: \$1.25



Constant Reference Parameters

- Sometimes structures can be quite large. Therefore, passing by value can decrease a program's performance. But passing by reference can cause problems.

- Instead, pass by constant reference:

```
void ShowItem(const InvItem &piece)
{
    cout.setf(ios::precision(2)|ios::fixed|ios::showpoint);
    cout << "Part Number: " << piece.partNum << endl;
    cout << "Price: $" << piece.price << endl;
}
```



11.8 Returning a Structure from a Function

- A function may return a structure.

```
struct Circle
{
    float radius, diameter, area;
};
Circle getData(void)
{
    Circle temp;
    temp.radius = 10.0;
    temp.diameter = 20.0;
    temp.area = 314.159;
    return temp;
}
```

<https://manara.edu.sy/>

47



Program 11-9

// This program uses a function to return a structure. This
// is a modification of Program 11-2.

```
#include <iostream.h>
#include <iomanip.h>
#include <math.h> // For the pow function
```

// Circle structure declaration

```
struct Circle
{
    float radius;
    float diameter;
    float area;
};
```

<https://manara.edu.sy/>

48



Program continues

```
// Function prototype
Circle getInfo(void);
// Constant definition for Pi
const float pi = 3.14159;
void main(void)
{
    Circle c;
    c = getInfo();
    c.area = pi * pow(c.radius, 2.0);
    cout << "The radius and area of the circle are:\n";
    cout.precision(2);
    cout.setf(ios::fixed | ios::showpoint);
    cout << "Radius: " << c.radius << endl;
    cout << "Area: " << c.area << endl;
}
```

<https://manara.edu.sy/>

49



Program continues

```
// Definition of function GetInfo. This function uses a
// local variable, Round, which is a Circle structure.
// The user enters the diameter of the circle, which is
// stored in Round.Diameter. The function then calculates
// the radius, which is stored in Round.Radius. Round is then
// returned from the function.
```

```
Circle getInfo(void)
{
    Circle Round;
    cout << "Enter the diameter of a circle: ";
    cin >> Round.Diameter;
    Round.Radius = Round.Diameter / 2;
    return Round;
}
```

<https://manara.edu.sy/>

50



Program Output with Example Input

Enter the diameter of a circle: **10 [Enter]**
 The radius and area of the circle are:
 Radius: 5.00
 Area: 78.54



11.9 Pointers to Structures

- You may take the address of a structure variable and create variables that are pointers to structures.

```
Circle *cirPtr;
```

```
CirPtr = &piePlate;
```

```
*cirPtr.Radius = 10; //incorrect way to access Radius because the dot operator has higher precedence
```

```
(*cirPtr).Radius = 10; //correct access to Radius
```

```
cirPtr->Radius = 10; //structure pointer operator, easier notation for dereferencing structure pointers
```



Program 11-10

```
// This program uses a structure pointer to dynamically allocate a
// structure
// variable in memory. It is a modification of Program 11-1.
#include <iostream.h>
#include <iomanip.h>
#include <stdlib.h>

struct PayRoll
{
    int empNumber;           // Employee number
    char name[25];          // Employee's name
    float hours;            // Hours worked
    float payRate;          // Hourly Payrate
    float grossPay;         // Gross Pay
};
```

<https://manara.edu.sy/>

53



Program continues

```
void main(void)
{
    PayRoll *employee;      // Employee is a pointer to a
                           // PayRoll structure
    employee = new PayRoll; // Dynamically allocate a struct
    if (employee == NULL)  // Test the allocated memory
    {
        cout << "Memory allocation error!\n";
        exit(EXIT_FAILURE);
    }
    cout << "Enter the employee's number: ";
    cin >> employee->empNumber;
    cout << "Enter the employee's name: ";
    cin.ignore();          // To skip the remaining '\n' character
    cin.getline(employee->name, 25);
```

