LINGUAXE C++ (2ª parte)

ARRAYS

Imaxinade que temos que realizar un programa no que temos que introducir a nota de catro alumnos dende o teclado e calcular a nota media. Para isto teríamos que definir catro variables nota1, nota2, nota3 e nota4. Ao ser só catro variables non é moi incómodo declaralas pero que pasaría se queremos calcular a nota media de trinta alumnos, a expresión que obteríamos sería enorme. A solución está nos arrays.

Un array é un colección de datos **do mesmo tipo** que se referencian por medio dun nome en común e que se almacenan en posicións contiguas de memoria.

Os arrays poden ser de diferentes dimensións. Os máis usados son os dunha dimensión (vectores) e os de dúas dimensións (táboas ou matrices). Un array ten unha dimensión e un tamaño. A dimensión ven dada polo número de pares de corchetes que ten e o número de elementos é o resultado da multiplicación dos números que aparecen nos corchetes. Por exemplo int matriz[4][5] é un array que ten dúas dimensións e 20 elementos.

Exemplos de declaracións de arrays son as seguintes:

int v[3]; //Vector de 3 elementos

int taboa[2][6]; //Táboa ou matriz de dous dimensións

float vol[1][3][4]; //Array de tres dimensións

int v[3]={2,4,5}; //Vector de tres elementos inicializados cos valores 2,4 e 5

int matriz[2][2]={4,3,2,1}; //Matriz con 2 filas e 2 columnas inicializada con estes valores

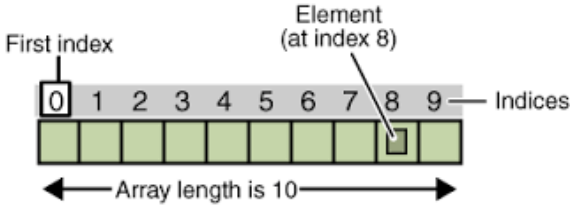
Para favorecer a súa visualización tamén pódense declarar así:

```
int matriz[2][2]=
{
   4, 3,
   2, 1
};
char letras[2][2]=
{
‘a’,‘b’
‘c’,’d’
}
```

Se queremos establecer o valor 10 no elemento da posición 0 dun array unidimensional (vector) faríamolo así:

```
v[0]=10;
```

A representación gráfica dun vector sería a seguinte:



Hai que fixarse que a primeira posición é a 0 non a 1.

A representación da matriz a[3][4] sería:

	Column 0	Column 1	Column 2	Column 3
Row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]

Darlle valor por teclado a un vector de 10 elementos e visualizalos posteriormente por pantalla.

```
//Darlle valores a un array de 10 enteiros e lelos despois por pantalla

#include<iostream>
using namespace std;

int main()
{
int v[10],i; //Declaramos as variables

for(i=0;i<10;i++)
{
cout<<"Introduce o valor do vector na posicion "<<i<<" ";
cin>>v[i];
}//Fin do for de entrada dos valores do array

for(i=0;i<10;i++)
{
cout<<v[i]<<" ";
}//Fin do for de imprimir os valores do array

return 0;
}//Fin do main
```

Inicializa percorrendo un array bidimensional (matriz) 4x3 todos os seus elementos a 2 e amosa o resultado por pantalla.

```
#include<iostream>
using namespace std;

int main()
{
int i,j,matriz[4][3];
//Percorremos a matriz e poñemos todos os valores a 2
for(i=0;i<4;i++)
{
for(j=0;j<3;j++)
{
matriz[i][j]=2;
}
}
//Percorremos a matriz e visualizamos cada valor
for(i=0;i<4;i++)
{
for(j=0;j<3;j++)
{
cout<<matriz[i][j]<<" ";
}
cout<<endl;
}
return 0;
}// Fin do main
```

Encher unha matriz 4x3 e amosar o resultado por pantalla

```
#include<iostream>
using namespace std;
int main()
{
int i,j,matriz[4][3];
//Percorremos a matriz e introducimos os valores

for(i=0;i<4;i++)
{
for(j=0;j<3;j++)
{
cout<<"Introduce o valor da posicion "<<i<<j<<":";
cin>>matriz[i][j];
}
}
for(i=0;i<4;i++) //Percorremos a matriz e visualizamos cada valor
{
for(j=0;j<3;j++)
{
cout<<matriz[i][j]<<" ";
}
cout<<endl;
}
return 0;
}// Fin do main
```

Crea unha matriz 10x10 na que apareza en cada posición o resultado da suma da posición na fila e na columna. Ex.: Na posición 23 ten que aparecer un 5.

```
#include <iostream>
using namespace std;

int main()
{
int i,j,matriz[10][10];

for(i=0;i<10;i++)
{
for(j=0;j<10;j++)
{
matriz[i][j]=i+j;
}
}

for(i=0;i<10;i++)
{
for(j=0;j<10;j++)
{
cout<<matriz[i][j]<<"       ";
}
cout<<endl;
}

return 0;
} //Fin do main
```

