Héctor Márquez García

887430

Programación Avanzada

Luis Ricardo Salgado

 Tarea 1

**1.- Prepocesador C.**

El preprocesador es un programa separado que es invocado por el compilador antes de que comience la traducción real. Puede realizar inclusión de otros archivos dentro del archivo a compilar, definición de constantes simbólicas y macros, compilación condicional del código de un programa y la ejecución condicional de las directivas del preprocesador. Toda directiva del preprocesador comienzan con #.

Las ventajas que tiene usar el preprocesador son:

-Los programas son más fáciles de desarrollar,

-Son más fáciles de leer,

-Son más fáciles de modificar

-El código de C es más transportable entre diferentes arquitecturas de máquinas.

Sintaxis:

**#*define*** *:* La directiva #define crea constantes simbólicas y macros (Operaciones definías como símbolos)

#define nombre\_de\_macro( x ) *macro* – Reemplaza nombre\_de\_macro con el macro especificado

Sintaxis: #define *identificador texto de reemplazo*

Ejemplo: #define PI 3.1416

En toda parte del código que aparezca PI será reemplazado por 3.1416

El uso de la sustitución de macros en el lugar de las funciones reales tiene un beneficio importante, ya que incrementa la velocidad del código porque no se penaliza con una llamada de función.

***#include* :** Esta directiva provoca la inclusión de una copia del archivo especificado en lugar de la directiva. Inlcuye encabezados de la biblioteca estándar como stdio.h.

#include <archivo>: Búsuqeda a través de diretorios preestablecidos.

#include "archivo": Busca el archivo dentro del mismo directorio.

***#if:***Las directivas condicionales permiten controlar la ejecución de las directivas del preprocesador y la compilación del código de un programa. La directiva #if evalua una expresión entera constante. Siempre se debe terminar con #endif para delimitar el fin de esta sentencia.

Se pueden así mismo evaluar otro código en caso se cumpla otra condición, o bien, cuando no se cumple ninguna usando #elif o #else. Por ejemplo,

La directiva #if también sirve para comentar código para evitar la compilación del mismo.

*#****ifdef*** *:* Sirve para saber si está definida una macro. Es una abreviatura de #if defined

Sintaxis:

#ifdef “nombre”

***#ifndef :***Sirve para saber si no está definida una macro. Es una abreviatura de #if !defined

Sintaxis:

#ifndef “nombre”

***#pragma****:* Directiva definida por el estándar como dependiente de la implementación. Se usa para pasarle instrucciones al compilador. Estas instrucciones dependen de dicho compilador, y el estándar no define ninguna.
Ejemplo:

#pragma inline: Directiva que definen los compiladores TURBO C para indicar que se va a usar ensamblador en línea dentro del código fuente C.

**2.- Definición de variables en C**

1. Para definir una variable en C, se utiliza el formato *tipo de variable, variable.* Este debe comenzar con una letra o un guión bajo y no debe ser una palabra reservada. Ejemplo

Tipo de variable = int

Nombre de variable var\_1

Palabras reservadas: **auto, double, int, struct, break, else, long, switch, case, enum, register, typedef,**  **char, extern, return, union,** **const, float, short, unsighed,** **continue, for, signed, void,**  **default, goto, sizeof, volatile, do, if, static, while.**

**3.-Impresión**

**a. Especificadores de conversión**

|  |  |
| --- | --- |
| d | Despliega un entero decimal con signo. |
| i | Despliega un entero decimal con signo. |
| o | Despliega un entero octal sin signo. |
| u | Despliega un entero decimal sin signo. |
| x | Despliega un entero hexadecimal sin signo |
| h o l | Se coloca antes de cualquier especificador de conversión entero para indicar que se despliega un entero corto o largo. (modificadores de longitud) |
| e | Despliega un valor de punto flotante con notación exponencial. |
| f | Despliega un valor de punto flotante con notación de punto fijo. |
| g | Despliega un valor de punto flotante con el formato flotante f, o con el formato exponencial e basado en la magnitud del valor. |
| L | Se coloca antes del especificador de conversión para indicar que se desplegara un valor de punto flotante long double. |
| p | Despliega un valor apuntador de manera definida por la implementación. |
| n | Almacena el número de caracteres ya desplegados en la instrucción printf actual. Proporciona un apuntador a un entero como el argumento correspondiente. No despliega valor alguno. |
| % | Despliega el carácter de porcentaje. |

