Łukasz Główka, 313803

Program dekodujący kody kreskowe zgodnie ze specyfikacją Code 128C.

# Struktura programu i danych

Program posiada następujące etapy:

1. Odczytanie pliku w C++.
2. Znalezienie najmniejszej kreski.
3. Znalezienie pierwszej pustej przestrzeni i znaku startu.
4. Znalezienie wzoru, zdekodowanie, powrót do szukania wzorów.
5. Zdekodowanie znaku stopu, sprawdzenie sumy kontrolnej i pustej przestrzeni końcowej.
6. Wypisanie wyniku na standardowym wyjściu.

Program korzysta z pliku constants.asm w którym zapisana jest tabela kodów odpowiadające odpowiednim wzorom, które w postaci binarnej posiadają jedynki w miejscu czarnej kreski i zera w miejscu białej kreski.

Zdekodowane wzory program przechowuje na stosie, a pod koniec programu dekoduje, oblicza sumę kontrolną i porównuje ją z odczytaną sumą.

# Implementacja

Program zaczyna się od przeczytania pliku bmp oraz zweryfikowaniu jego poprawności w pliku C++. W pliku asm program rezerwuje potrzebną na stosie pamięć. Następnie program oblicza najmniejszą kreskę przez przesunięcie logiczne w prawo (podzielenie na 2) pierwszej czarnej kreski, która w każdym kodzie jest równa długości dwa. Po obliczeniu najmniejszej długości weryfikowane jest pojawienie się pustej przestrzeni wymaganej przed kodem.

Następna część kodu znajduje ciąg sześciu kresek czarnych i białych – gdy natrafi na zmianę koloru, dodaje kreskę do rejestru jako logiczne przesunięcie w lewo oraz dodanie jedynki, a białą kreskę jako przesunięcie logiczne w lewo bez dodania jedynki. Po dodaniu całego ciągu powstaje binarna liczba mówiąca który kod został wczytany. W programie pokazywane są jako liczby szesnastkowe i tak są one zapisane w pliku data.asm.

Powstała z ciągu liczba jest porównywana z tymi w pliku. Po znalezieniu liczby jest ona dekodowana przy pomocy licznika, którym iterujemy po kodach z pliku constants.asm. Następnie liczba dodawana jest do rejestru obliczającego sumę kontrolną. Po wykryciu pierwszego ciągu sprawdzany jest znak startu.

W przypadku podejrzenia wykrycia znaku stop liczba jest sprawdzana pod kątem zgodności ze znakiem stopu.

Po zdekodowaniu znaku stopu program sprawdza sumę kontrolną porównując ją z sumą obliczoną z poprzednio wczytanych kodów. Następnie sprawdzana jest wymagana pusta przestrzeń na końcu programu.

Wynik działania programu jest wypisywany na standardowym wyjściu.

# Testowanie

Aby odtworzyć testowane przypadki należy uruchomić podane pliki dla 24. linijki.

Przy błędzie odczytu pliku pojawi się błąd "File incorrect", a przy podaniu złej linijki "Skanned line must be greater than 1 and lower than height - 1.". Brak podania linijki również skutkuje błędem "You must pass 2 arguments".

Błędy:

|  |  |  |
| --- | --- | --- |
| PLIK | BŁĄD | POWÓD |
| error\_wrong\_checksum.bmp | Wrong checksum error | Znak kontrolny nie jest prawidłowy. |
| error\_out\_of\_range.bmp  error\_out\_of\_range\_black.bmp  error\_out\_of\_range\_white.bmp | Out of range error | Błędny wzór. |
| error\_wrong\_start\_code.bmp | Wrong pattern error | Zły kod startu. |
| error\_no\_spaces\_after\_code.bmp  error\_no\_spaces\_before\_code.bmp | Wrong space error | Brak pustej przestrzeni przed/po kodzie. |
| error\_wrong\_stop.bmp | Wrong stop error | Zły kod stopu. |
| Abc.s | Format of file is wrong | Nazwa pliku się nie zgadza. |

Pliki kod.bmp, kod2.bmp, kod3.bmp są poprawne.