<https://manara.edu.sy/>

54



Program continues

```

cout << "How many hours did the employee work? ";
cin >> employee->hours;
cout << "What is the employee's hourly payrate? ";
cin >> employee->payRate;
employee->grossPay = employee->hours * employee->payRate;

cout << "Here is the employee's payroll data:\n";
cout << "Name: " << employee->name << endl;
cout << "Number: " << employee->empNumber << endl;
cout << "Hours worked: " << employee->hours << endl;
cout << "Hourly Payrate: " << employee->payRate << endl;
cout.precision(2);
cout.setf(ios::fixed | ios::showpoint);
cout << "Gross Pay: $" << employee->grossPay << endl;
delete employee; // Free the allocated memory
}

```

<https://manara.edu.sy/>

55



Program Output with Example Input

```

Enter the employee's number: 489 [Enter]
Enter the employee's name: Jill Smith [Enter]
How many hours did the employee work? 40 [Enter]
What is the employee's hourly payrate? 20 [Enter]
Here is the employee's payroll data:
Name: Jill Smith
Number: 489
Hours worked: 40
Hourly Payrate: 20
Gross Pay: $800.00

```

<https://manara.edu.sy/>

56



Program 11-11

// This program demonstrates a function that uses a
// pointer to a structure variable as a parameter.

```
#include <iostream.h>
#include <iomanip.h>
struct Student
{
    char name[35];           // Student's name
    int idNum;              // Student ID number
    int crdHrs;             // Credit hours enrolled
    float gpa;              // Current GPA
};

void getData(Student *); // Function prototype
```

<https://manara.edu.sy/>

57



Program continues

```
void main(void)
{
    Student freshman;
    cout << "Enter the following student data:\n";
    getData(&freshman);
    cout << "\nHere is the student data you entered:\n";
    cout.precision(2);
    // Now display the data stored in Freshman
    cout << "Name: " << freshman.name << endl;
    cout << "ID Number: " << freshman.idNum << endl;
    cout << "Credit Hours: " << freshman.crdHrs << endl;
    cout << "GPA: " << freshman.gpa << endl;
}
```

<https://manara.edu.sy/>

58



Program continues

```
// Definition of function GetData. Uses a pointer to a
// Student structure variable. The user enters student
// information, which is stored in the variable.
void getData(Student *s)
{
    cout << "Student Name: ";
    cin.getline(s->name, 35);
    cout << "Student ID Number: ";
    cin.ignore(); // Ignore the leftover newline
    cin >> s->idNum;
    cout << "Credit Hours Enrolled: ";
    cin >> s->crdHrs;
    cout << "Current GPA: ";
    cin >> s->gpa;
}
```



Program Output with Example Input

Enter the following student data:
 Student Name: **Frank Smith [Enter]**
 Student ID Number: **4876 [Enter]**
 Credit Hours Enrolled: **12 [Enter]**
 Current GPA: **3.45 [Enter]**
 Here is the student data you entered:
 Name: Frank Smith
 ID Number: 4876
 Credit Hours: 12
 GPA: 3.45



11.10 Focus on Software Engineering: When to Use ., When to Use ->, and When to Use *

Table 11-3

Expression	Description
$s \rightarrow m$	s is a structure pointer and m is a member. This expression accesses the m member of the structure pointed to by s .
$*a.p$	a is a structure variable and p , a pointer, is a member. This expression dereferences the value pointed to by p .

<https://manara.edu.sy/>

61

$(*s).m$	s is a structure pointer and m is a member. The $*$ operator dereferences s , causing the expression to access the m member of the structure pointed to by s . This expression is the same as $s \rightarrow m$.
$*s \rightarrow p$	s is a structure pointer and p , a pointer, is a member of the structure pointed to by s . This expression accesses the value pointed to by p . (The \rightarrow operator dereferences s and the $*$ operator dereferences p .)
$*(*s). p$	s is a structure pointer and p , a pointer, is a member of the structure pointed to by s . This expression accesses the value pointed to by p . ($(*s)$ dereferences s and the outermost $*$ operator dereferences p . This expression is the same as $*s \rightarrow p$.)