Encher e amosar en pantalla un array bidimensional de tamaño escollido polo usuario

/*Para introducir una matriz dun tamaño que varia no tempo de execución deberíamos usar memoria dinámica un truco para solucionalo sen ter que recurrir á mesma é o seguinte */

```
#include<iostream>
using  namespace std;

int main()
{
int filas,columnas,i,j;

cout<<"Introduce o numero de filas: ";
cin>>filas;
cout<<"Introduce o numero de columnas: ";
cin>>columnas;

int matriz[filas][columnas];

for(i=0;i<filas;i++)
{
for(j=0;j<columnas;j++)
{
cout<<"Introduce o valor da matriz na posicion "<<i<<j<<":";
cin>>matriz[i][j];
}
}
for(i=0;i<filas;i++)
{
for(j=0;j<columnas;j++)
{
cout<<matriz[i][j]<<" ";
}
cout<<endl;
}
return 0;
}//Fin do main
```

Tamén podería solucionarse definindo un valor máximo de filas e columnas:

```
#include<iostream>
#define maxfil 10
#define maxcol 10

using namespace std;

int main()
{
int matriz[maxfil][maxcol],filas,columnas,i,j;

cout<<”Introduce un número de filas menor o igual que 10: ";
cin>>filas;
cout<<"Introduce un número de columnas menor o igual que 10: ";
cin>>columnas;
```

Nunha clase de 6 alumnos temos que introducir o número de expediente e as notas das tres avaliacións, o programa ten que calcular a nota media dos alumnos e amosar por pantalla cal é a nota media máxima e o número de expediente do alumno que a obtivo. Axuda: Podedes crear un array 5x6 no que a primeira fila sería o número de expediente, as filas 2, 3 e 4 as notas das avaliacións e na quinta fila calcularíamos a nota media.

```
#include<iostream>
#include<iomanip>
using namespace std;

int main()
{
float notas[5][6],notamax=0,expmax;
int i,j;

//Introducimos os expedientes e as notas
for(j=0;j<6;j++)
{
cout<<"Introduce o expediente do alumno "<<j+1<<" ";
cin>>notas[0][j];
for(i=1;i<=3;i++)
{
cout<<"Introduce a nota "<<i<<" do alumno "<<j+1<<" ";
cin>>notas[i][j];
}
}

//Calculo a nota media de cada alumno
for(j=0;j<6;j++)
{
notas[4][j]=(notas[1][j]+notas[2][j]+notas[3][j])/3;
}
//Visualizo a matriz
for(i=0;i<5;i++)
{
for(j=0;j<6;j++)
{
if(i==0)
{
cout<<notas[i][j]<<"    ";
}
else
{
cout<<fixed<<setprecision(2)<<notas[i][j]<<"   ";
}
}
cout<<endl;
}
//Determino a nota máxima e o mellor expediente
for(j=0;j<6;j++)
{
if(notas[4][j]>notamax)
{
notamax=notas[4][j];
expmax=notas[0][j];
}
}
```

```
cout<<"A nota maxima e: "<<notamax<<endl;
cout<<"Os mellores expedientes son: "; //Pode haber varios coa nota máxima
for(j=0;j<6;j++)
{
if (notas[4][j]==notamax)
{
cout<<fixed<<setprecision(0)<<notas[0][j]<<" ";
}
}
return 0;
}//Fin do main
```

Ordear de menor a maior un vector de 10 números.

```
#include<iostream>
using namespace std;

int main()
{
int i,j,aux,vector[10];

//Introducimos os valores do vector
for (i=0;i<10;i++)
{
cout<<"Introduce o valor do elemento "<<i+1<<" ";
cin>>vector[i];
}

//Visualizamos o vector antes de ser ordenado
for(i=0;i<10;i++)
{
cout<<vector[i]<<" ";
}
cout<<endl;

//Ordenamos o vector
for(j=0;j<9;j++)
{
for(i=0;i<9;i++)
{
if(vector[i]>vector[i+1])
{
aux=vector[i+1];
vector[i+1]=vector[i];
vector[i]=aux;
}
}
} //Temos que repetir o proceso o (número de elementos do vector-1) veces

//Vemos o vector xa ordenado
for(i=0;i<10;i++)
{
cout<<vector[i]<<" ";
}
return 0;
}//Fin do main
```

Acepta o reto “Cantos días faltan para fin de ano” con vectores

```
#include<iostream>
using namespace std;
int main()
{
int i,j,casos,meses[12]={31,28,31,30,31,30,31,31,30,31,30,31},d,m,resultado;
cout<<"Cantas datas vas introducir? ";
cin>>casos;
for(i=0;i<casos;i++)
{
cout<<"Introduce o dia e o mes separado por un enter ";
cin>>d>>m;
resultado=meses[m-1]-d;
for(j=m;j<12;j++)
{
resultado=resultado+meses[j];
}
cout<<resultado<<endl;
}
}// Fin do main
```

