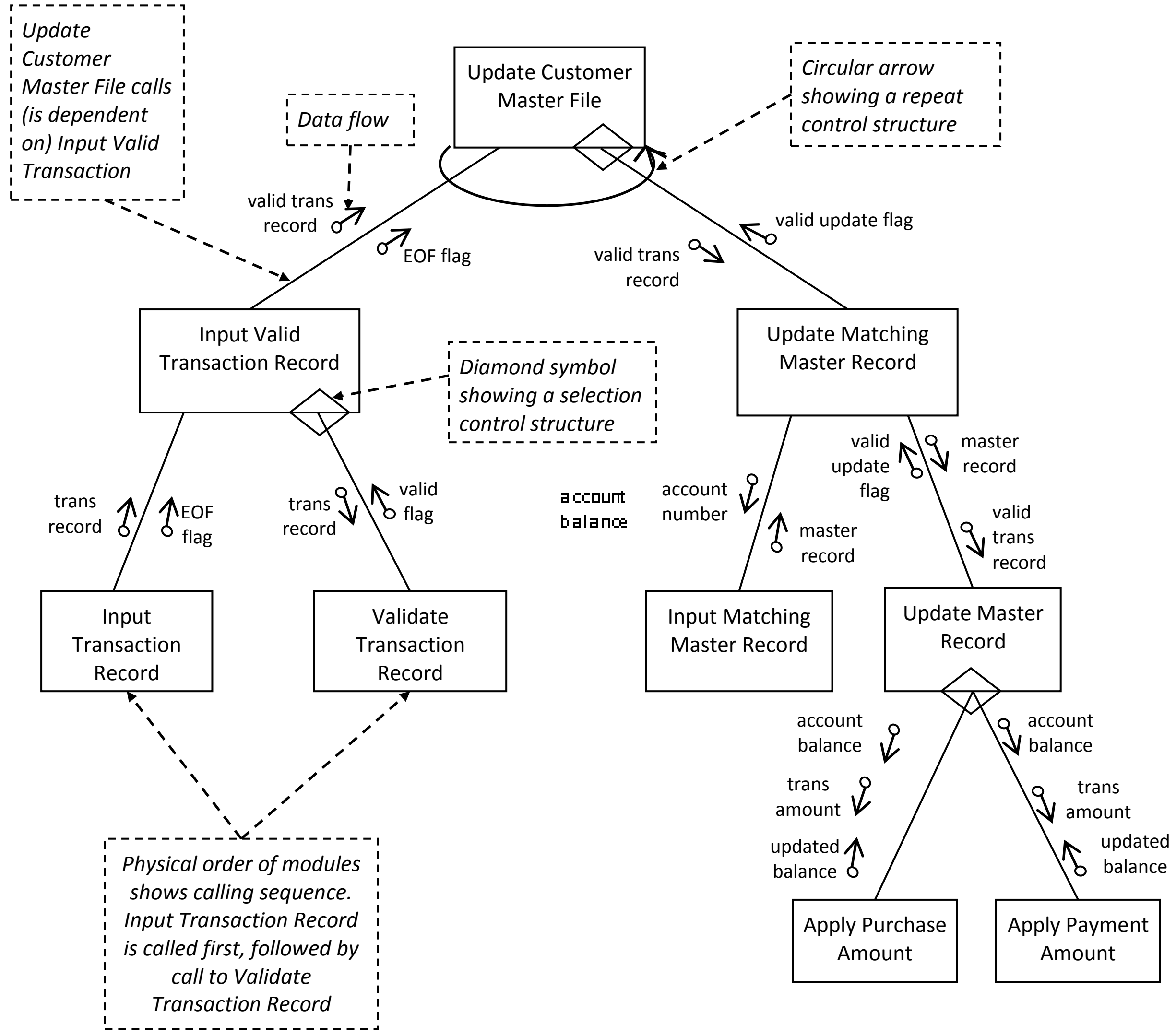
UECS2344 Software Design: Lecture 1

Procedural / Structured Approach

Structure Chart Example
(adapted from Structured Systems Development (Powers, Cheney, Crow))



Update Customer Master File calls (is dependent on) Input Valid Transaction

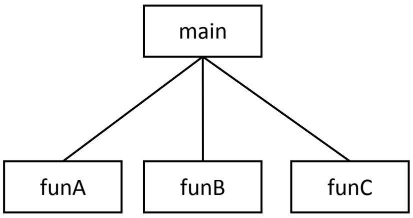
Data flow

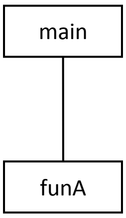
Circular arrow showing a repeat control structure

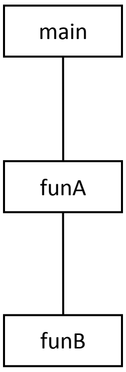
Diamond symbol showing a selection control structure

