Basic ++

Quick Reference Manual

Creación del programa

Sintaxis

```
program <nombre>;

<declaración de variables globales>
<definición de funciones>

main ()
<declaración de variables locales>
{
      <estatutos>
}
```

Nombre

Los nombres son identificadores que empiezan con una letra seguida de una secuencia de números, letras o guiones bajos.

```
id | var1 | loop_value
```

Declaración de variables (Global o Local)

Sintaxis

```
var
      <tipo> : <lista de ids>;
      <tipo> : <lista de ids>;
      …
```

Tipo

Puede ser cualquiera de estas 3 categorías:

```
int | float | char
```

Lista de ids

Identificadores separados por comas, pudiendo definir una o dos dimensiones con números enteros mayores a 0

```
id | id[10] | id [3][5]
```

Declaración de funciones

Sintaxis

```
<tipo de retorno> func <nombre> ( <parámetros> )
<declaración de variables locales>
{
      <estatutos>
}
```

Tipo de retorno

Puede ser cualquiera de estas 4 categorías:

```
int | float | char | void
```

En el caso de void es cuando una función no regresa ningún valor

Parámetros

Sintaxis

```
<tipo> : <nombre> , <tipo> : <nombre>, …
```

Las funciones pueden recibir 0 o más parámetros, siendo declarados con el tipo seguido del nombre.

Estatutos

Asignación

Sintaxis

```
<id> = <expresión>;
<id> = &<nombre función>(<argumentos>);
```

Llamada a función

Sintaxis

```
&<nombre función>(<argumentos>);
```

Retorno de función

```
return(<expresión>);
```

Lectura

Sintaxis

```
read(<lista de ids>);
```

Escritura

Sintaxis

```
print(<lista de letreros o expresiones>);
```

Letrero

Hace referencia a un string, se debe escribir con comilla doble

```
"string"
```

Estatuto de decisión

Sintaxis

```
if(<expresión>){
      <estatutos>
}
else {
      <estatutos>
}
```

Estatutos de repetición

Condicional

Sintaxis

```
while(<expresión>) do {
      <estatutos>
}
```

No condicional

Sintaxis

```
for <id> = <expresión> to <expresión> do {
      <estatutos>
}
```

Expresión

Las expresiones en Basic++ son las tradicionales (como en C y en Java). Existen los operadores aritméticos, lógicos y relacionales: +, -, *, /, % , &&(and) , || (or),< , >, <=, >=, ==, !=. Se manejan las prioridades tradicionales, se pueden emplear paréntesis para alterarla.

Ejemplo de un programa completo

```
program Ejemplo;
var
    int i, j, p;
    int Arreglo[12];
    int Matriz[12][9];

func int fact (int j)
var int i;
{
    i = j + (p - j*2+j) ;
    print(i, j);
    if (j <= 1)
        { return ( j ); }
    else
        { return ( j * (&fact(j - 1))); }
}

func void inicia (int y)
var int x;
{   x = -1;
    x = x + 1;
    while ( x < 11) do {
        Arreglo[x] = y * x;
        x = x + 1;
    }
}

main ( )
{
    read (p) ;
    j = p *2;
    i = -1;
    i = i + 1;
    &inicia(p*j - 5);

    while ( i < 11) do
    {
        Arreglo[i] = Arreglo[i] * &fact(Arreglo[i] - p);
        Matriz[i][(Arreglo[i] * -1) % p] = Arreglo[i];
        i = i + 1;
    }

    i = i - 1;

    while ( i >= 0) do
    {
        print ("resultado", Arreglo [ i ] , &fact( i +2)) ;
        print ("mat", Matriz[i][0]) ;
        i = i - 1;
    }
}
```