Acepta o reto “aboa María”

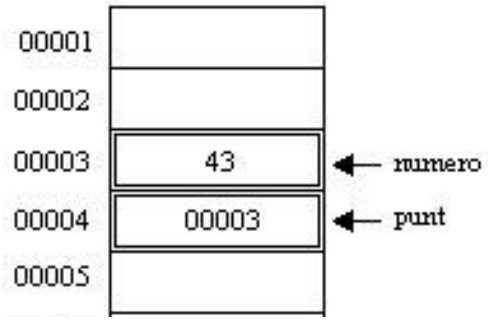
```
#include<iostream>
using namespace std;
int main()
{
int a[6],b[6],s[6],i,j,k,flag=0,casos;

cin>>casos;
for(k=0;k<casos;k++)
{
for(i=0;i<6;i++)
{
cin>>a[i];
}
for(i=0;i<6;i++)
{
cin>>b[i];
}
for(i=0;i<6;i++)
{
s[i]=a[i]+b[i];
}
for(i=0;i<5&&flag==0;i++)
{
if(s[i]!=s[i+1])
{
flag=1;
}
}
if (flag==1)
{
cout<<"NO";
}
else
{
cout<<"SI";
}
}
}//Fin do main
```

Acepta o reto “matrices triangulares”

```
#include<iostream>
using namespace std;
int main()
{
int f,i,j,a=1,b=1;

//Introduzo o número de filas
cout<<"Introduce o numero de filas: ";
cin>>f;
//Introduzo os elementos da matriz
cout<<"Introduce os elementos da matriz: "<<endl;
int matriz[f][f];
for(i=0;i<f;i++)
{
for(j=0;j<f;j++)
{
cin>>matriz[i][j];
}
}
//Visualizo a matriz
for(i=0;i<f;i++)
{
for(j=0;j<f;j++)
{
cout<<matriz[i][j]<<" ";
}
cout<<endl;
}
//Analizo se son triangulares
for(i=0;i<f;i++)
{
for(j=0;j<f;j++)
{
if(i<j)
{
if(matriz[i][j]!=0)
{
a=0;
}
}
if(i>j)
{
if(matriz[i][j]!=0)
{
b=0;
}
}
}
}
if(a==1&&b==1)
{
cout<<"Matriz triangular superior e inferior";
}
else if(a==1&&b==0)
{
cout<<"Matriz triangular inferior";
}
else if(a==0&&b==1)
{
cout<<"Matriz triangular superior";
}
else
{
cout<<"Non e triangular";
}
}
```

PUNTEIROS

Para sacarlle o máximo partido ás funcións, que será o seguinte apartado é necesario explicar previamente que é un punteiro. Un punteiro é unha variable cuxo valor é unha dirección de memoria.



Se declaran así:

int *j; //Cun * diante do punteiro e o tipo de variable ten que coincidir co tipo de dato que se almacena nesa dirección de memoria j, neste caso un enteiro.

float *p; //Punteiro a unha variable tipo float.

Cos punteiros trabállase con dous operadores: & e *.

Se declaramos unha variable a:

int a; //Declara unha variable enteira
a é o valor da variable.
&a é a dirección de memoria onde está a variable a.

Se declaramos un punteiro p:

int *p;
p é o valor da dirección na que hai un enteiro.
*p é o valor do enteiro almacenado na dirección apuntada por p.

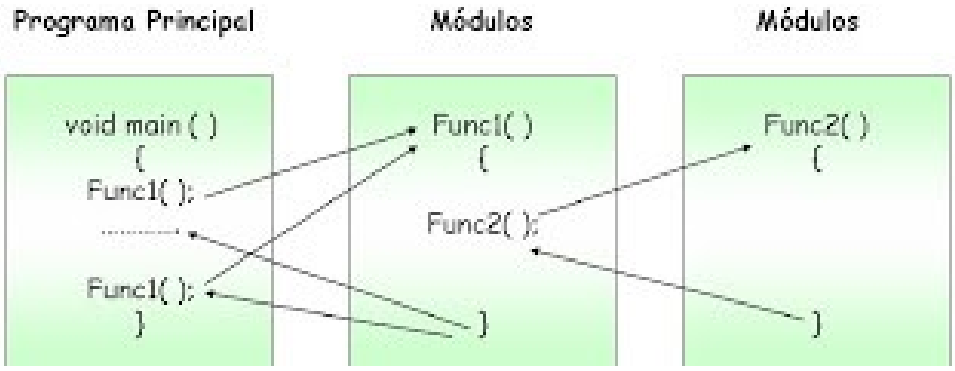
& coñécese como o operador dirección porque seguido por unha variable dá a dirección de dita variable.

* coñécese como o operador indirección porque seguido por un punteiro da o valor almacenado na dirección sinalada polo mesmo.