**b. Secuencias de escape**

|  |  |
| --- | --- |
| \’ | Despliega el carácter de comilla sencilla. |
| \” | Despliega el carácter de comilla doble. |
| \? | Despliega el carácter de signo de interrogación. |
| \\ | Despliega el carácter de diagonal invertida. |
| \a | Provoca una alerta sonora o una alerta visual. |
| \b | Mueve el cursor una posición hacia atrás en la línea actual. |
| \f | Mueve el cursor al inicio de la siguiente página lógica. |
| \n | Mueve el cursor al principio de la siguiente línea. |
| \r | Mueve el cursor al principio de la línea actual. |
| \t | Mueve el cursor a la siguiente posición del tabulador horizontal. |
| \v | Mueve el cursor a la siguiente posición del tabulador vertical. |

**c.**

Alineación entero: Se altera su ancho de campo. Si el ancho de campo es mayor que el dato a imprimir, el dato normalmente se justifica a la derecha dentro del campo. Si el campo de ancho es negativo, se justifica la salida a la izquierda dentro del campo. Para alinearlo a la derecha se pone %#d, donde # es el número de caracteres del margen a la derecha

**d.**

 Para especificar la precisión en la parte entera se utiliza el formato printf(“%\*.\*f”, #,#,#). Cada asterisco toma el valor respectivo en la lista de argumentos. El primer asterisco indica el tamaño del campo y el segundo indica la precisión en el campo decimal.

**4.Sintaxis rutinas**

**Scanf(…) :** Viene del encabezado stdio.h de la biblioteca estándar.

Sirve para leer una entrada del teclado.

scanf(“%tipo de variable”, variable);

**getchar():**Viene del encabezado stdio.h de la biblioteca estándar

Lee un character de standard input.

Se invoca getc() y regresa un carácter.

Para indicar error en la lectura getc() regresa EOF (end of file).

**getc():**Viene del encabezado stdio.h de la biblioteca estándar

Lee un character de un stream.

Regresa el carácter leído.

Para marcar error regresa EOF.

Sintaxis:

FILE\* fp;

..

..

ch = getc(fp))

 **Fuentes**

“Como programar en C/C++ y Java” – Deitel – Cuarta Edición

<http://www.fismat.umich.mx/mn1/manual/node13.html>

<http://www.elrincondelprogramador.com/default.asp?pag=articulos%2Fleer.asp&id=16>

<http://msdn.microsoft.com/es-es/library/5231d02a.aspx>

<http://msdn.microsoft.com/es-es/library/x198c66c.aspx>

**5.- PROGRAMAS**

**A)**

/\*Ejercicio A

 \*Programa que convierte

 \*grados F a celcius

\*/

#include <stdio.h>

void main() {

 double grados = 150;

 grados = (grados -32)/1.8;

 printf("%.2f\n", grados);

}

**B)**

/\*Ejercicio B

 \*Programa que calcula

 \*radio de una esfera

\*/

#include <stdio.h>

void main(){

 double vol;

 double radio = 30;

 vol = (4.0/3.0)\*(3.1415f)\*(radio \* radio \* radio);

 printf("%.2f\n", vol);

}

**C)**

/\*Ejercicio C

 \*Programa que convierte

 \*decimal a binario

\*/

#include <stdio.h>

void convBinario(int num, char \*x) {

 for(; num > 0; ) {

 \*x = ((num % 2) + 48);

 num = num>>1;

 x--;

 }

}

void main() {

 int num;

 char x[] = "00000000";

 x[8] = 0;

 printf("Ingresa un numero para convertirlo a binario\n");

 scanf("%d", &num);

 convBinario(num, &x[7]); //ando ultima direccion

 printf("Tu numero en dec es %s\n", x);

}

**D)**

/\*Ejercicio D

 \*Programa que convierte

 \*un numero de 0 a 1 millon

 \*de digitos a letra

\*/

#include <stdio.h>

void toCharArray(int n, char \*num) {

 for(; n > 0; n/=10, num--) {

 \*num = (n % 10) + 48;