Physical order of modules shows calling sequence. Input Transaction Record is called first, followed by call to Validate Transaction Record

Code	Output	Structure Chart

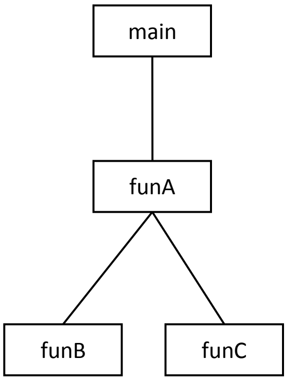
int main(void)
{
      funA();
      funB();
      funC();
}
 	
void funA()
{
      cout << “A called” << endl;
}
 	
void funB()
{
      cout << “B called” << endl;
}
 	
void funC()
{
      cout << “C called” << endl;
}
	A called
	B called
	C called
		

int main(void)
{
      funA();
}
 	
void funA()
{
      cout << “A called” << endl;
}
	A called	

int main(void)
{
      funA();
}
 	
void funA()
{
      cout << “A called” << endl;
      funB();
}
 	
void funB()
{
      cout << “B called” << endl;
}
	A called
	B called
		

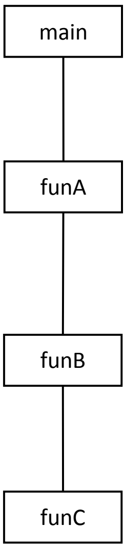
```
int main(void)
{
      funA();
}

void funA()
{
      cout << “A called” << endl;
      funB();
      funC();
}

void funB()
{
      cout << “B called” << endl;
}

void funC()
{
      cout << “C called” << endl;
}
```

```
A called
B called
C called
```



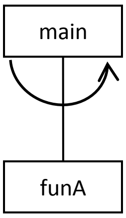
```
int main(void)
{
      funA();
}

void funA()
{
      cout << “A called” << endl;
      funB();

}

void funB()
{
      cout << “B called” << endl;
      funC();
}

void funC()
{
      cout << “C called” << endl;
}
```

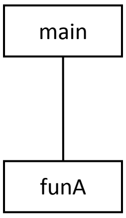
```
A called
B called
C called
```

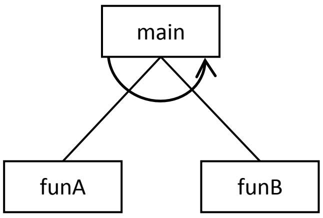


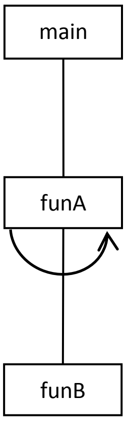
```
int main(void)
{
      for(int i=0; i<3; i++)
      {
          funA();
      }
}

void funA()
{
      cout << “A called” << endl;
}
```

```
A called
A called
A called
```



int main(void)
{
for(int i=0; i<3; i++)
{
cout << i << endl;
}
 	
funA();
}
 	
void funA()
{
cout << “A called” << endl;
}
	0
	1
	2
	A called
		

int main(void)
{
for(int i=0; i<3; i++)
{
funA();
funB();
}
}
 	
void funA()
{
cout << “A called” << endl;
}
 	
void funB()
{
cout << “B called” << endl;
}
	A called
	B called
	A called
	B called
	A called
	B called
		

int main(void)
{
funA();
 	
}
 	
void funA()
{
cout << “A called” << endl;
 	
for (int i=0; i<5; i++)
{
funB();
}
 	
}
 	
void funB()
{
cout << “B called” << endl;
}
	A called
	B called
	B called
	B called
	B called
	B called
		

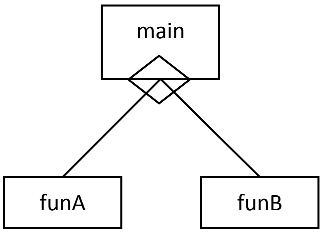
<table>
<tr>
<td>

```
int main(void)
{
     int number;

     cout << “Enter 1 or 2: ”;
     cin >> number;
     if (number == 1)
         funA();
     else
         funB();
}

void funA()
{
     cout << “A called” << endl;
}

void funB()
{
     cout << “B called” << endl;
}
```