Exemplo:

```
#include<iostream>
using namespace std;
int main(){

int a=20;
int *p;
p=&a;
cout<<a<<endl; //Obtemos o valor de a
cout<<*p<<endl; //Obtemos o valor de a
cout<<&a<<endl; //Obtemos o valor da dirección en memoria onde está o dato a
cout<<p<<endl; //Obtemos o valor da dirección en memoria onde está o dato a
return 0;
}
```

FUNCIÓNS (Divide e vencerás)



As funcións son a base da programación modular, na que cada parte do programa desenvolve un cometido específico. Permítenos descompoñer un programa en subprogramas de tal xeito que os programas sexan máis claros e a base de dividir un problema grande en problemas menores que sexa máis doado de resolver.

As chamadas as funcións realizanse dende a función principal indicando o seu nome, cando este aparece pasa a executarse o código da función e cando remata continuase executando a liña de código seguinte do programa principal.

Imos estudalas con exemplos:

//1. Táboa de multiplicar dun número traballando con funcións sen paso de parámetros

Temos que declaralas ao comezo do programa. No caso de meter o código da función antes da función main non teríamos que facelo.

```
#include <iostream>
using namespace std;

void taboa_multiplicar();  /*Declaramos a función táboa de multiplicar, é de tipo void porque non vai retornar algún valor*/

int main()
{
cout<<"Ola"<<endl;

taboa_multiplicar();

cout<<endl<<"Adeus"<<endl;

return 0;
}//Fin do main

void taboa_multiplicar()
{
int x,i;
cout<<endl<<"Introduce o numero do que queres facer a taboa de multiplicar: ";
cin>>x;
cout<<endl;
for(i=0;i<=10;i++)
{
cout<<x<<"*"<<i<<"="<<x*i<<endl;
}
}//Fin da función táboa de multiplicar
```

//2. función exemplo suma de dous enteiros con paso de parámetros por valor

```
#include<iostream>
using namespace std;

void suma(int a, int b); //Declaramos a función suma

int main(){

int num1,num2;
cout<<"Introduce o primeiro numero: ";
cin>>num1;
cout<<"Introduce o segundo numero: ";
cin>>num2;

suma(num1,num2); //Chamada á función suma e pasamos o valor de num1 a a e o de num2 a b

return 0;
}

void suma(int a, int b) //Función suma coa definición dos parámetros
{
int resultado;
resultado=a+b;
cout<<"O resultado e: "<<resultado;
}
```

//Táboa de multiplicar dun número traballando con funcións con paso de parámetro por valor

```
#include <iostream>
using namespace std;
void taboa_multiplicar(int a); //Declaramos a función táboa de multiplicar

int main(){

int numero;
cout<<"Ola"<<endl;
cout<<"Introduce o numero do que queres calcular a taboa de multiplicar: ";
cin>>numero;
taboa_multiplicar(numero);
cout<<"Adeus"<<endl;

return 0;
}

void taboa_multiplicar(int a)
{
int i,resultado;
for(i=0;i<11;i++){
resultado=a*i;
cout<<a<<" * "<<i<<" = "<<resultado<<endl;
}
}
```

//3. función exemplo paso de parámetros por valor e retorno de datos na suma de dous enteiros

```
#include<iostream>
using namespace std;

int suma(int a, int b);/*Declaramos a función suma que é de tipo int porque hai un valor de retorno de tipo enteiro*/

int main(){

int num1,num2,resultado_suma;
cout<<"Introduce o primeiro numero: ";
cin>>num1;
cout<<"Introduce o primeiro numero: ";
cin>>num2;

resultado_suma=suma(num1,num2); //Chamamos á función suma
cout<<"O resultado e: "<<resultado_suma;

return 0;
}

int suma(int a, int b)//función suma
{
int resultado;
resultado=a+b;
return resultado; // Devolve o valor resultado e introdúceo en suma (num1,num2) na función main
}
```

//4. funcións exemplo paso de parámetros por referencia intercambio dos valores de dúas variables

```
#include<iostream>

void cambio (int *a, int*b);//Declaramos a función cambio

int main(){

int x=10,y=5;

cambio(&x,&y);

cout<<"O valor actual de x e “<<x<<”e o valor actual de y e "<<y;

return 0;
}

void cambio (int *a, int *b)//función cambio
{
int aux;
aux=*a;
*a=*b;
*b=aux;
}
```

&x=a e &y=b

VECTORES E FUNCIÓNS

//Introducir, ordear e imprimir un vector de 6 elementos traballando con funcións

```
#include<iostream>
using namespace std;

void introducir(int a[6]); //Non é obrigatorio meter o número de elementos, podería quedar en branco
void ordear(int b[6]);
void imprimir(int c[6]);
```