 }

}

int main() {

 int num;

 int i;

 int decMil = 0;

 int centMil = 0;

 char numero[] = "0000000";

 printf("Ingresa numero : \n");

 scanf("%d", &num);

 if(num == 0) {

 printf("Cero\n");

 return(0);

 }

 toCharArray(num, &numero[6]);

 if(numero[1] != '0') {

 centMil = 1;

 }

 if(numero[2] != '0') {

 decMil = 1;

 }

 for(i = 0; i < 7 ; i++) {

 switch(i) {

 case(6) : //Unidades

 switch(numero[i]) {

 case '0':

 break;

 case '1':

 printf("uno");

 break;

 case '2':printf("dos");

 break;

 case '3':printf("tres");

 break;

 case '4':printf("cuatro");

 break;

 case '5':printf(" cinco");

 break;

 case '6':printf(" seis");

 break;

 case '7':printf(" siete");

 break;

 case '8':printf(" ocho");

 break;

 case '9':printf(" nueve");

 break;

 }

 break;

 case(5) : //Decenas

 switch(numero[i]) {

 case '1':

 if((numero[6] > 48) && (numero[6] < 54)) {

 switch(numero[6]) {

 case '1':printf(" once");

 break;

 case '2':printf(" doce");

 break;

 case '3':printf(" trece");

 break;

 case '4':printf(" catorce");

 break;

 case '5':printf(" quince");

 break;

 }

 numero[6] = '0';

 }

 else if((num % 10) > 0) {

 printf(" dieci");

 }

 else {

 printf(" diez");

 }

 break;

 case '2':

 if(num % 10 == 0) {

 printf(" veinte");

 }

 else {

 switch(numero[6]) {

 case '1': printf(" ventiuno");

 break;

 case '2': printf(" ventidos");

 break;

 case '3': printf(" ventitres");

 break;

 case '4': printf(" venticuatro");

 break;

 case '5': printf(" venticinco");

 break;

 case '6': printf(" ventiseis");

 break;

 case '7': printf(" ventisiete");

 break;

 case '8': printf(" ventiocho");

 break;

 case '9': printf(" ventinueve");

 break;

 }

 numero[6] = '0';

 }

 break;

 case '3':

 printf(" treinta");

 if((num % 10) > 0) {

 printf(" y");

 }

 break;

 case '4':

 printf(" cuarenta");

 if((num % 10) > 0) {

 printf(" y");

 }

 break;

 case '5':

 printf(" cincuenta");

 if((num % 10) > 0) {

 printf(" y");

 }

 break;

 case '6':

 printf(" sesenta");

 if((num % 10) > 0) {

 printf(" y");

 }

 break;

 case '7':

 printf(" setenta");

 if((num % 10) > 0) {

 printf(" y");

 }

 break;

 case '8':

 printf(" ochenta");

 if((num % 10) > 0) {

 printf(" y");

 }

 break;

 case '9':

 printf(" noventa");

 if((num % 10) > 0) {

 printf(" y");

 }

 break;

 }

 break;

 case(4) : //Centenas

 switch(numero[i]) {

 case '1':

 if((num % 100) > 0) {

 printf(" ciento");

 }

 else {

 printf(" cien");

 }

 break;

 case '2':printf(" doscientos");

 break;

 case '3':printf(" trescientos");

 break;

 case '4':printf(" cuatrocientos");

 break;

 case '5':printf(" quinientos");

 break;

 case '6':printf(" seiscientos");

 break;

 case '7':printf(" setecientos");

 break;

 case '8':printf(" ochocientos");

 break;

 case '9':printf(" novecientos");

 break;

 }

 break;

 case(3) : //miles

 switch(numero[i]) {

 case '0':

 if(decMil == 1) {

 printf(" mil");

 //printf("Existen decenas de miles");

 }

 else if(centMil == 1) {

 // printf("Existe cientos de miles");

 if(numero[3] == '1') {

 printf(" un mil");

 }

 else {

 //printf("Solo cientos de miles");

 printf(" mil");

