Sprawozdanie

Politechnika Wrocławska	Wydział Informatyki i Zarządzania				
	Laboratorium Programowania Równoległego i Rozproszonego				
Kierunek:	Informatyka	Rok studiów nr:	2	Semestr nr:	3
Rok akademicki:	2016/2017	Grupa administracyjna:	32b	Grupa ćwiczeniowa:	32b-g1

SPRAWOZDANIE

Nr ćwiczenia	Temat ćwiczenia
2b	Kompilowanie i uruchamianie programów testowych w języku C
Termin złożenia sprawozdania	
20.3.2017	
Data faktycznego złożenia sprawozdania	

Wykonawcy	Nazwisko	Imię	Nr indeksu	Ocena
	Arciszewski	Jacek	200412	
	Buczel	Kamil	200420	

Data i podpis prowadzącego ćwiczenia

1. Temat ćwiczenia

Tematem wykonywanego ćwiczenia było kompilowanie i uruchamianie programów testowych w języku C.

2. Zakres ćwiczenia

Zakresem ćwiczenia jest zapoznanie się oraz modyfikacja programów napisanych w języku programowania C.

Pierwszy program realizował algorytm realizujący tzw. Naiwny algorytm poszukiwania wzorca w tekście.

Algorytm, który wykonywał drugi program używany był do sprawdzania, czy podana wartość całkowita jest liczbą pierwszą.

3. Środowisko realizacji ćwiczenia

Sprzęt:

- Procesor: AMD A6-3420M APU 1.50 GHz
- Pamięć RAM: 6,00 GB

Ćwiczenie realizowane było na systemie operacyjnym Windows 8.1 oraz na podanym przez prowadzącego zdalnym serwerze pod adresem domenowym g133.pl.

W celu ułatwienia pracy nad skryptami pisanymi w języku C, posłużono się zintegrowanym środowiskiem programistycznym Netbeans 8.1, pozwalającym na szybką poprawę błędów i kompilację programów w docelowym kompilatorze GCC.

Do łączenia się z serwerem użyto programu PuTTY, natomiast do przenoszenia plików między lokalnym i zdalnym hostem użyto oprogramowania WinSCP. W celu rejestracji przebiegu sesji terminalowej wykorzystano opcje programu PuTTY w sposób podany przez prowadzącego laboratoria.

W celu zapoznania się z naiwnym algorytmem poszukiwania wzorca, posłużono się opisem znajdującym się na stronie *http://www.algorytm.org/przetwarzanie-tekstu/algorytm-n-naiwny.html* .

4. Przebieg ćwiczenia i uzyskane wyniki

4.1 Zadanie nr 1

4.1.1 Treść polecenia

Według przedstawionych podpunktów w celu wykonania ćwiczenia należało pobrać do katalogu prywatnego program `Algorytm-N.c`, realizujący tzw. naiwny algorytm poszukiwania wzorca w tekście. Po zapoznaniu się z ideą algorytmu w źródłach online, należało się zapoznać się z samym programem, skompilować go kompilatorem `gcc` i uruchomić plik wykonywalny.

Następne czynności związane były z modyfikacją kodu. Należało zapewnić funkcjonalność umożliwiającą:

- wczytywanie tekstu z pliku, który zapisywany będzie do odpowiedniej zmiennej
- wyprowadzenie długości wczytanego tekstu oraz długości wzorca

Po modyfikacji programu należało go przetestować. W tym celu trzeba było stworzyć plik zawierający minimalnie 500 znaków alfabetycznych oraz spacji. Treść tego pliku należało wczytać do programu i odnaleźć wzór zawierający przynajmniej 8 znaków, podany z klawiatury użytkownika.

4.1.2 Cel czynności

Celem czynności było zrozumienie idei użytego algorytmu, a także przypomnienie sposobu kompilacji i uruchamiania skryptów napisanych w języku programowania C. Zadanie kładło również nacisk na modyfikację kodu o dodatkowe funkcje, co wiąże się z umiejętnością korzystania z dokumentacji.

