

Programming and Problem Solving

-Problem Analysis and Specification

-Problem Statement

-Input

-Output

-Data Organization & Algorithm Design

-Theory

-Algorithm (flowchart/pseudocode)

-Program Coding

-Execution and Testing

-Example

(Homework steps in **bold**)

Conditional (Ternary) Operator “?”

expr1 ? expr2: expr3

(note the “:” not a semicolon)

-If expr 1 is true, expr2 is evaluated.

Otherwise, expr3 is evaluated.

Input and Output

-printf(“Enter a number: “); //output

-scanf(“%f”, &Num); //input

-& is the address operator and is not always used.

-“\n” denotes a new line

Increment/Decrement Operators

X++; //expression is equal to X, then X is increased by 1.

++X; //X is increased, and expression is equal to X+1.

X--; //expression is equal to X, then X is decreased by 1.

--X; //X is decreased, and expression is equal to X-1.

Basic Data/Variable Types

-Integer: whole numbers, no decimals: int Num1 = 1; (format descriptor %d)

-Float: numbers with decimal points: float Num2 = 2.0; (format descriptor %f)

-Double: twice as precise as float: double Num3 = 3.0; (format descriptor %lf)

-Character: string of length 1: char Grade = ‘C’; (format descriptor %c)

-String: series of characters: char Name = ‘Imbrie’; (format descriptor %s)

-atoi(String), atof(String), and atol(String) convert strings/characters into numbers

-strncpy(String2, String1, 5); copies the first 5 characters of String 1 into String2

-strncpy(String2, String1+4, 2); makes a length 2 substring of String1 starting at 4

-Other types: short int, long int, long double, unsigned, signed, etc.

-#define variables for them to be constant i.e. #define pi 4.0*atan(1.0) //no “=” sign

Arrays

-IMPORTANT: The index of all arrays starts from 0.

-int ABC[3] = {1, 2, 3}; //can initialize when you define

-float Values[100][7]; //2-D array of 700 float values

-int Nums[2][3] = {{1, 3, 5}, {2, 4, 6}}; //initialize 2-D

-char Bagel[] = “Bites”; //strings are arrays

-Strings need one extra space to store \0 in the final cell.

for example, Bagel[] would be {B, I, T, E, S, \0}, length 6.

-To pass arrays to functions: function(ArrayName); ... void function(ArrayName[]){ }

Making a C program (Basic Structure)

```
#include <stdio.h>
int func(int A); //declare function before use
float main(void) //must have a main function
{
    ...
    C = func(B); //use a function
    return B*2.0;
}
int func(int A)
{
    ...
    return A;
}
```

Simple Arithmetic Operations

- + addition
- subtraction
- / division
- * multiplication
- % modulus (remainder of division)
- pow(x,y) exponentiation (math.h)

Conditionals

```
if (logical expression)
{
    ...
}
else if (logical expression)
{
    ...
}
else
{
    ...
}
```

Libraries to #include <>

- stdio.h //Input & Output
- stdlib.h //Standard Library
- string.h //Strings
- math.h //Math functions

Loops

```
while (logical expression)
{
    ...
    if (logical expression2)
        break; //exit while/do loop
}
```

```
do
{
    ...
} while (logical expression); //don't forget the ";" here
```

```
for(expr1; expr2; expr3)
{
    //expr1 initializes loop variable
    //expr2 is checked every iteration
    //if true, expr3 is used to update loop
    ...
    if (logical expression2)
        continue; //go to next iteration
}
```

Logic Operations

- == equal to
- != not equal to
- > greater than
- >= greater than or equal to
- < less than
- <= less than or equal to
- && 1 if both operands true
- || 1 if either operand is true
- ! 1 when operand is false

Misc.

- Spaces don't matter in C.
- Statements must be followed by semicolon ;
- Functions must start and end with { }
- //Comment or /* Comment */
- to compile, gcc -Wall -Werror -lm file.c

Pointers

- Must include type when declaring a pointer.
- Placement of * doesn't matter.
- Example program:


```
int* Ptr1 = NULL; //Pointer of type int
char *Ptr2 = NULL; //Pointer of type char
Ptr1 = &X; //Ptr1 points to variable X
printf("%p", Ptr1); //prints memory location of Ptr
printf("%d", *Ptr1); //prints value Ptr points to
Ptr2 = &Char[3] //Ptr1 points to 3rd character
Ptr2 --; //Ptr1 points to 2nd character now
```
- You can add, subtract, and compare pointers.
- You cannot multiply or divide pointers.
- C uses pass by value for function calls, but pass by reference can be simulated by using pointers.

Opening Files

- To open a file:


```
FILE* FilePointer; //Initialize the "file pointer"
if( FilePointer = fopen( "FileName.txt", "r" ) ) == NULL )
    printf("Cannot Open File"); //Error opening file
else //If the file opens correctly
{
    ...
    fclose(FilePointer); //Close the file
}
```
- File modes (settings used in fopen): r (reading), r+ (read & write), w (write), w+ (write & read), a (write at end of file), a+ (write at end of file & read)

Reading and Writing to Files

- ```
char A = fgetc(FilePointer); //Returns next character in file
putc(char A, FilePointer); //Write a character to a file
puts(String, FilePointer); //Write String to file FilePointer

//Stores string from FilePointer of length StringLength into String
fgets(String, StringLength, FilePointer);

//Store a value of specified format type
fscanf(FilePointer, "%f", Var);

//Write to a file using formatted output
fprintf(FilePointer, "%s", Name);
```