C-Crash Course

What is programming?

Programming is the art of telling a computer to do a tedious action that a human would never want to do.

How do we tell a computer to do something?

Algorithms: An algorithm is an ordered set of unambiguous instructions which must terminate.

How do we eliminate ambiguity?

Syntax: reducing the amount of words we can use.

Semantics: giving all our words very definite meanings.

What is our reduced vocabulary?

The keywords:
for, while, int, float, define, if, else, include, long. . .

C is "main()"ly easy

main is where everything happens. Anything that will compile in your program must happen from main.

```
int main(void) {
   return 0;
}
```

A Basic Program

A program to print something to the screen

```
#include <stdio.h>

int main(void) {
   printf("Hello World");
   return 0;
}
```

#include a way to tell the computer to include a library like: stdio.h (standard output and input)

#printf a function included in stdio.h that prints something to the screen.

Variables:
Most programs require variables which represent values that we can manipulate.

These variables can be of these types: int (integer), float (real), char (character)

We may also use the prefixes **long** or **static** infront of any of these.

What operations are we allowed?

%,/,*,+,- (BraMoD Mulas)

Shorthand Notation

```
x++;    x=x+1;
x--;    x=x-1;
```

There are others but they are just confusing.

How Do We print out variables?

printf("%(dataType),var_name);

int (%d) float (%f)
char (%c) string (%s)

```
int x=7;
printf("Our int is %d.\n",x);
```

\n is a newline.

```
Output:   Our int is 7
          _
```

Types of Variables

local, global, static

Scope and Visibility

```
#include <stdio.h>

#define name anything
#define PI 4.14
#defein OR ||

dataType name;
int globX;

int main (void) {
   data_Type locX;
}
```

Arrays. Its as easy as 0,1,2. . .

Arrays are a group of variables of similar attributes that you would like to keep together.

```
dataType name[]={data};
int x[]={1,2,3,4};
```

You can have **x[0]..x[3]** and you can treat these like regular variables.

C Has Tons of "Function"ality

A function, like in math, will accept some value(s) and return a single value.

f(x,y)=x^2+y^2 f(2,3)=4+9

Lets model this function in C.

```
int f(int x, int y) {
   return x*x+y*y;
}
```

In general we have:

```
data_type name(accept) {
   return name of data_type;
}
```

To invoke (call) a function we do it from the main program like:

f(2,3); or f(var1,var2);

assuming var 1 and var 2 have some value

Procedure? Whats A Procedure?
Procedures are functions which don't return anything.

```
void fun_name (accept) {
   return void;
}
```

In C we have void functions, not procedures.

Notes about functions
You can call a function from any other function. Yes, including itself (recursion).

User Defined Data Types
We are not restricted to int, float, and char.
We can make whatever type we want.

typedef and *enum*

```
enum week {sun, mon, tue, wed,
thu, fri, sat};
```

typedef will now allow us to take this enumerated data type and do something with us.

```
typedef enum week day_type;

int main() {
   day_type day=mon;
}
```

in general

```
enum name {el1,el2,...,eln};
typedef enum name nameType;
```

Now you can use nameType the same way you would use int or float. However we are not given a % to use in printf.

Strucutres

Is a means of making one variable have many qualities.

```
struct emp {
   int emp_num;
   int salary;
   int start_date[3];
}

typedef struct emp empType;

empType employee1;

employee1.emp_num = 1000;
```

We can even have an array of this if we wanted.

```
empType allEmp[10];
```

in which case we could have

```
allEmp[3].start_date[2]=1998
```

READ LAB 6

Strings

A string is an array of characters declared like:

char string[]={“Hello World”};

which gives you an array like this:

|H|e|l|l|o|_|W|o|r|l|d|\0|

\0 is called the null terminator and is a single character to the computer.

ALL CHARACTERS END WITH THE NULL TERMINATOR!

Remember ‘a’ is how you refer to a character.

Control Strucutes

A control structure is characterized by a boolean expression. There are three fundamental control structures in C.

if for while

A boolean expression is something that evaluates to true or false.

equality x==y
inequality >, <, <=, >=

(logical conectors)

or || and &&

```
(x==y)&&(y<5)||(y=0)&&(1)
```

if

```
if(boolean statement) {
   code
} else (boolean statement) {
   code
} else {
   code
}
```

while

```
while(boolean==true) {
   code
}
```

```
while(x>7) {
   x--;
}
```

you must make sure to have something incrementing in your loops

for

```
for(i=0;i<n;i++) {
   code
}
```

Will execute n times and is equivelant to:

```
i=0;
while(i<n){
   i++;
}
```

You should use a for loop when you know how many times you would like to repeat something.

i will have all values between 0..n. Don't be afraid to use it!

hit the "break"s!!!

The break statement will break out of any control structure.

```
while(1==1) {
   if (condition) break;
}
```

Will break you out of the loop.

The ‘p’ Word

A pointer is a variable which holds the memory location of another variable.

dataType *varName; ~or~
dataType* varName;

***varName;** refers to what var name is pointing too

&varName; refers to the numerical adress that varName is pointing too.

int *arrow, realVal=12;

***arrow=&realVal;**

***arrow=6 is the same as realVal=6**

Parents say: "Pointing is rude". They're wrong. . .

Pointing is actually the only way we can pass arrays to functions.

```
int a[]={1,2};
```

where **a** is an address. **&a** is incorrect.

i.e. (id est, by example):

```
int main(void) {
   int x[]={1,2,3};
   proc(x);
   return;
}

void proc(int *a) {
   a[1]=12;
}
```

Passing by Reference or Value

Passing w/out a pointer is pass by reference. A copy is made in the function.

Passing w/ a pointer is pass by value and the value passed will be manipulated.

```
int main(void){
   int x=7;
   proc(&x);
```

- - - - - - - - - - - - - - - - -

```
void proc(int *a) {
   *a=12;
}
```

The Heap as your friend

You know those friends that you ignore till you need them? That's what type of friend the heap is.

The heap will give you some memory during run-time. Usually you have to tell the compiler before hand.

```
data_type *pointer;

pointer=(data_type*)malloc(the lengthYouWant *sizeof(dataType))
```

-Concretely-

```
int *A

A=(int*)malloc(10*sizeof(int));
```

Will make us an array of 10 spots.

Now we can declare array dynamically.

Generalizaed Program:

```
#include <library.h>

#define name anything

dataType globalVars;

dataType funcName(input);
//function protocall

int main(void) {
   return 0;
}

dataType funcName(input){
   return dataType;
}
```