Moi importante: En linguaxe C++ cando pasamos un array (vector, matriz) como parámetro, a modificación do contido do array nunha función implicará a modificación automática do array orixinal da función principal. (Paso de parámetros por referencia)

```
int main(){
int v[6];            //Declaramos un vector de 6 elementos de tipo enteiro

introducir(v);     //Chamada á función introducir
ordear(v);     //Chamada á función ordear
imprimir(v);     //Chamada á función imprimir

return 0;
}

void introducir(int a[6]){
int i;
for (i=0;i<6;i++){
cout<<"Introduce o valor do vector na posicion "<<i+1<<" ";
cin>>a[i];
}     // fin do for
}     // fin da función introducir

void ordear (int b[6]){
int i,j,aux;
for (i=0;i<5;i++){
    for (j=0;j<5;j++){
      if(b[j]>b[j+1]){
        aux=b[j+1];
        b[j+1]=b[j];
        b[j]=aux;
        }//Fin do if
   }     //Fin do segundo for
}     //Fin do primeiro for
}     //Fin da función ordear


void imprimir (int c[6]){
int j;
for(j=0;j<6;j++){
cout<<c[j]<<" ";

}    //fin do for
}    //fin da función imprimir
```

MATRICES E FUNCIÓNS

É similar ao traballo con vectores e funcións, pero temos que declarar o número de filas (este dato podería quedar en branco) e o número de columnas.

//Introducir valores aleatorios nunha matriz e visualizalos por pantalla

```
#include<iostream>
#include<time.h>
#include<stdlib.h>
using namespace std;

void introducir(int matriz[15][10]);

void visualizar(int matriz[15][10]);

int main(){

int matriz[15][10];

introducir (matriz);         //Chamada á función introducir

visualizar (matriz);        //Chamada á función visualizar

return 0;
}

void introducir(int matriz[15][10]){
int i,j;
srand(time(NULL));           //Toma como valor semilla para xerar os números aleatorios o tempo medido en segundos

for(i=0;i<15;i++){
   for(j=0;j<10;j++){
      matriz[i][j]=rand()%10;         //%10 porque toma valores entre 0 e 9
      }                               //Fin do for das columnas
   }                                 //Fin do for das filas

}                                    //Fin da función introducir

void visualizar(int matriz[15][10]){
int i,j;
for(i=0;i<15;i++){
   for(j=0;j<10;j++){
         cout<<matriz[i][j]<<" ";
         }                             //Fin do for das columnas
        cout<<endl;
        }                              //Fin do for das filas

}                                      //Fin da función visualizar
```

Recursividade

Unha función recursiva é unha función que se chama a si mesma. Nestas funcións sempre temos que definir un **caso base** e outro **caso xeral**. O caso base é o que evita que nos metamos nun bucle infinito, cando chegamos a el fai parar a recursión xa que non chama a función. No caso xeral a función chámase a sí mesma e fai que a recursión continue ata que cheguemos ao caso base.
Aquí temos tres exemplos de funcións recursivas (factorial dun número, suma de n números e termo n da serie de Fibonacci).

//Factorial con recursividade

```
#include<iostream>
using namespace std;

int factorial(int n);
int main()
{
int num;
cout<<"Introduce o numero do que queres calcular o factorial: ";
cin>>num;
cout<<"O factorial e: "<<factorial(num);
return 0;
}
int factorial(int n)
{
int resultado=0;
if(n==0)  //Caso base
{
resultado=1;
        }
else
{
resultado=n*factorial(n-1);   //Caso xeral
        }
return resultado;
}
```

//Suma con recursividade

```
#include<iostream>
using namespace std;

int suma(int n);
int main()
{
int num;
cout<<"Introduce ata que numero queres sumar ";
cin>>num;
cout<<"A suma total e: "<<suma(num);
return 0;
}
```

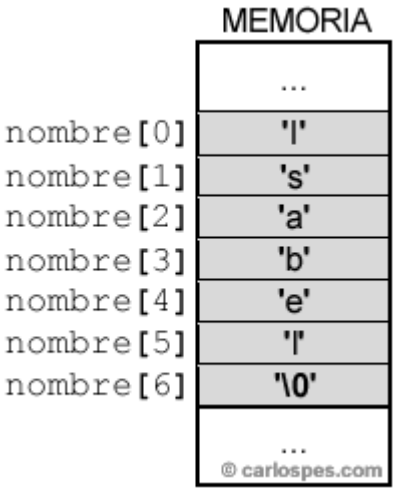
```
int suma(int n)
{
int s=0;   //Caso base
if(n==0)
{
s=0;
        }
else
{
s=n+suma(n-1);   /Caso xeral
        }
return s;
}
```

//Termo da serie de Fibonacci con recursividade

```
#include<iostream>
using namespace std;
int fib(int n);
int main()
{
int num;
cout<<"Introduce o numero do termo que queres calcular: ";
cin>>num;
cout<<"O termo e: "<<fib(num);
return 0;
}
int fib(int n)
{
int f=0;
if(n==1||n==2)
{
f=1; //Caso base
}
else
{
f=fib(n-1)+fib(n-2); //Caso xeral
}
return f;
}
```

CADEAS DE CARACTERES (STRINGS)

As cadeas de caracteres trátanse en C++ como arrays de tipo char.



