2021

Informe Trabajo Practico final

PROFESORA: MELINA SANCHEZ

ALUMNO: FRANCISCO IGNACIO MUZIO

COMISION 4

Pipeline 1: Creación de los componentes.

Funcion 1: Carga de arreglo en base a archivo

En primer lugar, el tp nos pedía cargar un arreglo de tipo PaqueteMateriaPrima desde un archivo.bin, por lo que, llamamamos a la siguiente función en el main.

```
int main()
{
    PaqueteMateriaPrima a[MAX];
    Componente b [MAX];
    int validos=CargarArr(a);

int CargarArr (PaqueteMateriaPrima m[])
{
    FILE *arch=fopen("Paquetes.bin","rb");
    PaqueteMateriaPrima p;
    int i=0;
    if(arch!=NULL)
    {
        while(fread(&p,sizeof(PaqueteMateriaPrima),1,arch)>0)
        {
            m[i]=p;
            i++;
        }
        fclose(arch);
    }
    return i;
}
```

Donde recibimos el arreglo por parámetro, luego dentro de la función, abrimos el archivo, y mientras lo vamos leyendo, lo vamos cargando en una variable de tipo PaqueteMateriaPrima y luego lo cargamos en el arreglo e incrementamos i en 1 para avanzar dentro de ese arreglo y no estar sobrescribiendo siempre la misma casilla. Cuando termina de leer el archivo y, por lo tanto, de cargar el arreglo, nos devuelve i que es igual a validos en el main.

Funcion 2: Creación de componentes en base a arreglo PaqueteMateriaPrima

En segundo lugar, nos pide que, con ese arreglo cargado con paquetes de distinta cantidad de materia prima, calidad (‘a’,’b’ o ‘c’) y costo. Creemos componentes que requieren determinadas cantidades de materia de cada paquetet, adoptando así, la calidad de la materia prima en ellos y mediante un calculo de costo por grama y costo de producción su costo final. Las únicas condiciones son que la producción debe ser siempre en orden, es decir, Muñeco luego Copa de Vino y luego Biblia y que una vez que la cantidad de materia prima del paquete no sea suficiente para fabricar el siguiente componente, la materia que quede dentro de ese paquete será descartada y se pasara al siguiente paquete. Por lo que llamamos a esta función en el main.

```
int valComp=Fabricar(a,validos,b);
```

```
int Fabricar (PaqueteMateriaPrima m[],int val, Componente c[])
{
    float g;
    int i=0;
    int j=0;
    int flag=0;
    g=m[i].cantMateria/m[i].costo;
    while(flag==0)
    {

        if(m[i].cantMateria<DOLL && i<val-1)
        {
            i++;
            g=m[i].cantMateria/m[i].costo;
        }
        if(m[i].cantMateria>=DOLL && flag==0)
        {
        m[i].cantMateria=(m[i].cantMateria)-25;
        c[j].calidad=m[i].calidad;
        c[j].falla=rand()%3;
        c[j].costo=g*25+6;
        strcpy(c[j].nombre,"Doll");
        j++;
        }
        else if (flag==0)
        {
            flag=1;
        }


        if(m[i].cantMateria<COPA && i<val-1)
        {
            i++;
            g=m[i].cantMateria/m[i].costo;
        }
        if(m[i].cantMateria>=COPA && flag==0)
        {
        m[i].cantMateria=(m[i].cantMateria)-3;
        c[j].calidad=m[i].calidad;
        c[j].falla=rand()%3;
        c[j].costo=g*3+0.75;
        strcpy(c[j].nombre,"Copa de vino");
        j++;
        }
        else if (flag==0)
        {
            flag=1;
        }
```

```
        if (m[i].cantMateria<BIBLIA && i<val-1)
        {
            i++;
            g=m[i].cantMateria/m[i].costo;
        }
        if(m[i].cantMateria>=COPA && flag==0)
        {
        m[i].cantMateria=(m[i].cantMateria)-2;
        c[j].calidad=m[i].calidad;
        c[j].falla=rand()%3;
        c[j].costo=g*2+0.5;
        strcpy(c[j].nombre,"Biblia");
        j++;
        }
        else if (flag==0)
        {
            flag=1;
        }

    }
    return j;
```

Lo que hacemos en esta función es primero, sacar el costo por gramo del primer paquete, luego chequeamos si la cantidad de materia dentro del paquete es suficiente, si lo es, no entra el primer if y crea el componente y lo carga en un arreglo de tipo componente, en caso de que no, avanza en el arreglo al siguiente paquete y vuelve a calcular su costo por gramo. En esta función tuve un problema, donde había puesto que el while tenga la condición de i<val y no tenia el i if(m[i].cantMateria>=COPA && flag==0) y el else if (flag==0) por lo que al llegar al último paquete, con la condición de que i<val-1, no incrementaba más i por lo que nunca rompía el while y hacia un bucle infinito creando así infinitos componentes. Al usar un flag, lo que nos permite es que si llegamos al ultimo paquete y su contenido ya no alcanza para fabricar el siguiente componente activa el flag haciendo así que no pueda fabricar tampoco los otros componentes ya que debemos descartar lo que sobra del paquete y también así cortar el while. Luego la función nos retorna j, que es igual a los validos del arreglo componente y nos va a servir para la siguiente función.

