

Top - Down Design -1

- Also called Modular Design and Structured Design
- Develop a program by dividing problem into sub-modules that are more easily coded and debugged.
- Then after sub-modules are coded, debugged, and tested, Assemble modules to create program.

Copyright © 2001 R.M. Laurie 1

Top - Down Design 2

- Best to do Top Flow Chart first.
- Create flow charts for modules.
- Code modules.
- Debug Test Debug Test.
- Assemble modules into program.
- Debug Test Debug Test.

Copyright © 2001 R.M. Laurie 2

Top - Down Design 3

- Don't forget to test all possible selection structure paths of final program.
- Use breakpoints and watch variables in the testing phase as well as debugging phase.

Copyright © 2001 R.M. Laurie 3

Functions

- Vehicle for achieving Top Down Design
- Programs constructed in a Modular Way
- Can test individual functions before assembly
- Code Reuse

Copyright © 2001 R.M. Laurie 4

Library Functions

- Set of functions which can be called by your program (pp. 118 – 123, Appendix C)
- Requires inclusion of header file at beginning of program
 - #include <iostream>
 - #include <cmath>
 - #include <cstdlib>
 - #include <cctype>
 - #include <cstring>
 - using namespace std;
- Header file provides declaration or "prototype" of the function

Copyright © 2001 R.M. Laurie 5

cmath Functions

- | | |
|--------------------|-----------------|
| • abs | • exp, expl |
| • acos, acosl | • log, logl |
| • cosh, coshl | • log10 |
| • sin, sinl | • pow, powl |
| • tan, tanl | • pow10, pow10l |
| • complex (struct) | • sqrt, sqrtl |

Copyright © 2001 R.M. Laurie 6

cstdlib Functions

- srand
- malloc
- strtol
- free
- rand
- random
- randomize
- atof
- atoi
- atol
- min
- time

Copyright © 2001 R.M. Laurie 7

sqrt() Function

```
#include <cmath>
double sqrt(double x);
long double sqrtl(long double x);
```

Description
 Calculates the positive square root.
 sqrt calculates the positive square root of the argument x.
 sqrtl is the long double version; it takes a long double argument and returns a long double result.

Copyright © 2001 R.M. Laurie 8

pow() Function

```
#include <cmath>
double pow(double x, double y);
long double powl(long double x, long double y);
```

Description
 Calculates x to the power of y.
 powl is the long double version; it takes long double arguments and returns a long double result.

Copyright © 2001 R.M. Laurie 9

Library Function Example

```
#include <cmath>
#include <iostream>
using namespace std;
int main(void)
{
    double dfA, dfB=4.0;
    dfA = sqrt(dfB);
    cout << dfA << endl;
    dfA = sqrt(dfA);
    cout << dfA << endl;
    dfA = pow(pow(dfA, dfB), 3);
    cout << dfA << endl << sqrt(dfA) end;
    return 0;
}
```

← Declares Library Functions

2
1.41421
64
8

Copyright © 2001 R.M. Laurie 10

User Defined Functions

- User functions can be created that modularize a program
- Good for divide and conquer solution approach Best for repeated sections
- Best for blocks with only one result
- Naming Convention:
 - Use Title Case for User Functions
 - CalcArea PrintGraph GetData

Copyright © 2001 R.M. Laurie 11

Organization of Source Code

```
void CalcA(); ← Prototypes
void PrintB(); (Note: end with ';' )
int main()
{
    CalcA(); ← Function Calls
    PrintB();
}
void CalcA() ← Function Definition 1
{
    ... (Note: Doesn't end with ';' )
}
void PrintB() ← Function Definition 2
{
    ... (Note: Doesn't end with ';' )
}
```

Copyright © 2001 R.M. Laurie 12

Organization of Source Code

```

void CalcA(); ← Prototypes
void PrintB();
int main()
{
    CalcA(); ← Function Calls
    PrintB();
}
void CalcA()
{
...
}
void PrintB()
{
...
}
    
```

Copyright © 2001 R.M. Laurie 13

Organization of Source Code

```

void CalcA(); ← Prototypes
void PrintB();
int main()
{
    CalcA(); ← Function Call
}
void CalcA()
{
    PrintB(); ← Function Call
}
void PrintB()
{
...
}
    
```

Copyright © 2001 R.M. Laurie 14

Organization of Source Code

```

void CalcA(); ← Prototypes
void PrintB();
int main()
{
    CalcA(); ← Function Call
    PrintB(); ← Function Call
    CalcA(); ← Function Call
}
void CalcA()
{
    PrintB(); ← Function Call
}
void PrintB()
{
...
}
    
```

Copyright © 2001 R.M. Laurie 15

User Functions

- Void Functions do not return a value to their caller.
- May contain parameters.
- Prototype of void user function
 void CalcArea(double);
 void PrintArea(double, double);
- Calling a void user function
 CalcArea(6);
 PrintArea(6, dfArea);

Copyright © 2001 R.M. Laurie 16

Example Problem

- Create a program that prompts and receives the radius from the user
- Calculate and display the circumference and area of a circle with this radius
- Create three functions in your program

Copyright © 2001 R.M. Laurie 17

```

// PROTOTYPES = Function Declarations
#include <iostream>
void CalcCircumf(float);
float CalcArea(float);

float CalcSquare(float);
using namespace std;
// GLOBAL VARIABLE DECLARATIONS */
const double PI = 3.141592653;
int main(void)
{
    float fRadius, fArea;

    cout << "Enter Circle Radius\n";
    cin >> fRadius;
    CalcCircumf(fRadius);
    fArea = CalcArea(fRadius);
    cout << "Area = " << fArea << endl;
    return(0);
}

void CalcCircumf(float fRad)
{
    float fCirc;
    fCirc = PI * fRad * 2;
    cout << "Circumference = " << fCirc << endl;
}

float CalcArea(float fRad)
{
    return( PI * CalcSquare(fRad));
}

float CalcSquare(float fValue)
{
    return( fValue * fValue);
}
    
```

Copyright © 2001 R.M. Laurie 18

```

Enter Circle Radius
>10
Circumference = 62.83
Area = 314.15
    
```

Prototypes

```
// PROTOTYPES = Function Declarations
#include <iostream>

void CalcCircumf(float);
float CalcArea(float);
float CalcSquare(float);

using namespace std;
```

Copyright © 2001 R.M. Laurie 19

main function

```
// GLOBAL VARIABLE DECLARATIONS
const double PI = 3.141592653;

int main(void)
{
    float fRadius, fArea;

    cout << "Enter Circle Radius\n>";
    cin >> fRadius;
    CalcCircumf(fRadius);
    fArea = CalcArea(fRadius);
    cout << "Area = " << fArea << endl;
    return(0);
}
```

Copyright © 2001 R.M. Laurie 20

Other Functions

```
void CalcCircumf(float fRad)
{
    float fCirc;
    fCirc = PI * fRad * 2;
    cout << "Circumference = " << fCirc <<
    endl;
}

float CalcArea(float fRd )
{
    return( PI * CalcSquare(fRd));
}

float CalcSquare( float fValue)
{
    return( fValue * fValue);
}
```

Copyright © 2001 R.M. Laurie 21