Se eu declaro char nombre[7] teño que ter coidado porque ao final dunha cadea sempre engadese un carácter máis o \0 que sinala onde termina a cadea, polo que só podería gardar 6 caracteres.

Para almacenar en variables cadenas lidas por teclado temos diferentes opcións: cin, gets e fgets que as comentamos nos seguintes exemplos:

//Traballo con cadeas de caracteres

```
#include<iostream>
#include<string.h>
using namespace std;

int main(){

char nome[10]="Pablo";    //Exemplo de inicialización dunha cadea de caracteres

cout<<"O teu nome e "<<nome<<endl;

char palabra[10];    //Almacenar unha palabra con cin

cout<<"Escribe una palabra: "<<endl;
cin>>palabra;
cout<<palabra<<endl;

char frase[20];      //Almacenar unha frase con cin. Problema: Remata no momento no que aparece un espazo

cout<<"Escribe unha frase: "<<endl;
cin>>frase;
cout<<endl<<frase;

fflush(stdin);      //Limpia o buffer de memoria

char texto[100];      /*Almacenar unha frase con gets. Moito coidado co desbordamento,
se lle introduces máis caracteres dos sinalados na cadea sobreescribe posicións de memoria*/

cout<<endl<<"Escribe unha frase: ";
gets(texto);
cout<<endl<<texto;

char escrito[20];   //Agora utilizamos fgets que evita problemas de desbordamento de memoria

cout<<endl<<"Escribe unha frase: ";
fgets(escrito,14,stdin);    //14 é o número máximo de caracteres e stdin e a entrada standard que é o teclado
cout<<endl<<escrito;
}
```

```
char text[11];       //Cambiar o salto de liña que pode quedar ao facer a entrada con fgets

cout<<endl<<"Escribe unha frase: ";
fgets(text,11,stdin);

int i;

for(i=0;i<11;i++){
   if(text[i]=='\n'){
         text[i]='\0';
}
}
cout<<endl<<text;
```

Na librería string.h temos moitas funcións para o traballo con cadeas, aquí vemos algúns exemplos:

```
// Funcións de cadeas de caracteres

#include<iostream>
#include<string.h>
using namespace std;

int main(){

char cadea[10];              //strcpy Copia a cadea de orixe na de destino strcpy(destino,orixe)
strcpy(cadea,"Pablo");
cout<<endl<<cadea;

char texto[]="Imos medir a lonxitude dunha cadea";  //strlen danos o número de caracteres dunha cadea sen contar o \0
cout<<endl<<"Imos medir a lonxitude dunha cadea";
int lonxitude;
lonxitude=strlen(texto);
cout<<endl<<lonxitude;

//Agora imos comparar dúas cadeas coa función strcmp que devolve un 0 se as dúas cadeas son iguais

char cadea1[15],cadea2[15];

cout<<endl<<"escribe a primeira frase"<<endl;
fgets(cadea1,15,stdin);
cout<<endl<<"escribe a segunda frase"<<endl;
fgets(cadea2,15,stdin);

if (strcmp(cadea1,cadea2)==0)
   cout<<endl<<"As cadeas son iguais";
else
         cout<<endl<<"As cadeas son distintas";

return 0;
}
```

Estruturas (Rexistros) C++

As estruturas (ou rexistros) é unha agrupación de datos (campos) que non teñen porque ser do mesmo tipo. Defínense coa orde struct. Para acceder a cada un dos datos que forman a estrutura, tanto para lelo ou para introducilo ou cambialo, débese indicar o nome da variable da estrutura e o dos campos separados por un punto. Podense declarar na función principal ou antes do int main.

//Rexistros (struct) C++

```
#include<iostream>
using namespace std;
int main()
{
struct alumno
{
string nome;
char inicial;
int idade;
float nota;
} AL1,AL2,AL3;  //Son os nomes das variables desta estrutura, cada variable ten todos os campos
```

Nome da estrutura

Tamén podemos introducir as variables así:

```
struct alumno
{
string nome;
char inicial;
int idade;
float nota;
};
alumno AL1,AL2,AL3;
```

Se queremos introducir os datos do primeiro alumno faríamos:

```
AL1.nome="Manel";
AL1.inicial='M';
AL1.idade=25;
AL1.nota=9.5;

cout<<"O nome do alumno e: "<<AL1.nome<<endl;
cout<<"A inicial do alumno e: "<<AL1.inicial<<endl;
cout<<"A idade do alumno e: "<<AL1.idade<<endl;
cout<<"A nota do alumno e: "<<AL1.nota<<endl;
return 0;
}
```

Tamén poderíamos telo feito na propia declaración da estrutura:

```
struct alumno
{
string nome;
char inicial;
int idade;
float nota;
}
AL1={“Manel”,’M’,25,9.5},
AL2={“Pepe”,’P’,18,7},
AL3={“Elena”,’E’,17,10};
```