4.1.3 Sposób wykonania i rezultaty

Treść ściągniętego programu `Algorytm-N.c` przed modyfikacją została zaprezentowana poniżej.

```
#include <stdio.h>
#include <string.h>

int main(void)
{
    char wzorzec[1000];
    char tekst[100];
    int m,n,i,j;
    printf("Podaj tekst\n");
    scanf("%s", tekst);
    printf("Podaj wzorzec\n");
    scanf("%s", wzorzec);
    n=strlen(tekst);
    m=strlen(wzorzec);
    printf("Indeksy wystapien wzorca w tekscie\n");
    i=0;
    while (i<=n-m)
    {
     j=0;
     while ((j<m)&&(wzorzec[j]==tekst[i+j])) j++;
     if (j==m) printf("%d\n", i+1);
     i++;
    }
    getchar();
}
```

Z ideą naiwnego algorytmu wyszukiwania wzorca w tekście zapoznano się na stronie znajdującej się w komentarzu programu: *http://www.algorytm.org/przetwarzanie-tekstu/algorytm-n-naiwny.html* . Kompilacja oraz uruchomienie pliku wykonywalnego programu zostało zaprezentowane poniżej.

```
[32b-g1@intek Z1]$ gcc Algorytm-N.c
[32b-g1@intek Z1]$ ./a.out
Podaj tekst
programowanie_rownolegle_i_rozproszone
Podaj wzorzec
ro
Indeksy wystapien wzorca w tekscie
2
15
28
32
```

Według zaleceń znajdujących się w instrukcji do ćwiczenia, w kodzie dokonano odpowiednich modyfikacji. Usunięto funkcję wczytywania treści tekstu z klawiatury użytkownika, zamiast tego dodano możliwość wczytywania tekstu z pliku za pomocą funkcji `fopen`. Dodatkowo, dodano wyświetlanie długości wczytanego tekstu oraz wzorca.

Zmodyfikowany kod znajduje się poniżej.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main(void)
```

```
{
    FILE *fp;
    char *bufor = NULL;

    size_t dlugoscPliku;
    int dlugoscWzorca;

    char wzorzec[1000];
    char sciezkaPliku[1000];
    int m,n,i,j;

    printf("Podaj sciezke do pliku z tekstem: ");
    scanf("%s", sciezkaPliku);

    fp = fopen(sciezkaPliku, "r");

    if(fp == NULL)
    {
            perror("Blad podczas otwierania pliku.\n");
            exit(EXIT_FAILURE);
    }

    fseek(fp, 0, SEEK_END);
    dlugoscPliku = ftell(fp);
    rewind(fp);

    bufor = malloc((dlugoscPliku + 1) * sizeof(*bufor));
    fread(bufor, dlugoscPliku, 1, fp);

    fclose(fp);

    printf("Podaj wzorzec: ");
    scanf("%s", wzorzec);

    dlugoscWzorca = strlen(wzorzec);

    printf("\nDlugosc pliku: %zu\n", dlugoscPliku);
    printf("Dlugosc wzorca: %i\n\n", dlugoscWzorca);

    n = dlugoscPliku;
    m = dlugoscWzorca;

    printf("Indeksy wystapien wzorca w tekscie\n");
    i = 0;

    while (i<=n-m)
    {
            j=0;
            while ((j<m)&&(wzorzec[j]==bufor[i+j])) j++;
            if (j==m) printf("%d\n", i+1);
            i++;
    }

    getchar();
}
```

Zmodyfikowany kod został przetestowany na niewielkich danych, w międzyczasie zostały poprawione błędy, zarówno na etapie kompilacji jak i błędy logiczne.

W celu sprawdzenia poprawności działania programu, należało wykonać obliczenia kontrolne za pomocą specjalnie przygotowanego pliku, zawierającego przynajmniej 500 znaków alfabetycznych oraz spacji. W naszym przypadku był to tekst angielskiej rozprawki na temat Hamleta. Wydruk tego tekstu został przedstawiony poniżej.