Funcion 3: Cargar archivo con arreglo componentes

Por último, en el primer pipeline nos pedía cargar el archivo componente generado dentro de la función anterior en un nuevo archivo.bin llamado “Componentes”. Llamamos a la siguiente función:

```
CargarArch(b,valComp);
```

```
void CargarArch(Componente c[],int val)
{
    FILE *arch=fopen("Componentes.bin","wb");
    int i=0;
    if(arch!=NULL)
    {
        while(i<val)
        {
            fwrite(&c[i],sizeof(Componente),1,arch);
            i++;
        }
        fclose(arch);
    }
}
```

Donde recibimos por parámetro tanto el arreglo como los validos y a medida que avanzamos en este aumentando i hasta llegar a validos vamos escribiendo en el archivo.

```
int main()
{
    PaqueteMateriaPrima a[MAX];
    Componente b [MAX];
    int validos=CargarArr(a);
    int valComp=Fabricar(a,validos,b);
    CargarArch(b,valComp);

    return 0;
}
```

Quedando así el main de nuestro primer pipeline.

Pipeline 2: Ensamblaje de componentes.

Funcion 1: Carga de arreglo en base a archivo

El segundo pipeline empieza igual donde debemos cargar un arreglo en base a un archivo de tipo componentes. Pero en este nos piden que, en base a los componentes, descartemos los que tienen 2 fallas y que separemos a los componentes por tipo. Por lo que, lo primero que hago es filtrar mientras leo el archivo, cuales componentes tienen 2 fallas y cuales tienen menos, cargando asi un arreglo a general o cargando asi un arreglo de tipo float perdidas como nos pide la consigna.

```
int main()
{
    Componente a[MAX];
    Componente b[MAX];
    Componente c[MAX];
    Componente d[MAX];
    float Fallas[MAX];
    PlayMobil e[MAX];
    int valFal=0;
    int validos;
    int validosB=0;
    int validosC=0;
    int validosD=0;
    int validosE;
    validos= CargarA(a,Fallas,&valFal);
```

```
int CargarA(Componente x[],float Fal[],int *valF)
{
    FILE* arch=fopen("Componentes.bin","rb");
    Componente t;
    int i=0;
    if (arch!=NULL)
    {
        while (fread(&t,sizeof(Componente),1,arch)>0)
        {
            if(t.falla<2)
            {
                x[i]=t;
                i++;
            }
            else
            {
                Fal[*valF]=t.costo;
                (*valF)++;
            }
        }
        fclose(arch);
        return i;
    }
}
```

Recibimos por parámetro el arreglo de componentes a, el arreglo tipo float fallas y con un puntero los validos de este último. Luego mientras filtra aumenta i o los validos de falla según el componente y luego retorna i que es igual a los validos del arreglo a.

Función 2: Carga de arreglos según tipo de componente

Como nos piden separar en 3 arreglos lo que recibimos por archivo hacemos lo siguiente:

```
CargarComponentes(a,b,c,d,validos,&validosB,&validosC,&validosD);
```

```
void CargarComponentes (Componente Aa[],Componente Ab[],Componente Ac[],
{
    CargarB(Aa,Ab,val,valb);
    CargarC(Aa,Ac,val,valc);
    CargarD(Aa,Ad,val,vald);
}
```

Generalizamos una función para cada tipo de dato.

```
void CargarB (Componente Aa[],Componente Ab[],int val, int *valb)
{
    for(int i=0; i<val; i++)
    {
        if (strcmp(Aa[i].nombre,"Biblia")==0)
        {
            Ab[*valb]=Aa[i];
            (*valb)++;
        }
    }
}
```

Donde cada vez que llamamos a la función de cada tipo de componente, recorremos el arreglo a y en caso de coincidir este tipo aumentamos los validos del arreglo del tipo y lo cargamos en el arreglo del mismo.