 }

 }

 break;

 case '1':

 if((centMil == 0) && (decMil == 0)) {

 printf("mil");

 }

 else {

 printf(" un mil");

 }

 break;

 case '2':printf(" dos mil");

 break;

 case '3':printf(" tres mil");

 break;

 case '4':printf(" cuatro mil");

 break;

 case '5':printf(" cinco mil");

 break;

 case '6':printf(" seis mil");

 break;

 case '7':printf(" siete mil");

 break;

 case '8':printf(" ocho mil");

 break;

 case '9':printf(" nueve mil");

 break;

 }

 break;

 case(2) : //decenas miles

 switch(numero[i]) {

 case '1':

 if((numero[3] > 48) && (numero[3] < 54)) {

 switch(numero[3]) {

 case '1':printf(" once");

 break;

 case '2':printf(" doce");

 break;

 case '3':printf(" trece");

 break;

 case '4':printf(" catorce");

 break;

 case '5':printf(" quince");

 break;

 }

 numero[3] = '0';

 }

 else if(numero[3] != '0') {

 printf(" dieci");

 }

 else {

 printf(" diez");

 }

 break;

 case '2':

 if(numero[3] == '0') {

 printf(" veinte");

 }

 else {

 switch(numero[3]) {

 case '1': printf(" ventiun");

 break;

 case '2': printf(" ventidos");

 break;

 case '3': printf(" ventitres");

 break;

 case '4': printf(" venticuatro");

 break;

 case '5': printf(" venticinco");

 break;

 case '6': printf(" ventiseis");

 break;

 case '7': printf(" ventisiete");

 break;

 case '8': printf(" ventiocho");

 break;

 case '9': printf(" ventinueve");

 break;

 }

 numero[3] = '0';

 }

 break;

 case '3':

 printf(" treinta");

 if(numero[3] != 48) {

 printf(" y");

 }

 break;

 case '4':

 printf(" cuarenta");

 if(numero[3] != 48) {

 printf(" y");

 }

 break;

 case '5':

 printf(" cincuenta");

 if(numero[3] != 48) {

 printf(" y");

 }

 break;

 case '6':

 printf(" sesenta");

 if(numero[3] != 48) {

 printf(" y");

 }

 break;

 case '7':

 printf(" setenta");

 if(numero[3] != 48) {

 printf(" y");

 }

 break;

 case '8':

 printf(" ochenta");

 if(numero[3] != 48) {

 printf(" y");

 }

 break;

 case '9':

 printf(" noventa");

 if(numero[3] != 48) {

 printf(" y");

 }

 break;

 }

 break;

 case(1) : //cientos

 switch(numero[i]) {

 case '1':

 if((num % 10000) > 0) {

 printf(" ciento");

 }

 else {

 printf(" cien");

 }

 break;

 case '2':printf(" doscientos");

 break;

 case '3':printf(" trescientos");

 break;

 case '4':printf(" cuatrocientos");

 break;

 case '5':printf(" quinientos");

 break;

 case '6':printf(" seiscientos");

 break;

 case '7':printf(" setecientos");

 break;

 case '8':printf(" ochocientos");

 break;

 case '9':printf(" novecientos");

 break;

 }

 break;

 case(0) : //millon

 if(numero[0] == '1') {

 printf("Un millon");

 }

 break;

 }

 }

 printf("\n");

 return(1);

}

**E)**

/\*Ejercicio E

 \*Programa que calcula el dia de la semana

 \*Solo acepta fechas validas

 \*A partir del mes 2 anio 0

 \*Cualquier anio

\*/

#include <stdio.h>

void main() {

 int acum = 0;

 int bisiesto = 1;

 int dia;

 int mes;

 int ano;

 int anoActual = 0;

 int diaActual = 1;

 int mesActual = 1;

 int m[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

 int res;

 int tmp = 0;

 printf("Ingresa dia : ");

 scanf("%d", &dia);

 printf("Ingresa mes : ");

 scanf("%d", &mes);

 printf("Ingresa ano : ");

 scanf("%d", &ano);

 if(anoActual <= ano) {

 while(1) {

 if(anoActual <= ano) { //Mientras alcanzo al mes que me piden

 if((bisiesto == 0) && (mesActual == 2)) {

 diaActual += 29 ;

 acum += 29;

 tmp = (acum-1)%7;

