student: Imię Nazwisko (GRUPA)
student@elka.pw.edu.pl
prowadzący: Imię Nazwisko

Podstawy Programowania (PRM) — projekt
dokumentacja projektu

Temat: Znajdowanie wszystkich liczb pierwszych zawartych w pewnym przedziale.

Modyfikacje w stosunku do zakładanej funkcjonalności programu W opracowanej wersji, dla każdej liczby nie będącej pierwszą program przedstawia listę wszystkich liczb pierwszych, które są jej dzielnikami.

Algorytm rozwiązania zadania

Do rozwiązania zadania wykorzystano metodę sita Eratostenesa, opisaną w specyfikacji technicznej. Po modyfikacji założeń, algorytm wygląda następująco. W metodzie rozważane są kolejne liczby od 2 do ograniczenia górnego N. Każda liczba z przedziału $[2, N]$ ma status:

0 do rozważenia

1 pierwsza

2 nie pierwsza

oraz listę liczb pierwszych będących jej dzielnikami.
Algorytm działa w sposób następujący:

procedura sito Eratostenesa
dane wejściowe M (ograniczenie dolne), N (ograniczenie górne)
{
  dla każdego $i \in [2, N]$
    $status(i) = 0$
    $dzielniki(i) = \emptyset$
  dla każdego $i \in [2, N]$
    jeśli $status(i) == 0$
      dla każdego $j \in [i, N]$
        $status(j) = 2$
        $dzielniki(j) = dzielniki(j) \cup \{i\}$
  dla każdego $i \in [M, N]$
    jeśli $status(i) == 1$
      $wypisz(i)$
    w przeciwnym przypadku
      $wypisz(dzielniki(i))$
}

Struktury danych

Program będzie się posługiwał główną tablicą zawierającą $(N - M + 1)$ wskaźników na struktury, w których będzie przechowywana wartość statusu liczby oraz wskazanie na listę dzielników. Lista dzielników będzie zrealizowana za pomocą struktur zawierających wartość dzielnika oraz wskazanie na następny dzielnik. Dołączanie elementów będzie zawsze na początek listy, dlatego też po zakończeniu działania sita Eratostenesa lista dzielników będzie uporządkowana w kolejności malejącej.

Element o indeksie i głównej tablicy (numeracja od zera) będzie odpowiadał liczbie i+M.

Własne typy danych

```
struct kostka{
int wartosc;
struct kostka* nast;
};
```

Typ strukturalny odpowiadający zarówno elementowi tablicy liczb, jak i elementowi listy dzielników

Podział na funkcje

- `int wykonajSito(struct kostka *liczby[], int M, int N);`

  Funkcja wykonująca obliczenia wg algorytmu Eratostenesa. Przyjmuje następujące argumenty

  - `struct kostka* liczby[]` Tablica zawierająca wskaźniki do struktur opisujących poszczególne liczby; konwencja numeracji elementów tablicy opisana wyżej. Zakłada się, że początkowo wszystkie liczby mają puste listy pierwszych dzielników
  - `int M, int N` Ograniczenia: dolne i górne przedziału poszukiwań

  Wartością zwracaną jest liczba różnych liczb pierwszych z zakresu $[M, N]$ (0 oznacza brak).

- `struct kostka* dodaj(struct kostka *glowa, int wartosc)`

  Funkcja tworzy kostkę typu `struct kostka`, w polu `wartosc` wpisuje wartość argumentu `wartosc`, dodaje tę kostkę na początek listy `struct kostka* glowa` i zwraca nową listę z dodanym elementem jako pierwszym.

- `void wypiszListe(struct kostka* lista)`

  Na standardowe wyjście wyprowadza pola `wartosc` kolejnych elementów listy, rozpoczynając od głowy.

- `void wypiszSito (struct kostka* sito[], int n)`

  Na standardowe wyjście wyprowadza informacje o pierwszości lub podzielnikach liczb, przechowywaną w sicie Eratostenesa (argument `sito`); `n` jest liczbą elementów sita.

- `int czytNat(char* zacheta);`

  Funkcja wyprowadza na standardowe wyjście napis będący argumentem `char* zacheta`, a następnie wczytuje kolejne łańcuchy znakowe ze standardowego wejścia, dopóki nie dadzą się zintepretować jako liczba naturalna w formacie dziesiętnym. Wartość tej liczby jest zwracana jako wynik funkcji.

- `int przygotuj(int M, int N);`

  Funkcja przygotowująca dane dla funkcji `sito`. Przechowuje tablicę zawierającą dane liczb. Wykonywane są kolejno następujące czynności:

  1. jeśli wartości M, N są ujemne, są one wczytywane funkcją czytNat; jeśli tylko M lub tylko N jest ujemna, użytkownik jest proszony o podanie brakującej wartości;
  2. ewentualna zamiana wartości M, N, jeśli $M > N$;
  3. alokacja tablicy status o $M - N + 1$ elementach; w przypadku niepowodzenia, funkcja wyprowadza na standardowy strumień błędów stosowny komunikat i zwraca wartość -1;
  4. alokacja struktur wskazywanych przez każdy z elementów tablicy
  5. wywołanie funkcji `sito`;
  6. wypisanie rozwiązań;
  7. zwolnienie pamięcii (zwalniane są listy dzielników, kostki na liczby oraz sama tablica);
  8. zwrócenie wartości 0.

- `int main(int argc, char* argv[]);`

  Funkcja główna. Wywołuje funkcję `przygotuj`:

  1. jeśli program został wywołany bez argumentów, przekazywane są wartości M, N równe -1
  2. jeśli został podany jeden argument wywołania, jest on wartością M, a wartością N jest -1;
  3. jeśli zostały podane dwa argumenty lub więcej, pierwsze dwa są traktowane jako wartości M, N, a pozostałe są odrzucane;
  4. w przypadku gdy argument nie daje się zinterpretować jako liczba całkowita, przyjmowana jest jego wartość równa -1.

  Zwraca wartość zwracaną przez `przygotuj`.

Sposób testowania Program był testowany dla różnych wartości M, N. Stwierdzono, że wartość M ma wpływ na szybkość działania programu (liniowy wzrost liczby obliczeń wraz ze wzrostem M), natomiast różnica N-M wpływa na zajętość pamięci operacyjnej (kwadratrowa zależność pamięci od różnicy N-M). Największe wartości testowane to: N-M=1000, M=1000.

Argumenty wywołania nie będące liczbami naturalnymi są interpretowane przez program jako wartość 0. Przy wywołaniu bezargumentowym, pobierana wartość jest czytana dopóty, dopóki nie da się zinterpretować jako liczba naturalna.

Literatura

1. Wirth, N. *Algorytmy + struktury danych = programy.*
2. Kernighan, B., Ritche, D. *Język ANSI C*
3. Silvester, P. *System operacyjny unix.*