Función 3: Ensamble

Acá nos piden que ensamblemos los muñecos con piezas que sean de la misma calidad y si queda alguna suelta no pasa nada porque se utilizan con otro lote. Por lo que hacemos la siguiente función

```
validosE=Ensamble(b,c,d,e,validos,validosB,validosC,validosD);
```

```
int Ensamble (Componente Ab[],Componente Ac[],Componente Ad[],PlayMobil Ae[],
{
    float costComp;
    float costEn;
    int j=0;
    int y=0;
    for (int i=0; i<valb; i++)
    {
        j=0;
        while(Ac[j].calidad!=Ab[i].calidad && Ac[j].falla<2 && j<valc)
        {
            j++;
        }
        if (j<valc)
        {
            costComp=Ab[i].costo+Ac[j].costo;
            Ac[j].falla=2;
            j=0;
            while(Ad[j].calidad!=Ab[i].calidad && Ad[j].falla<2 && j<vald)
            {
                j++;
            }

             if (j<vald)
             {
                 costComp=costComp+Ad[j].costo;
                 Ad[j].falla=2;
                 if(Ab[i].calidad=='a')
                 {
                     costEn=costComp+14;
                     Ae[y].calidad='a';
                     Ae[y].costo=costEn;
                     y++;
                 }
                 else if (Ab[i].calidad=='b')
                 {
                     costEn=costComp+12;
                     Ae[y].calidad='b';
                     Ae[y].costo=costEn;
                     y++;
                 }
                 else
                 {
                     costEn=costComp+10;
                     Ae[y].calidad='c';
                     Ae[y].costo=costEn;
                     y++;
                 }

return y;
```

En esta función lo que hacemos es utilizar un for para los validos de cualquiera de los tres productos, dado que no importa cual sea ya que con que se acabe uno ya no se puede continuar con la fabricación. En mi caso la biblia. Luego comparamos la calidad de ese componente con el de otro, en mi caso la copa. Si no encontramos ninguno seguimos avanzando en el arreglo copa, si llegamos al final del arreglo y no encontramos uno con la misma calidad avanzamos en biblia. En caso de encontrarlo, cambiamos las fallas de la copa a 2 para que cuando volvamos a recorrer el archivo no lo volvamos a utilizar, calculamos el costo parcial sumando el costo de ambos componentes y definimos la calidad del muñeco, hacemos el mismo proceso con el ultimo componente, si no hay muñeco con la misma calidad que biblia se descarta todo y se avanza en biblia, en caso de que si lo haya se suma nuevamente al costo parcial anterior, el costo del muñeco y el plus del ensamblaje dependiendo de la calidad, luego se carga al arreglo de tipo Playmobil y se aumenta y, que luego se retorna para tener el valor de validos del arreglo Playmobil.

Funciones 4 y 5: Carga Archivo ProductoFinal y Archivo perdidas

Aca nos piden por último, cargar nuestro arreglo Playmobil en un archivo.bin tipo Playmobil llamado “ProductosFInales” y cargar el arreglo perdidas en un archivo.bin de tipo float llamado “Perdidas”.

```
CargaFinales(e,validosE);
CargaPerdidas(Fallas,valFal);
```

```
void CargaFinales (PlayMobil e[],int val)
{
    FILE* arch= fopen("ProductosFinales.bin","wb");
    int i=0;
    if(arch!=NULL)
    {
        while(i<val)
        {
            fwrite(&e[i],sizeof(PlayMobil),1,arch);
            i++;
        }
        fclose(arch);
    }
}
```

```
void CargaPerdidas (float f[],int val)
{
    FILE* arch= fopen("Perdidas.bin","wb");
    int i=0;
    if(arch!=NULL)
    {
        while(i<val)
        {
            fwrite(&f[i],sizeof(float),1,arch);
            i++;
        }
        fclose(arch);
    }
}
```

Quedándonos el main del pipeline 2 de la siguiente manera:

```
int main()
{
    Componente a[MAX];
    Componente b[MAX];
    Componente c[MAX];
    Componente d[MAX];
    float Fallas[MAX];
    PlayMobil e[MAX];
    int valFal=0;
    int validos;
    int validosB=0;
    int validosC=0;
    int validosD=0;
    int validosE;
    validos= CargarA(a,Fallas,&valFal);
    CargarComponentes(a,b,c,d,validos,&validosB,&validosC,&validosD);
    validosE=Ensamble(b,c,d,e,validos,validosB,validosC,validosD);
    CargaFinales(e,validosE);
    CargaPerdidas(Fallas,valFal);

    return 0;
}
```

Pipeline 3: Venta.