 }

 else {

 diaActual += m[mesActual-1];

 acum += m[mesActual-1];

 }

 mesActual += 1;

 diaActual = 1;

 if(mesActual == 13) {

 anoActual += 1;

 bisiesto = (anoActual - 2008) % 4;

 mesActual = 1;

 }

 }

 if ((mesActual == mes) && (anoActual == ano)) { //Mientras alcanzo al mes que me piden

 acum += (dia - diaActual);

 break;

 }

 }

 }

 res = (acum-1)%7;

 printf("Dias = %d\n", acum);

 switch(res) {

 case 0: printf("Domingo\n");

 break;

 case 1:printf("Lunes\n");

 break;

 case 2:printf("Martes\n");

 break;

 case 3:printf("Miercoles\n");

 break;

 case 4:printf("JUeves\n");

 break;

 case 5:printf("Viernes\n");

 break;

 case 6:printf("Sabado\n");

 break;

 }

}

**F)**

/\*Programa que calcula serie de taylor

 \*de un coseno hasta 20 repeticiones

 \*

\*/

#include <stdio.h>

/\*Metodo para guardar los factores

 \*de un factorial muy grande

 \*

\*/

void llenaArrFact(unsigned int \*arr, unsigned int num) {

 int x = 0;

 unsigned int i = num;

 unsigned int acum = 1;

 unsigned int max = 4294967295;

 for(i = num; i >= 1; i--, x++) {

 acum \*= i;

 if(x == 4) { //Si los factoriales son menores a 2^32 los guardo en el arreglo

 \*arr = acum;

 acum = 1;

 arr++;

 x = 0;

 }

 }

 \*arr = acum; //Si no alcanzo a guardarla por que el numero era muy pequeno, guarda aqui

}

/\*Metodo llenaArrExp

 \*Recibe arreglo a llenar

 \*Recibe potencia

 \*Recibe numero a elevar

 \*Llena un arreglo de una potencia de num^exp por partes.

\*/

void llenaArrExp(unsigned int \*arr, unsigned int exp, int num) {

 int cont = 1;

 unsigned int acum = 1;

 int i;

 for(i = exp; i > 0; i--, cont++ ) {

 acum \*= num;

 if(cont == 4) {

 cont = 0;

 \*arr = acum;

 acum = 1;

 arr++;

 }

 }

 \*arr = acum;

}

void arrEnUnos(unsigned int \*arr) {

 int i = 0;

 for(;i < 12; i++, arr++) {

 \*arr = 1;

 }

}

/\*Divide Arreglos

 \*Divide 2 arreglos

 \*Permite hacer divisiones con numeros gigantes

\*/

double divideArreglos(unsigned int \*numerador, unsigned int \*denominador) {

 double resultado = 1;

 int i;

 for(i = 0; i < 12; i++) {

 resultado \*= ((numerador[i] \* 1.0) / (denominador[i]\*1.0) );

 }

 return resultado;

}

void main(void) {

 int i;

 double operacion;

 double div;

 unsigned int x;

 unsigned int den[] = {1,1,1,1,1,1,1,1,1,1,1,1};

 unsigned int num[] = {1,1,1,1,1,1,1,1,1,1,1,1};

 printf("Ingresa angulo \n");

 scanf("%d", &x);

 for(i = 0; i <= 19; i++) {

 llenaArrExp(&num[0], (i\*2), x); //arreglo, elevado a, numero a elevar

 llenaArrFact(&den[0], (i\*2)); // matriz, factorial de

 div = divideArreglos(&num[0], &den[0]);

 if( i == 0 ) {

 operacion = 1;

 printf("1");

 }

 else if( (i % 2) > 0) {

 operacion -= div;

 printf(" -%f\n", div);

 }

 else if( (i % 2) == 0 ) {

 operacion += div;

 printf(" +%f", div);

 }

 arrEnUnos(&num[0]);

 arrEnUnos(&den[0]);

 }

 printf("\n Operacion = %f\n", operacion);

}