//Exemplo dunha estrutura con cancións, no que introducimos por teclado os valores

```
#include<iostream>
using namespace std;

int main()
{
struct musica
{
string titulo;
string cantante;
int duracion;
float peso;
}cancion;

cout<<"Introduce o nome da cancion: ";
getline(cin,cancion.titulo);  //Utilizamos getline para poder poñer espazos.
cout<<"Introduce o nome do cantante: ";
getline(cin,cancion.cantante);
cout<<"Introduce a duracion da cancion en segundos: ";
cin>>cancion.duracion;
cout<<"Introduce o tamano do ficheiro da cancion en Mb: ";
cin>>cancion.peso;

cout<<"O nome da cancion e: "<<cancion.titulo<<endl;
cout<<"O nome do cantante e: "<<cancion.cantante<<endl;
cout<<"A duracion da cancion e: "<<cancion.duracion<<endl;
cout<<"O tamano do ficheiro e: "<<cancion.peso<<endl;

return 0;
}
```

É posible almacenar os datos de varias persoas dentro dunha única variable se combinamos as struct cos arrays, como podemos ver no seguinte exemplo que nos permite ver as cancións de u2 introducidas nunha lista.

//Array de estruturas

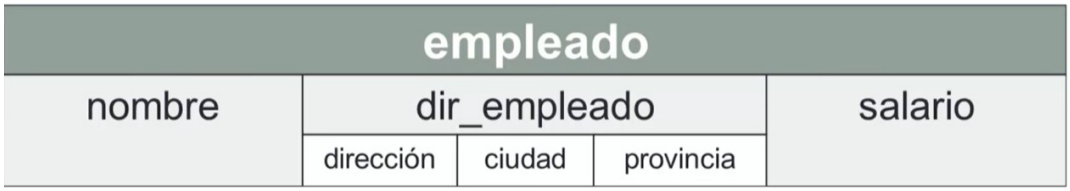
```
#include<iostream>
using namespace std;

int main()
{
int i;
struct musica
{
string titulo;
string cantante;
int duracion;
float peso;
};
musica cancion[4];

for(i=0;i<4;i++)
{
```

```
cout<<"Introduce o nome da cancion: ";
getline(cin,cancion[i].titulo);
cout<<"Introduce o nome do cantante: ";
getline(cin,cancion[i].cantante);
cout<<"Introduce a duracion da cancion en segundos: ";
cin>>cancion[i].duracion;
cout<<"Introduce o tamano do ficheiro da cancion en Mb: ";
cin>>cancion[i].peso;
fflush(stdin);
}

for(i=0;i<4;i++)
{
if(cancion[i].cantante=="u2")
{
cout<<"O nome da cancion e: "<<cancion[i].titulo<<endl;
cout<<"O nome do cantante e: "<<cancion[i].cantante<<endl;
cout<<"A duracion da cancion e: "<<cancion[i].duracion<<endl;
cout<<"O tamano do ficheiro e: "<<cancion[i].peso<<endl<<endl;
}
}
return 0;
}
```

Exercicio: Programa un array de estruturas chamado atleta para 5 atletas que conteña os seguintes campos: nome, país, numero_medallas e devolva os datos do atleta (nome e país) que gañou o maior número de medallas.

Estruturas anidadas

Aquí tedes o seguinte exemplo:




```
struct info_direccion{
    char direccion[30];
    char ciudad[20];
    char provincia[20];
};

struct empleado{
    char nombre[20];
    struct info_direccion dir_empleado;
    double salario;
};
```

Dentro da estrutura empleado, un dos campos é a estrutura info_direccion.

O programa para introducir e visualizar os datos quedaría do seguinte xeito para un array de dous empleados :

```
#include<iostream>
using namespace std;
int main()
{
int i;
struct info_direccion
{
string direccion;
string cidade;
string provincia;
};
struct empleado
{
string nome;
struct info_direccion dir_empleado;
double salario;
}empleados[2];
for(i=0;i<2;i++)
{
fflush(stdin);
cout<<"Introduza o nome do empleado: ";
getline(cin,empleados[i].nome);
cout<<"Introduza a sua direccion: ";
getline(cin,empleados[i].dir_empleado.direccion);
cout<<"Introduza a sua cidade: ";
getline(cin,empleados[i].dir_empleado.cidade);
cout<<"Introduza a sua provincia: ";
getline(cin,empleados[i].dir_empleado.provincia);
cout<<"Introduza o seu salario: ";
cin>>empleados[i].salario;
cout<<endl;
}
//Visualizamos os datos
for(i=0;i<2;i++)
{
cout<<"Nome: "<<empleados[i].nome<<endl;
cout<<"Direccion: "<<empleados[i].dir_empleado.direccion<<endl;
cout<<"Cidade: "<<empleados[i].dir_empleado.cidade<<endl;
cout<<"Provincia: "<<empleados[i].dir_empleado.provincia<<endl;
cout<<"Salario: "<<empleados[i].salario<<endl<<endl;
}
return 0;
}
```

Como campo en vez da variable string podemos utilizar arrays, en vez de string nome; podemos poñer char nome[30]; . Nese caso cambia como temos que poñer o getline que sería:

cin.getline(empleados[i].nome,20,'\n'); onde 20 é o número de caracteres do array e \n é o carácter final da cadea.

