Reporte de Laboratorio 1
Punteros y funciones en C++

Dunia Barahona - B40806

11 de septiembre de 2016

Índice

---

1. Código

A continuación se mencionan las funciones del programa.

- `cont`: obtiene el tamaño del código genético (ARN) ingresado desde la línea de comandos.

```
int cont (char **bss)
{
    int n = 0;
    while (bss[1][n] != '\0')
    {
        n++;
    }
    return n;
}
```

Figura 1: función `cont`

- `amino`: determina cual aminoácido corresponde al codón de ARN ingresado según la tabla de conversión ARN a aminoácido.

```
char amino(int p1, int p2, int p3)
{
    char **comb[4];
    comb[0] = new char*[4];
    comb[1] = new char*[4];
    comb[2] = new char*[4];
    comb[3] = new char*[4];

    comb[0][0] = new char[4];
    comb[0][1] = new char[4];
    comb[0][2] = new char[4];
    comb[0][3] = new char[4];

    comb[1][0] = new char[4];
    comb[1][1] = new char[4];
    comb[1][2] = new char[4];
    comb[1][3] = new char[4];

    comb[2][0] = new char[4];
    comb[2][1] = new char[4];
    comb[2][2] = new char[4];
    comb[2][3] = new char[4];

    comb[3][0] = new char[4];
    comb[3][1] = new char[4];
    comb[3][2] = new char[4];
    comb[3][3] = new char[4];
```

Figura 2: tabla de conversión ARN-aminoácido

```
    comb[0][0][0] = 'K';
    comb[0][0][1] = 'N';
    comb[0][0][2] = 'K';
    comb[0][0][3] = 'N';

    comb[0][1][0] = 'T';
    comb[0][1][1] = 'T';
    comb[0][1][2] = 'T';
    comb[0][1][3] = 'T';

    comb[0][2][0] = 'R';
    comb[0][2][1] = 'S';
    comb[0][2][2] = 'R';
    comb[0][2][3] = 'S';

    comb[0][3][0] = 'I';
    comb[0][3][1] = 'I';
    comb[0][3][2] = 'M';
    comb[0][3][3] = 'I';
```

Figura 3: tabla de conversión ARN-aminoácido

```
comb[1][0][0] = 'Q';
comb[1][0][1] = 'H';
comb[1][0][2] = 'Q';
comb[1][0][3] = 'H';

comb[1][1][0] = 'P';
comb[1][1][1] = 'P';
comb[1][1][2] = 'P';
comb[1][1][3] = 'P';

comb[1][2][0] = 'R';
comb[1][2][1] = 'R';
comb[1][2][2] = 'R';
comb[1][2][3] = 'R';

comb[1][3][0] = 'L';
comb[1][3][1] = 'L';
comb[1][3][2] = 'L';
comb[1][3][3] = 'L';
```

Figura 4: tabla de conversión ARN-aminoácido

```
comb[2][0][0] = 'E';
comb[2][0][1] = 'D';
comb[2][0][2] = 'E';
comb[2][0][3] = 'D';

comb[2][1][0] = 'A';
comb[2][1][1] = 'A';
comb[2][1][2] = 'A';
comb[2][1][3] = 'A';

comb[2][2][0] = 'G';
comb[2][2][1] = 'G';
comb[2][2][2] = 'G';
comb[2][2][3] = 'G';

comb[2][3][0] = 'V';
comb[2][3][1] = 'V';
comb[2][3][2] = 'V';
comb[2][3][3] = 'V';
```

Figura 5: tabla de conversión ARN-aminoácido

```
    comb[3][0][0] = '|';
    comb[3][0][1] = 'Y';
    comb[3][0][2] = '|';
    comb[3][0][3] = 'Y';

    comb[3][1][0] = 'S';
    comb[3][1][1] = 'S';
    comb[3][1][2] = 'S';
    comb[3][1][3] = 'S';

    comb[3][2][0] = '|';
    comb[3][2][1] = 'C';
    comb[3][2][2] = 'W';
    comb[3][2][3] = 'C';

    comb[3][3][0] = 'L';
    comb[3][3][1] = 'F';
    comb[3][3][2] = 'L';
    comb[3][3][3] = 'F';

    char pt = comb[p1][p2][p3];

    delete[] *(comb);

    return pt;
}
```

