# Introduction to C Programming

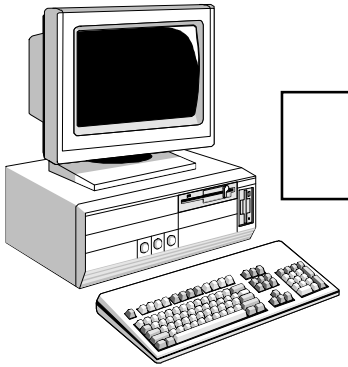
# A Brief History

- Created by Dennis Ritchie at AT&T Labs in 1972
- Originally created to design and support the Unix operating system.
- There are only 27 keywords in the original version of C.
  - for, goto, if, else ……
- Easy to build a compiler for C.
  - Many people have written C compilers
  - C compilers are available for virtually every platform
- In 1983 the American National Standards Institute (ANSI) formed a committee to establish a standard definition.
  - Called ANSI Standard C.
  - As opposed to K&R C (referring to the general “standards” that appeared in the first edition of Brian Kernighan and Ritchie’s influential book: *The C Programming Language*)

# Why use C?

- C is intended as a language for programmers
  - BASIC was for nonprogrammers to program and solve simple problems.
  - C was created, influenced, and field-tested by working programmers.
- C is powerful and efficient
  - You can nearly achieve the efficiency of assembly code.
  - System calls and pointers allow you do most of the things that you can do with an assembly language.
- C is a structured language
  - Code can be written and read much easier.
- C is standardized
  - Your ANSI C program should work with any ANSI C compiler.

# The C Development Cycle



Edit Program

Source Code

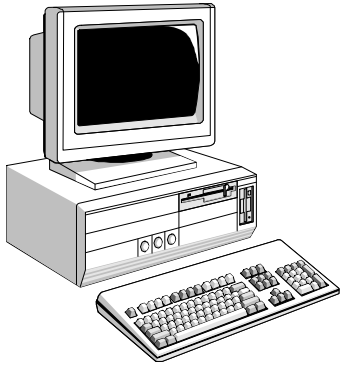
Compile

Object Code

Library Files

Link Object Code

Executable



# “Hello World”

- Everyone writes this program first

```
#include <stdio.h>
int main ( )
{
    printf ("Hello, World!\n");
    return 0;
}
```

# Compilation (1)

- Compilation translates your source code (in the file hello.c) into object code (machine dependent instructions for the particular machine you are on).
  - Note the difference with Java:
    - The javac compiler creates Java byte code from your Java program.
    - The byte code is then executed by a Java virtual machine, so it's machine independent.
- Linking the object code will generate an executable file.
- There are many compilers for C under Unix
  - SUN provides the Workshop C Compiler, which you run with the cc command
  - There is also the freeware GNU compiler gcc

# Compilation (2)

- To compile a program:
  - Compile the program to object code.

    obelix[2] > cc –c hello.c
  - Link the object code to executable file.

    obelix[3] > cc hello.o –o hello
- You can do the two steps together by running:

  obelix[4] > cc hello.c –o hello
- To run your program:

  obelix[5] > ./hello

  Hello World!

If you leave off the -o, executable goes into the file a.out

# Compilation (3)

- Error messages are a little different than you may be used to but they can be quite descriptive.
- Suppose you forgot the semi-colon after the printf

```
obelix[3] > cc hello.c –o hello
"hello.c", line 5: syntax error before or at: return
cc: acomp failed for hello.c
```

- Notice that the compiler flags and informs you about the error at the first inappropriate token.
  - In this case, the return statement.
- Always try to fix problems starting with the first error the compiler gives you - the others may disappear too!

# Example 1

```
#include <stdio.h>
int main ()
{
    int radius, area;

    printf ("Enter radius (i.e. 10) : ");
    scanf ( "%d", &radius);
    area = 3.14159 * radius * radius;
    printf ("\nArea = %d\n\n", area);
    return 0;
}
```

# Example 2

```
#include <stdio.h>
int main ()
{
    int i, j;
    for (i = 0; i < 10; i++)
    {
            printf ("\n");
            for (j = 0; j < i+1; j++ )
                printf ( "A");
    }
    printf("\n");
    return 0;
}
```

# Example 3

```
/* Program to calculate the product of
     two numbers */

#include <stdio.h>

int product(int x, int y);

int main ()

{

   int a,b,c;

   /* Input the first number */

   printf ("Enter a number between 1
              and 100: ");

   scanf  ("%d", &a);


   /* Input the second number */

   printf ("Enter another number
              between 1 and 100: ");

   scanf  ("%d", &b);

   /* Calculate and display the product */

   c = product (a, b);

   printf ("%d times %d = %d \n", a, b, c);

   return 0;

}


/* Functions returns the product of its
     two arguments */

int product (int x, int y)

{

   return (x*y);

}
```