Exercicio: Crea dúas estruturas, unha chamada promedio que terá os seguintes campos: nota1, nota2 e nota3; e outra chamada alumno con: nome, sexo, idade e a estrutura promedio anidada

aquí. Logo pedir todos os datos para un alumno, calcular a súa nota promedio como media das tres e para rematar imprimir todos os datos, incluído a nota media.

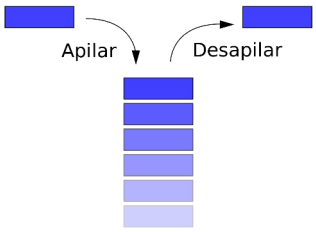
Exercicio: Define unha estrutura que sirva para representar a unha persoa. A estructura debe conter dous campos: o nome da persoa e un valor de tipo lóxico que indica se a persoa ten algún tipo de discapacidade. Fai un programa que dado un vector de persoas rechee dos novos vectores da mesma estrutura: un que conteña as personas que non teñen ningunha discapacidade e outro que conteña as personas con discapacidade e amose os nomes por pantalla.

Estruturas dinámicas: Pilas, colas e listas

Ao contrario que un array, que ten un espazo para almacenar un número fixo de elementos, unha estrutura dinámica ampliase e contraese durante a execución do programa. Algunhas das estruturas dinámicas máis coñecidas son as pilas, as colas e as listas.

Pilas

As pilas (stack) son estruturas nas que os datos recuperanse pola última posición, polo que obtense na orde inversa na que se gardaron. (LIFO: last in first out).



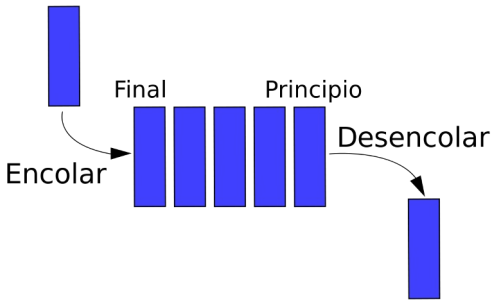
//Pilas

```
#include<iostream>
#include<stack>
using namespace std;

int main()
{
stack <int> pila; //declaro a pila

pila.push(35); //Introduzo datos na pila
pila.push(40);
pila.push(20);
pila.push(3);
pila.push(10);
cout<<pila.top()<<endl; //A función top permite visualizar un elemento, pero só ten acceso ao último elemento.
//Para poder visualizar todos os elementos teño que sacar este elemento (10) que está na cima. Para iso utilizo pila.pop().
pila.pop(); //Para ver todos os elementos teño que repetir este código 5 veces, para mellorar isto utilizo un bucle while
return 0;
}
while(pila.size()>0)  //pila.size danos o tamaño da pila
{
cout<<pila.top()<<endl;
pila.pop();
}
```

Exercicio: Calcula o máximo dos elementos dunha pila.

Colas

As colas (queue) son estruturas nas que os datos recuperanse pola primeira posición, polo que obtense na mesma orde inversa na que se gardaron. (FIFO: first in first out).




```
//Exemplo de cola
#include<iostream>
#include<queue>
using namespace std;
int main()
{
queue <int> cola; //declaro a cola
cola.push(35); //Introduzo datos na cola
cola.push(40);
cola.push(20);
cola.push(3);
cola.push(10);
while(cola.size()>0)  //cola.size danos o tamaño da cola
{
cout<<cola.front()<<endl; //Imprime o primeiro elemento que metimos.
cola.pop(); //Primeiro elimina o elemento 35 que está na primeira posición para saír.
}
}// Fin do main
```

Listas

Coa clase list é fácil insertar elementos ao principio e ao final, así como lelos dende o principio e dende o final. Aquí tedes un exemplo.

```
#include<iostream>
#include<list>
using namespace std;
int main()
{
int i;
list<int>lista;
cout<<"Engadimos 5 elementos ao principio"<<endl;
for(i=1;i<=5;i++)
{
lista.push_front(i); //lista.push_front inserta un elemento en la parte delantera de una lista
}
cout<<"Engadimos 5 elementos ao final"<<endl;
for(i=6;i<=10;i++)
{
lista.push_back(i);   //lista.push_back inserta un elemento después del último elemento de la lista
}
cout<<"Primeiro dato: "<<lista.front()<<endl;
cout<<"Ultimo dato: "<<lista.back()<<endl<<endl;

//Ahora extraemos toda a lista e mostramos todos os valores
while(lista.size()>0)
{
cout<<lista.front()<<" ";
lista.pop_front();
}
}// Fin do main
```