```
What is mankind? Who am I? What is the meaning of life? These are multifa-
ceted existential questions that ancient and modern philosophies have yet
to adequately answer. Countless philosophers have spent their lifetimes in
search of answers to these questions but died before finding a suitable
answer. Certainly, the philosophy of existentialism is an interesting phe-
nomenon. The dictionary defines existentialism as a "philosophical move-
ment . . . centering on analysis of individual existence in an unfathoma-
ble universe and the plight of the individual who must assume ultimate
responsibility for acts of free will" ("Existentialism"). The character
Hamlet from Shakespeare's tragedy Hamlet explores these existential que-
stions, seeking truth and understanding as he tries to come to grips with
his father's death. In the end, Hamlet proves to be an exceedingly exi-
stential character. Prince Hamlet is a university student who enjoys con-
templating difficult philosophical questions. When his father, king of
Denmark, dies, he returns home to find evidence of foul play in his fa-
ther's death. The Ghost of Hamlet (the dead king) tells Prince Hamlet that
his uncle Claudius is the murderer. Throughout the rest of the play, Ham-
let seeks to prove Claudius' guilt before he takes action against Clau-
dius. However, Hamlet is pensive ad extremum, at times even brooding; he
constantly overuses his intellect while ignoring his emotions and ignoring
what "feels right." His extreme logic causes him to delay his revenge aga-
inst Claudius until the final scene of the play where he kills Claudius
and proves that he has progressed into a truly existential character.  At
the beginning of the play, Hamlet acts out of pure intellect and processed
logic. He suppresses his natural instincts, his emotions, and trusts only
in the power of his intelligence. For instance, when Hamlet encounters his
father's ghost, he does not believe it is his father, even though he has
an emotional reaction upon seeing it.
```

Plik zawierający ten tekst został użyty w zmodyfikowanej wersji programu. Należało znaleźć wzorzec zawierający przynajmniej 8 znaków, dlatego postanowiono na znalezienie wzorca „`existential`”. Kompilacja i wyniki działania programu przestawione są niżej.

```
[32b-g1@intek Z1]$ gcc Algorytm-NZ.c
[32b-g1@intek Z1]$ ./a.out
Podaj sciezke do pliku z tekstem: tekst
Podaj wzorzec: existential

Dlugosc pliku: 2001
Dlugosc wzorca: 11

Indeksy wystapien wzorca w tekscie
80
332
400
710
866
1639
```

4.1.4 Komentarze i wnioski

Sposób implementacji pierwotnego programu nie pozwala na wczytywanie tekstu, który zawiera białe znaki. Dlatego wpisany w konsoli tekst jest oddzielony znakami podkreślenia.

Kod zaimplementowany w algorytmie jest podatny na wielkość znaków, dlatego pomimo występowania w tekście słów `Existential` zastosowanie wzorca `existential` nie odnajduje tych przypadków.

Indeksy wystąpień wzorca w tekście zgadzają się z faktycznymi indeksami słów, które można sprawdzić na przykład przy wykorzystaniu narzędzia szukania w edytorze tekstu `gedit`.

Zaimplementowany algorytm jest algorytmem zachłannym i istnieją sposoby jego ulepszenia.

4.2 Zadanie nr 2

4.2.1 Treść polecenia

Według poleceń znajdujących się w podpunktach ćwiczenia należało pobrać do katalogu prywatnego program `bad-czy-pierwsza.c`, który sprawdza, czy podana wartość całkowita jest liczbą pierwszą. Należało skompilować i uruchomić program, sprawdzając jego działanie.

Następne czynności związane były z modyfikacją kodu. Należało:

- Przetłumaczyć napisy wyjściowe programu na język polski, zachowując ich sens
- Zmienić sposób działania programu w taki sposób, by w pętli generował liczbę naturalną i sprawdzał, czy jest ona liczbą pierwszą, powtarzając pętlę tyle razy, jaka jest wczytana przy uruchomieniu programu liczba prób; próby udane (ilość liczb pierwszych) miały zostać zliczane
- Wyświetlić liczbę udanych prób oraz całkowitą ilość prób po wykonaniu pętli programu

Po modyfikacji programu należało sprawdzić jego poprawność. W tym celu trzeba było wykonać obliczenia dla minimalnie 500 liczb naturalnych wygenerowanych w pętli programu.

4.2.2 Cel czynności

Celem czynności była modyfikacja kodu o dodatkowe funkcje, co wiąże się z umiejętnością programowania w języku C i korzystania z dokumentacji.

4.2.3 Sposób i rezultaty

Treść pobranego programu `bad-czy-pierwsza.c` przed modyfikacją została zaprezentowana poniżej.

