Presentacion

Holy C, como su nombre sugiere, es un lenguaje de programación al estilo de *C* con varias diferencias y mejoras clave. Al igual que *C*, no depende de espacios en blanco y se compila a ensamblador.

Algunas diferencias

HolyC **no** requiere una función `Main()`. Las expresiones fuera de las funciones se evalúan simplemente de arriba a abajo en el código fuente. Esto permite que el lenguaje actúe como un shell y, de hecho, **es** el shell de TempleOS. Las funciones que se llaman sin argumentos pueden acortarse sintácticamente escribiendo solo el nombre de la función seguido de un punto y coma.

```
// Las siguientes son equivalentes.

x = Foo();  // C
y = Foo;    // HolyC
```

Terry explicó en varias ocasiones cómo las declaraciones de *switch* son las construcciones más potentes en *HolyC*. En este lenguaje, las declaraciones de *switch* siempre utilizan tablas de saltos en ensamblador (y, por lo tanto, se menciona en la documentación que no se deben usar en casos con rangos de valores grandes o dispersos).

Las de *HolyC* ofrecen una amplia gama de mejoras de conveniencia en comparación con sus contrapartes en *C*.

```
I64 i;
switch (i) {
    case: "zero\n"; break;          //
Declaraciones de caso implícitas comienzan en 0
    case: "one\n"; break;           // ... e
incrementan de 1 en 1 cada vez.
    case: "two\n"; break;
    case 3: "three\n"; break;       // Casos
explícitos funcionan como se esperaría.
    case 4...8: "others\n"; break; // Casos del
4 al 8 imprimirán "others\n".
}
```

Las expresiones de los switch pueden estar anidadas en lo que se conocen como declaraciones "sub_switch" mediante las palabras clave `start` y `end`.

```
U0 SubSwitch ()
{
    I64 i;
    for (i=0;i<10;i++)
        switch(i) {
            case 0: "Zero ";     break;
            case 2: "Two ";      break;
            case 4: "Four ";     break;
```

```
            start:
                "[";
                case 1: "One";   break;
                case 3: "Three"; break;
                case 5: "Five";  break;
            end:
                "] ";
                break;
        }
    '\n';
}
SubSwitch;
```

Que regresara Zero [One] Two [Three] Four [Five]

Las constantes de tipo string se enviaran automaticamente a imprimir

```
U0 PrintMessage(char *first, char *last)
{
    "Hello person!\n";
    "Your name is %s %s.\n", first, last;
}
```

bibliografia

A Language Design Analysis of HolyC - Harrison Totty. (2019, 15 marzo). https://harrison.totty.dev/p/a-lang-design-analysis-of-

holyc#:~:text=HolyC%2C%20as%20the%20name%20would,TempleOS%20uses%20nonstandard%20opcodes.