

# Apuntadores e Iteradores

Por Ariel Parra.

on my puter

# How Pointer Works in C++



Un apuntador también conocido como punteros, son la representación de una dirección, estos sirven para ahorrar memoria, crear vectores y matrices con memoria dinámica y como paso de parámetros.

Este se declara con el operador "\*", y se le asignan direcciones de memoria con el operador "&".

## Ejemplo:

```
int main(){  
    int var = 20;  
    int* ptr; //ó int *ptr=NULL;  
    ptr = &var; // el tipo de dato tiene que ser igual  
  
    cout<<endl<< ptr;  
    cout<<endl<< var;  
    cout<<endl<<*ptr; //operador de desreferencia  
    *ptr=5;  
    cout<<endl<<*ptr;  
    return 0;  
}
```

# Paso de parametros

## Con punteros

```
void squarePtr(int* n){  
    *n *= *n;  
}
```

## Con Referencias

```
void squareRef(int& n){  
    n *= n;  
}
```

# Vectores

los vectores apuntan a una dirección de memoria que es el inicio de este.

```
int main(){
    int vec[3] = { 10, 20, 30 };
    int* ptr;
    ptr = vec; // ó ptr=&vec[0];
    for(int i=0;i<3;i++){
        cout<<endl<<*(ptr+i); //logica de punteros
        //cout<<endl<< ptr[i];
    }
    return 0;
}
```



# Matrices

Las matrices no son más que vectores donde cada valor indica la dirección de memoria de otro vector.

```
int main(){
    const int REN=3, COL=3;
    int mat[REN][COL]={ { 10, 20, 30 },
                        { 40, 50, 60 },
                        { 70, 80, 90 } };

    int* ptr;
    ptr = &mat[0][0];
    for(int i=0; i<REN; i++){
        for(int j=0; j<COL; j++){
            cout<<endl<<*(ptr+i * COL+j) ;
        }
    }
    return 0;
}
```

# Memoria Dinamica

Es la cantidad de memoria que se va usar por tipo de datos y su longitud se define en tiempo de ejecución.

```
int main(){
    int n; cin>>n;
    int* vec = new int[n];
    for(int i=0;i<n;i++){
        cin>>*(vec+i); //cin>>vec[i];
    }
    for(int i=0;i<n;i++){
        cout<<endl<<*(vec+i);
    }
    delete[] vec; // ó delete []vec;
    return 0;
}
```

# Funciones de tipo puntero

```
int *crearVec(int tam){
    int* aux = new int[n];
    for(int i=0;i<tam;i++){
        *(aux+i)=i;
    }
    return aux;
}

int main(){
    int n; cin>>n;
    int* vec = crearVec(n);
    for(int i=0;i<n;i++){
        cout<<endl<<*(vec+i);
    }
    delete[] vec;
    return 0;
}
```

# Matrices con memoria dinamica

```

int main() {
    int REN, COL; cin>>REN>>COL;
    int** mat = new int*[REN];
    for (int i=0;i<REN;i++){
        mat[i] = new int[COL];
    }
    for (int i=0;i<REN;i++){
        for (int j= 0;j<COL;j++){
            cin>>*(*(mat+i)+j); //cin>>mat[i][j];
        }
    }
    for (int i=0;i<REN;i++){
        for (int j= 0;j<COL;j++){
            cout<<mat[i][j]<<" ";
        };cout<<endl;
    }
    for (int i=0;i<REN;i++){
        delete[] mat[i];
    }
    delete[] mat;
    return 0;
}

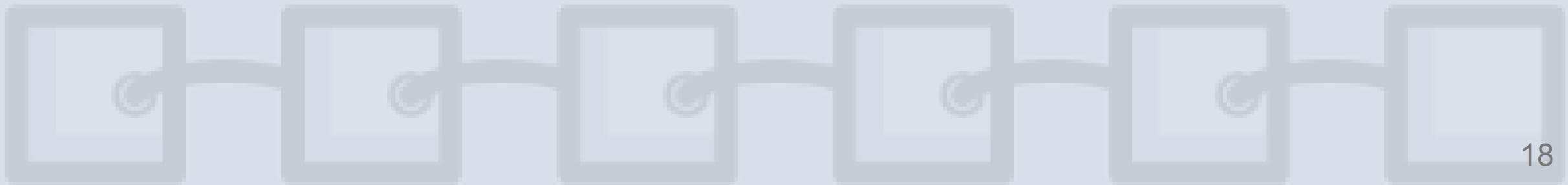
```

# Matrices con vectores STD



```
int main() {  
    int REN, COL; cin>>REN>>COL;  
    vector <vector<int>> mat (REN,vector<int>(COL));  
    for (int i = 0; i < REN; i++) {  
        for (int j = 0; j < COL; j++) {  
            cin >> mat[i][j];  
        }  
    }  
    for (int i = 0; i < REN; i++) {  
        for (int j = 0; j < COL; j++) {  
            cout << mat[i][j] << " ";  
        };cout << endl;  
    }  
    return 0;  
}
```

**¿Qué es un iterador?**



Los iteradores son un objeto de c++ que generalizan punteros de manera que permite acceder y trabajar con diferentes estructuras de datos y rangos de estos, hay cinco principales tipos de iteradores: Forward, Bidirectional, Output, Input y Random access. Los primeros dos se usan como una alternativa a los índices de los ciclos for.

Ejemplo de un ciclo for regular:

```
int main() {  
    vector<int> numeros = {1, 2, 3, 4, 5};  
    for(size_t i=0; i<numeros.size(); i++){  
        cout<<i<<" ";  
    }  
    return 0;  
}
```

# Iterador implícito de rango (Forward)

```
int main() {  
    vector<int> numeros = {1, 2, 3, 4, 5};  
    for(int numero : numeros){// usando rangos de iteradores  
        cout<<numero<<" ";  
    }  
    return 0;  
}
```

# Iteradores explícitos de vectores STD (Bidirectional)

```
int main() {  
    vector<int> numeros = {1, 2, 3, 4, 5};  
    for(vector<int>::iterator it=numeros.begin(); it!=numeros.end(); it++){  
        cout<<*it<<" ";  
    }  
    return 0;  
}
```

# Iterador de tipo automatico (Forward)

```
int main() {  
    vector<int> numeros = {1, 2, 3, 4, 5};  
    for(auto it=numeros.begin(); it!=numeros.end(); it++){  
        cout<<*it<<" ";  
    }  
    return 0;  
}
```

# Problemas

Resolver los siguientes problemas usando memoria dinamica, punteros y logica de punteros para la impresion de datos.

344A: Magnets

263A: Beutiful Matrix

# Referencias

[https://yewtu.be/watch?v=2ybLD6\\_2gKM](https://yewtu.be/watch?v=2ybLD6_2gKM)

[https://www.w3schools.com/cpp/cpp\\_pointers.asp](https://www.w3schools.com/cpp/cpp_pointers.asp)

<https://www.geeksforgeeks.org/cpp-pointers/>

<https://www.geeksforgeeks.org/new-and-delete-operators-in-cpp-for-dynamic-memory/>

<https://www.geeksforgeeks.org/introduction-iterators-c/>

<https://en.cppreference.com/w/cpp/iterator>