```
#include<stdio.h>

int main() {
   int num, i, count = 0;

   printf("Enter a number:");
   scanf("%d", &num);

   for (i = 2; i <= num / 2; i++) {
      if (num % i == 0) {
         count++;
         break;
      }
   }

   if (count == 0)
      printf("%d is a prime number\n", num);
   else
      printf("%d is not a prime number\n", num);

   return 0;
```

```
}
```

Kompilacja i uruchomienie programu dało oczekiwane rezultaty, różniące się zależnie od wprowadzonej liczby.

```
[32b-g1@intek Z2]$ gcc bad-czy-pierwsza.c
[32b-g1@intek Z2]$ ./a.out
Enter a number:11
11 is a prime number

Enter a number:14
14 is not a prime numer
```

Znajdujące się w programie napisy wyjściowe zostały przetłumaczone na język polski. Został zachowany ich pełny sens. W programie zmieniony został przepływ, który zaczął działać na pętli, której ilość prób zależna jest od liczby wprowadzonej przez użytkownika. Po udanym wykonaniu pętli, program wyprowadza liczbę wylosowanych liczb pierwszych oraz całkowitą liczbę prób.

Zmodyfikowany kod programu znajdujący się w pliku `bad-czy-pierwsza-M.c` pokazany jest poniżej.

```
#include <stdio.h>
#include <stdlib.h>

int main()
{
   int num, i, j, tryCount;
   int primeCount = 0;
   int isNotPrime;

   printf("Wprowadz liczbe prob: ");
   scanf("%d", &tryCount);
   printf("\n");

   for(j = 0; j < tryCount; j++)
   {
       num = lrand48();
       printf("Sprawdzam %d wygenerowana liczbe..\r", j+1);
       fflush(stdout);

       isNotPrime = 0;

       for (i = 2; i <= num / 2; i++)
       {
          if (num % i == 0)
          {
             isNotPrime = 1;
             break;
          }
       }

       if (isNotPrime == 0)
       {
          primeCount++;
       }

   }
```

```
    printf("\rNa %d prob, %d z wygenerowanych wartosci to liczby pierw-
sze\n",
            tryCount, primeCount);

   return 0;
}
```

Napisany kod został przetestowany. Tak jak zakładało zadanie, zbadano jego działanie na 500 wykonaniach pętli generującej naturalną liczbę pseudolosową. Działanie kodu zostało zaprezentowane poniżej.

```
[32b-g1@intek Z2]$ gcc bad-czy-pierwsza-M.c
[32b-g1@intek Z2]$ ./a.out
Wprowadz liczbe prob: 500

Sprawdzam 1 wygenerowana liczbe..
Sprawdzam 2 wygenerowana liczbe..
Sprawdzam 3 wygenerowana liczbe..

[…]

Sprawdzam 498 wygenerowana liczbe..
Sprawdzam 499 wygenerowana liczbe..
Sprawdzam 500 wygenerowana liczbe..

Na 500 prob, 29 z wygenerowanych wartosci to liczby pierwsze
```

4.2.4 Komentarze i wnioski

Kod zmodyfikowanego programu zawiera w sobie znak powrotu karetki `'\r'`. W zapisanych logach jego użycie nie jest zauważalne (widziane są kolejne linie wyświetlanego tekstu). Znak trójkropka w przykładowym uruchomieniu programu został użyty do zmniejszenia objętości sprawozdania.

Polecana przez prowadzącego funkcja `lrand48()` generuje dodatnie liczby naturalne pseudolosowe. Z racji statycznego ziarna używanego przy generacji liczby, program wykonywalny będzie generował zawsze taki sam ciąg liczb, co zostało potwierdzone ponownym wykonywaniem tego programu.

5. Wnioski z przeprowadzonych prac

Celem ćwiczenia było przypomnienie wiedzy i przywołanie umiejętności opracowywania programów w języku C w środowisku Linuxa. Dzięki modyfikacji prostych skryptów i uruchamianiu ich na zdalnym serwerze Linuxa założenie to zostało w pełni spełnione.

Poprawność zmodyfikowanych programów została sprawdzona przez wykonanie testów, wybrana część została przedstawiona w sprawozdaniu.

Zastosowana w drugim zadaniu funkcja `lrand48()` generuje liczby pseudolosowe za pomocą ziarna statycznego. W celu wygenerowania ciągów liczb, które mają mniejszą szansę na powtórzenie się, należy użyć ziarna bazującego np. na podstawie czasu.