Figura 6: tabla de conversión ARN-aminoácido

- `triplete`: traduce las bases del codón recibido según lo siguiente: A= 0, C= 1, G= 2, U= 3.

```
char triplete (char *bss)
{
    int* temp= new int[3];      // arreglo temporal
    for (int i = 0; i < 3; i++)
    {
        switch (*(bss+i))
        {
            case 'A':
                *(temp+i) = 0;
                break;
            case 'C':
                *(temp+i) = 1;
                break;
            case 'G':
                *(temp+i) = 2;
                break;
            case 'U':
                *(temp+i) = 3;
                break;
            default:
                *(temp+i) = -1;
                break;
        }
    }
    char AA = amino(temp[0], temp[1], temp[2]);
    delete[] temp;
    return AA;
}
```

Figura 7: función `triplete`

- `insp`: verifica si el ARN ingresado cumple con las siguientes condiciones:

  1. La cantidad de bases que lo componen es multiplo de 3.
  2. Está compuesto únicamente por los caracteres que corresponden a bases nitrogenadas (A, C, G, U).

```
int insp(char **bss)
{
    int swch= 0;
    int error = 0;
    int n = cont(bss);

    int resto = n % 3;
    if (resto!= 0)
    {
        swch= 1;
        cout << "La serie ingresada no es multiplo de 3" << endl;
    }
    bss++;
    for (int i = 0; i < n; i++)
    {
        char tp = *(*bss+i);
        if (tp!='A' && tp!='C' && tp!='G' && tp!='U')
        {
            error= 1;
            swch= 1;
        }
    }
    if (error != 0)
    {
        cout << "La serie ingresada contiene caracteres que no corresponden a las bases nitrogenadas (A, G, C, U)" << endl;
    }
    return swch;
}
```

Figura 8: función `insp`

- `traducirARNaAA`: tiene como parámetro un puntero de char `char** arn` el cual contiene la dirección de memoria del ARN ingresado desde la línea de comandos y retorna un nuevo arreglo de char con los animoácidos resultantes.

```
char* traducirARNaAA(char **arn)
{
    int n = cont(arn);
    char* pt = new char[n/3];
    int swch= insp(arn);
    if (swch==0)
    {
        char* temp = new char[3];
        char amac;

        int m= 0;


        for (int i = 0; i < n; i++)
        {
            *temp = *(*(arn+1)+i);
            *(temp+1) = *(*(arn+1)+i+1);    // Es lo mismo que: temp[1]= bss[1][i+1];
            *(temp+2) = *(*(arn+1)+i+2);

            amac = triplete(temp);
            *(pt+m)= amac;
            m++;
            //cout<< *(pt+m) << endl;
            i+=2;
        }
        delete temp;
    }
    return pt;
}
```

Figura 9: función `traducirARNaAA`

- `imprimirArregloDeChar`: recibe un puntero de char `char* a` y un entero `int n`, e imprime la cantidad especificada de caracteres del puntero.

```
int imprimirArregloDeChar(char* a, int n)
{
    for (int i = 0; i < n; i++)
    {
        cout << *(a+i);
    }
    cout << endl;

    return 0;
}
```

Figura 10: función `imprimirArregloDeChar`

2. Conclusiones

- Cuando no se sabe de antemano la cantidad de elementos con los que el programa va a trabajar se hace uso de la memoria dinámica.
- Es responsabilidad del programador liberar el espacio de memoria que se reservó a la hora de crear punteros.
- Usando punteros se pueden crear arreglos dinámicos, definiendo el tamaño mientras se ejecuta el programa.