</td>
<td>Enter 1 or 2: 1
A called

OR

Enter 1 or 2: 2
B called</td>
<td>



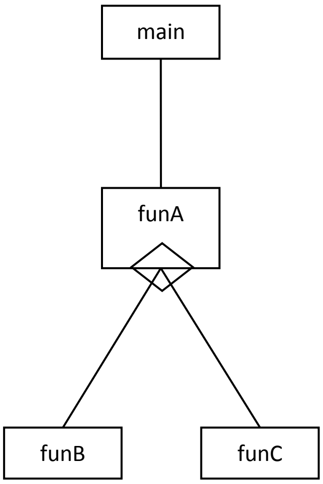
</td>
</tr>
<tr>
<td>

```
int main(void)
{
     funA();
}

void funA()
{
     int number;

     cout << “A called” << endl;

     cout << “Enter 1 or 2: ”;
     cin >> number;

     if (number == 1)
          funB();
     else
          funC();
}

void funB()
{
     cout << “B called” << endl;
}

void funC()
{
     cout << “C called” << endl;
}
```

</td>
<td>A called
Enter 1 or 2: 1
B called

OR

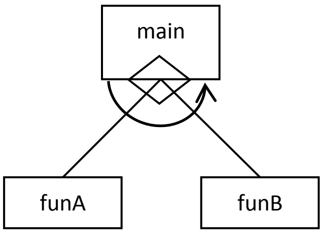
A called
Enter 1 or 2: 2
C called</td>
<td>



</td>
</tr>
</table>

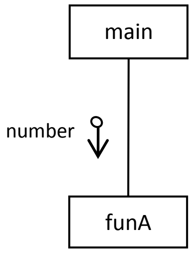
```
int main(void)
{
     int number;

     for (int i=0; i<3; i++)
     {
        cout << “Enter 1 or 2: ”;
        cin >> number;
        if (number == 1)
            funA();
        else
            funB();
     }
}

void funA()
{
     cout << “A called” << endl;
}

void funB()
{
     cout << “B called” << endl;
}
```

```
Enter 1 or 2 : 1
A called
Enter 1 or 2: 2
B called
Enter 1 or 2: 2
B called
```



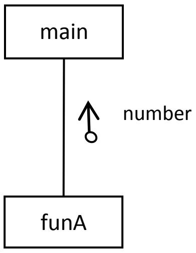
```
int main(void)
{
     int number;

     cout << “Enter number: ”;
     cin >> number;

     funA(number);
}

void funA(int number)
{
     cout << “A called” << endl;
     cout << “Number is ”
          << number << endl;
}
```

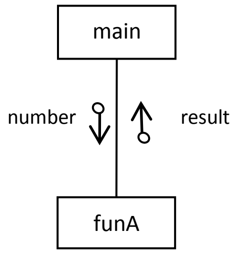
```
Enter number: 10
A called
Number is 10
```

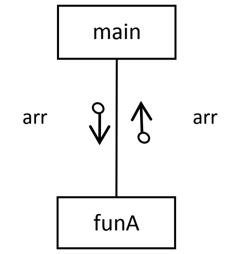


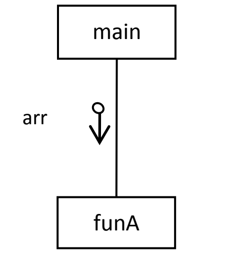
```
int main(void)
{
     int number;

     number = funA();
     cout << “Number is ”
          << number << endl;
}

int funA()
{
     int number;

     cout << “A called” << endl;
     cout << “Enter number: ”;
     cin >> number;

     return number;
}
```

```
A called
Enter number: 10
Number is 10
```



int main(void)
{
     int number, result;
 	
     cout << “Enter number: ”;
     cin >> number;
 	
     result = funA(number);
 	
     cout << “Result is ”
          << result << endl;
}
 	
int funA(int number)
{
     int result;
 	
     cout << “A called” << endl;
     result = number * 2;
     return result;
}
	Enter number: 10
	A called
	Result is 20
		

int main(void)
{
     int arr[5];
 	
     funA(arr);
 	
     cout << “In main” << endl;
     for (int i=0; i<5; i++)
         cout << arr[i] << endl;
}
 	
void funA(int arr[])
{
     cout << “A called” << endl;
 	
     for (int i=0; i<5; i++)
          arr[i] = i;
}
	A called
	In main
	0
	1
	2
	3
	4
		
		funA changes values of the array

int main(void)
{
     int arr[5];
 	
     for (int i=0; i<5; i++)
         arr[i] = i;
 	
     funA(arr);
}
 	
void funA(int arr[])
{
     cout << “A called” << endl;
 	
     cout << “In funA” << endl;
     for (int i=0; i<5; i++)
         cout << arr[i] << endl;
}
	A called
	In funA
	0
	1
	2
	3
	4
		
		funA does not change values of the array

Data Dictionary

Contains descriptions of data.