Funcion 1: Cargar arreglo en base a archivo

En este pipeline debemos calcular la contaduría que consta de ganancia bruta, costo del lote, costo perdidas y cantidad total de playmobil armados. Por otro lado, cargar en un arreglo de tipo detalleLote, el precio final y cantidad de cada calidad. Primero cargamos el archivo de productos finales en un arreglo.

```
int main()
{
    PlayMobil e[MAX];
    DetalleLote d[CAL];
    Contaduria c;
    int val=0;
    float prom;
    CargarArr (e,&val);
```

```c
void CargarArr (PlayMobil a[],int *val)
{
    FILE *arch=fopen("ProductosFinales.bin","rb");
    PlayMobil f;
    if(arch!=NULL)
    {
        while(fread(&f,sizeof(PlayMobil),1,arch)>0)
        {
            a[*val]=f;
            (*val)++;
        }
        fclose(arch);
    }
}
```

Funcion 2: Cargar contaduría parte 1

Aca lo que hacemos es recorrer el archivo para calcular el costo total del lote y la cantidad muñecos armados, aprovechamos también y cargamos el coste de perdida.

```c
prom=CargCont1 (&c,e,val);
```

```c
float CargCont1(Contaduria* a, PlayMobil b[],int val)
{
    int i=0;
    float j;
    (*a).costoLote=0;
    (*a).cantTotal=0;
    (*a).costoPerdidas=0;
    while(i<val)
    {
        ((*a).cantTotal)++;
        (*a).costoLote=(*a).costoLote+(b[i].costo);
        i++;
    }
    FILE *arch=fopen("Perdidas.bin","rb");
    if (arch!=NULL)
    {
        while(fread(&j,sizeof(float),1,arch)>0)
        {
            (*a).costoPerdidas=(*a).costoPerdidas+j;
        }
        fclose(arch);
    }
    j=(*a).costoLote/(*a).cantTotal;
    return j;
}
```

Por último, en la función, calculamos el promedio de precio para luego utilizarlo en el precio final.

Funcion 3: Cargar DetalleLote

Acá nos piden el precio final unitario de cada calidad que varía según la calidad, costo promedio + 125% arriba si es a, 123% si es b y 120% si es c. También cargamos que cantidad tenemos en cada calidad.

```
c.gananciaBruta=CargDet(d,e,val,prom);

float CargDet(DetalleLote a[],PlayMobil b[],int val,float prom)
{
    int i=0;
    float total=0;
    a[0].precioFinal=prom*2.25;
    a[0].calidad='a';
    a[0].cant=0;
    a[1].precioFinal=prom*2.23;
    a[1].calidad='b';
    a[1].cant=0;
    a[2].precioFinal=prom*2.20;
    a[2].calidad='c';
    a[2].cant=0;
    while (i<val)
    {
        if(b[i].calidad=='a')
        {
            (a[0].cant)++;
        }
        else if (b[i].calidad=='b')
        {
            (a[1].cant)++;
        }
        else
        {
           (a[2].cant)++;
        }
        i++;
    }
    for(int j=0;j<CAL;j++)
    {
        total=a[j].cant*a[j].precioFinal+total;
    }
    return total;
}
```

Aprovechamos que sacamos el precio final y cantidad de cada calidad para calcular la ganancia bruta.

Función 4: Ganancia

Luego de calcular todo, nos piden realizar una función que calcule si el lote tuvo ganancia o pérdida, comparando la ganancia bruta con la suma del coste de lote y el coste de perdida. Hacemos lo siguiente:

```
Ganancia(c);
```

```
void Ganancia (Contaduria c)
{
    float costTot;
    costTot=c.costoLote+c.costoPerdidas;
    if(c.gananciaBruta>costTot)
    {
        printf("\nEl lote consiguio ganancias por sobre el costo.\n");
    }
    else
    {
        printf("\nEl lote genero perdidas por coste.\n");
    }
}
```

Funciones 5 y 6: Cargar archivo de Contaduria y archivo de DetalleLote

Por ultimo y para finalizar el tp, nos piden cargar la variable contaduría en un único registro dentro de un archivo del mismo nombre. Y también cargar el arreglo de DetalleLote en otro archivo también con su mismo nombre. Hacemos esto:

```
CargarCon2(c);
CargarDet(d);
```

```
void CargarCon2 (Contaduria c)
{
    FILE *arch=fopen("Contaduria.bin","wb");
    if (arch!=NULL)
    {
        fwrite(&c,sizeof(Contaduria),1,arch);
        fclose(arch);
    }
}

void CargarDet (DetalleLote d[])
{
    FILE *arch= fopen("DetalleLote.bin","wb");
    if(arch!=NULL)
    {
        for (int i=0;i<CAL;i++)
        {
            fwrite(&d[i],sizeof(DetalleLote),1,arch);
        }
        fclose(arch);
    }
}
```

Quedándonos de esta manera el main del pipeline 3.

```
int main()
{
    PlayMobil e[MAX];
    DetalleLote d[CAL];
    Contaduria c;
    int val=0;
    float prom;
    CargarArr (e,&val);
    prom=CargCont1 (&c,e,val);
    c.gananciaBruta=CargDet(d,e,val,prom);
    Ganancia(c);
    CargarCon2(c);
    CargarDet(d);
    return 0;
}
```