Temat: Arytmetyka modularna

Pierwszy raport z postępu prac

Tak, jak zostało wcześniej ustalone, nasz projekt został napisany w języku C++. Poszczególne funkcje implementujące algorytmy zostały napisane w języku asembler i wstawione do programu.

Program, który obecnie udało nam się stworzyć składa się między innymi z pliku *main.cpp*, w którym znajduje się wywołanie funkcji wyświetlania menu głównego programu oraz logika odpowiedzialna za wybieranie określonego algorytmu. Zostało to zaimplementowano z używając switch`a.

Pierwszą implementacją w programie jest wyznaczanie NWD za pomocą algorytmu Euklidesa. Aby zobrazować działanie algorytmu, załóżmy że należy wyznaczyć NWD z liczb *a* oraz *b*. Na początku wykonywane jest dzielenie z resztą liczby *a* przez liczbę *b*. Jest to realizowane za pomocą instrukcji *div* w języku assembler. Gdy reszta z dzielenia, która jest umieszczona w rejestrze %edx wynosi 0 to największym wspólnym dzielnikiem jest liczba *b*. W przypadku gdy reszta jest różna od zera to następuje przypisanie liczbie *a* wartości liczby *b*. Następnie liczbie *b* jest przypisywana wartość reszty. Ponownie jest realizowane dzielenie liczby *a* przez *b*, aż reszta nie będzie równa zero. Cała operacja jest realizowana w pętli while, poprzez użycie instrukcji *cmp*. Implementacja algorytmu znajduje się w pliku *Euklides.cpp*. Dodatkowo znajduje się tam również funkcja wyświetlająca menu oraz pobierająca dane od użytkownika.

Kolejnym algorytmem, który został zaimplementowany jest naiwny algorytm wyznaczania pierwszości liczby. Algorytm polega na próbnym dzieleniu sprawdzanej liczby *a* przez liczby z zakresu od *2* do *sqrt(a)*. Przy każdej takiej operacji badana jest reszta z dzielenia. Jeśli reszta podczas którejś operacji dzielenia będzie wynosiła 0, to liczba *a* nie będzie liczbą pierwszą. Interesujące może się wydawać, dlaczego wystarczy sprawdzić liczby z podanego wyżej zakresu. Dzieje się tak, ponieważ jeśli liczba posiada czynnik większy od *sqrt(a)*, to drugi jego czynnik musi być mniejszy od pierwiastka z *a*, aby ich iloczyn musiał być równy *a*. Zatem wystarczy podzielić liczbę *a* przez liczby z danego przedziału, aby wykluczyć liczby złożone. Tak jak zostało wcześniej założone, algorytm został napisany w języku asembler. Z powodu braku dobrej znajomości tego języka, pierwiastkowanie liczby *a* zostało wykonane w języku *C++.* Wynik tej operacji został jednak wykorzystany bezpośrednio we wstawionym fragmencie z kodem algorytmu. Algorytm zwraca podmienioną liczbę *a*, w zależności od tego czy jest pierwsza, czy nie. Jeśli liczba *a* była pierwsza, to zostanie zwrócone *1*, a jeśli nie była liczbą pierwszą to zwróci *0*. Zostało to wykorzystane w wywołaniu algorytmu. W pliku *NaivePrime.cpp*, w którym znajduje się implementacja algorytmu, znajduje się również funkcja odpowiedzialna za wczytywanie liczby podanej obserwacji. Wykorzystując zaimplementowany algorytm w funkcji *checkPrime()* pokazuje ona, czy dana liczba jest liczbą pierwszą wyświetlając stosowną informację.

W następnym etapie projektu mamy zamiar zaimplementować kolejne algorytmy. Będą one bardziej złożone co powoduje że napisanie pełnych funkcji w języku assembler może być problematyczne. Realizowany jednak projekt pozwala na poznanie praktyczne assemblera, a więc będziemy dalej próbować implementować poszczególne funkcje przy jego użyciu.