<https://manara.edu.sy/>

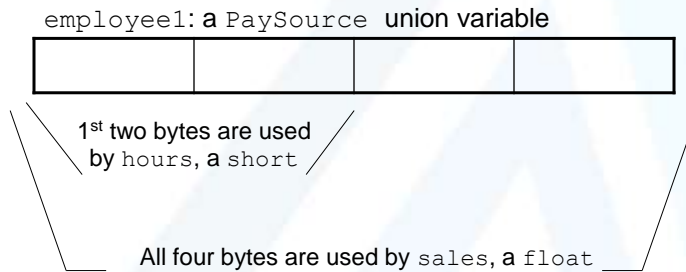
62



11.11 Unions

- A union is like a structure, except all the members occupy the same memory area.

Figure 11-4



<https://manara.edu.sy/>

63



Program 11-12

```
// This program demonstrates a union.
#include <iostream.h>
#include <iomanip.h>
#include <ctype.h> // For toupper

union PaySource
{
    short hours;
    float sales;
};

void main(void)
{
    PaySource employee1;
    char payType;
    float payRate, grossPay;
```

<https://manara.edu.sy/>

64



Program continues

```

cout.precision(2);
cout.setf(ios::showpoint);
cout << "This program calculates either hourly wages or\n";
cout << "sales commission.\n";
cout << "Enter H for hourly wages or C for commission: ";
cin >> payType;

if (toupper(payType) == 'H')
{
    cout << "What is the hourly pay rate? ";
    cin >> payRate;
    cout << "How many hours were worked? ";
    cin >> employee1.hours;
    GrossPay = employee1.hours * payRate;
    cout << "Gross pay: $" << grossPay << endl;
}

```



Program continues

```

else if (toupper(payType) == 'C')
{
    cout << "What are the total sales for this employee? ";
    cin >> employee1.sales;
    grossPay = employee1.sales * 0.10;
    cout << "Gross pay: $" << grossPay << endl;
}

else
{
    cout << payType << " is not a valid selection.\n";
}
}

```



Program Output with Example Input

This program calculates either hourly wages or sales commission.

Enter H for hourly wages or C for commission: **C [Enter]**

What are the total sales for this employee? **5000 [Enter]**

Gross pay: \$500.00



Anonymous Unions

- The members of an anonymous union have names, but the union itself has no name.

```
Union
```

```
{
  member declarations;
  ...
};
```

Program 11-13



// This program demonstrates an anonymous union.

```
#include <iostream.h>
#include <iomanip.h>
#include <ctype.h>           // For toupper

void main(void)
{
    union                    // Anonymous union
    {
        short hours;
        float sales;
    };
    char payType;
    float payRate, grossPay;
```

<https://manara.edu.sy/>

69

Program continues

```
cout.precision(2);
cout.setf(ios::fixed | ios::showpoint);
cout << "This program calculates either hourly wages or\n";
cout << "sales commission.\n";
cout << "Enter H for hourly wages or C for commission: ";
cin >> payType;

if (toupper(payType) == 'H')
{
    cout << "What is the hourly pay rate? ";
    cin >> payRate;
    cout << "How many hours were worked? ";
    cin >> hours;
    union member                    // Anonymous
    {
        grossPay = hours * payRate;
        cout << "Gross pay: $" << grossPay << endl;
    }
}
```

<https://manara.edu.sy/>

70



Program continues

```

else if (toupper(payType) == 'C')
{
    cout << "What are the total sales for this employee? ";
    cin >> sales; //
    Anonymous union member
    grossPay = sales * 0.10;
    cout << "Gross pay: $" << grossPay << endl;
}

else
{
    cout << payType << " is not a valid selection.\n";
}
}

```

<https://manara.edu.sy/>

71



Program Output with Example Input

This program calculates either hourly wages or sales commission.

Enter H for hourly wages or C for commission: **C [Enter]**

What are the total sales for the employee? **12000 [Enter]**

Gross pay: \$1200.00

<https://manara.edu.sy/>

72