Notation:

Symbols	Meanings
+	and
{ ... }	many
(...)	optional
[... | ...]	alternatives

Example:

Employee_File = { employee_record }

employee_record = employee_number
+ employee_name
+ identification_number
+ department
+ year_joined
+ { positions_held }
+ academic_qualification
+ (professional_qualification)

identification_number = [IC_number | passport_number]

positions_held = position
+ appointment_start_date
+ (appointment_end_date)

timesheet = employee_number
+ employee_name
+ { daily_work_record }

daily_work_record = date
+ day_of_week
+ time_clocked_in
+ time_clocked_out

Exercise

Consider the C++ program below which uses the procedural / structured approach.

1. Draw a **Structure Chart** which shows all the functions in the program and their calling structure. Include the data flows.
2. Create **Data Dictionary** entries for the data flows, particularly for the structure and array used.

C++ program using array of structures

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

#define MAX_NUMBER_OF_POINTS 100

// Point record (structure)
struct Point
{
        string name; // name of Point
        int x;       // x-coordinate of Point
        int y;       // y-coordinate of Point
};

// function prototypes for application
int  createPoints(Point[]);
void processPoints(Point[], int);
void displayAllPoints(Point[], int);
void moveAllLeft(Point[], int);
void moveAllRight(Point[], int);
void moveAllUp(Point[], int);
void moveAllDown(Point[], int);
void moveLeft(Point&);
void moveRight(Point&);
void moveUp(Point&);
void moveDown(Point&);

// the application
int main(void)
{
        Point pointsArray[MAX_NUMBER_OF_POINTS];
        int pointsCount;

        pointsCount = createPoints(pointsArray);

        processPoints(pointsArray, pointsCount);

        return 0;
}

// function definitions
int createPoints(Point pointsArray[])
{
        Point aPoint;
        string endlChar;
        int pointsCount;
```

```
    cout << "How many points do you want? (minimum is 1, maximum is 100): ";
    cin >> pointsCount;

    // create the points
    for (int i = 0; i < pointsCount; i++)
    {
        cout << "Enter name: ";
        // clear previous input before reading string
        getline(cin, endlChar);
        getline(cin, aPoint.name);
        cout << "Enter x value: ";
        cin >> aPoint.x;
        cout << "Enter y value: ";
        cin >> aPoint.y;
        cout << endl;

        pointsArray[i] = aPoint;
    }
    return pointsCount;
}

void processPoints(Point pointsArray[], int pointsCount)
{
    int choice;
    do
    {
        cout << "Do you want to move all points" << endl;
        cout << "1. Left" << endl;
        cout << "2. Right" << endl;
        cout << "3. Up" << endl;
        cout << "4. Down" << endl;
        cout << "5. Or Display all points" << endl;
        cout << "6. Or Exit" << endl;

        cout << "Enter your choice (1-6): ";
        cin >> choice;
        while (choice < 1 || choice > 6) {
            cout << "Invalid choice. Please enter again:" << endl;
            cin >> choice;
        }

        switch (choice)
        {
        case 1: moveAllLeft(pointsArray, pointsCount); break;
        case 2: moveAllRight(pointsArray, pointsCount); break;
        case 3: moveAllUp(pointsArray, pointsCount); break;
        case 4: moveAllDown(pointsArray, pointsCount); break;
        case 5: displayAllPoints(pointsArray, pointsCount); break;
        default: break;
        }

        cout << endl;

    } while (choice != 6);
}

void moveAllLeft(Point pointsArray[], int pointsCount)
{
    for (int i = 0; i < pointsCount; i++)
```

```
        {
                moveLeft((pointsArray[i])); // pass by reference
        }
}

void moveAllRight(Point pointsArray[], int pointsCount)
{
        for (int i = 0; i < pointsCount; i++)
        {
                moveRight((pointsArray[i])); // pass by reference
        }
}

void moveAllUp(Point pointsArray[], int pointsCount)
{
        for (int i = 0; i < pointsCount; i++)
        {
                moveUp((pointsArray[i])); // pass by reference
        }
}

void moveAllDown(Point pointsArray[], int pointsCount)
{
        for (int i = 0; i < pointsCount; i++)
        {
                moveDown((pointsArray[i])); // pass by reference
        }
}

void displayAllPoints(Point pointsArray[], int pointsCount)
{
        cout << "Position of points are:" << endl;
        for (int i = 0; i < pointsCount; i++)
        {
                cout << "Point: " << (pointsArray[i]).name
                        << " x=" << (pointsArray[i]).x
                        << ",y=" << (pointsArray[i]).y << endl;
        }
}

void moveLeft(Point& aPoint)
{
        aPoint.x = aPoint.x - 1; // by reference
}

void moveRight(Point& aPoint)
{
        aPoint.x = aPoint.x + 1; // by reference
}

void moveUp(Point& aPoint)
{
        aPoint.y = aPoint.y + 1; // by reference
}

void moveDown(Point& aPoint)
{
        aPoint.y = aPoint.y - 1; // by reference
}
```