PRZYKŁADY ZASTOSOWAŃ SYSTEMU OPERACYJNEGO *DEBIAN*

NOGA Henryk, PL

Resumé

Projekt ma na celu pokazanie bezpłatnej alternatywy dla systemu operacyjnego Windows firmy Microsoft do prowadzenia obliczeń inżynierskich.[1] Na potrzeby tego projektu złożono tanią i wydajną jednostkę centralną, na której zostaną wykonane wszelkie obliczenia. Przedstawione programy są stosunkowo łatwe i nie wymagają kosztownego sprzętu czy profesjonalnego oprogramowania. wystarczy niewielka wiedza z zakresu budowy i obsługi komputera oraz znajomość podstaw programowania.[2]

Klíčová slova: obliczenia inżynierskie, system operacyjny, programy komputerowe, projekt, Linux.

ALTERNATIVE OPERATING SYSTEMS FOR ENGINEERING CALCULATIONS ON THE EXAMPLE OF *DEBIAN* OPERATING SYSTEM

Abstract

The aim of this project is to show a cost free alternative to the Windows operating system produced by Microsoft Corporation, to carry out engineering calculations. For the purpose of this project, a cost-effective and efficient central processing unit was assembled, on which the calculations will be performed. Presented programs are relatively easy and do not require expensive equipment nor professional software, only little knowledge of operating a computer, computer construction and understanding the basis of programming in needed.

Key words: engineering calculations, operating system, computer programs, project, Linux.

1. Cele projektu

Ma to na celu uzmysłowienie faktu, że 10 letni sprzęt może być w dalszym ciągu użyteczny.

Podstawę projektu stanowi jednostka centrala, w skład której wchodzą następujące niezbędne do działania składniki:

- procesor P4 1.5 Ghz,
- 256 MB Ram,
- karta graficzna 32Mb,
- pendrive, 1Gb [W naszym przypadku wykorzystany, jako dysk twardy].

Jak można zauważyć są to podzespoły tanie i dające duże możliwości. System operacyjnym, na którym wykonywane będą obliczenia został wybrany z szerokiej gamy dystrybucji Linux. W odróżnieniu od systemu operacyjnego Windows największą zaletą systemu Linux jest bezpłatność. W naszym przypadku wybór padł na Debiana, oczywiście do prowadzenia obliczeń można wykorzystać inne dystrybucje, Linuxa takie jak: Ubuntu, Fedora, openSUSE, Gentoo itp. Debian, jako system operacyjny stwarza ogromne możliwości pod względnym ustawień, jak i pracy. Jego największą zaletą jest stabilność, wydajność, niskie wymagania, co przyczyniło się bezpośrednio do wyboru tego systemu operacyjnego z szerokiej gamy innych dystrybucji.

Nasz system operacyjny został przygotowany specjalnie od prowadzenie obliczeń inżynierskich, dlatego nie znajdziemy w nim zainstalowanego środowiska graficznego czy różnego

[1] Burgerová J., Internet vo výučbe a štýly učenia, Prešov 2001, s. 8-16 oraz Burgerová J., Burger V., Systémové a aplikačné program pre personálne počítače, FHPV PU, Prešov 2002, s.12-32.

[2] S. Prata, Język C. Szkoła programowania, 2006, s.20-38.

rodzaju pakietów graficznych i dźwiękowych.[3] Jednakże każda z dystrybucji Linuxa posiada szereg środowisk graficznych, które możemy wybrać w trakcie instalacji oto kilka z nich: KDE, GNOME, Xfce. Wielką zaletą środowiska graficznego są różnego rodzaju narzędzia za pomocą, których możemy w prosty sposób skonfigurować system pod własne potrzeby.

Artykuł zawiera kilka przykładowych kodów służących do wykonywania obliczeń twardości, obciążeń i pomiarów strat ciepła w domu pasywnym. Są to przykłady obliczeń inżynierskich, które uświadamiają, jakie możliwości daje znajomość prostego programowania w języku, np. C oraz odkrywają możliwości taniego sprzętu w prowadzeniu badań. Nie ma przeszkody, by na tak przygotowanej jednostce centralnej nie prowadzić obliczeń długoterminowych.

Poniżej zaprezentowano kilka prostych programów. Wszystkie programy zostały napisane w języku C i skompilowane przy pomocy kompilatora gcc dołączonego do każdej dystrybucji systemu Linux.

2. Obliczanie twardości

Program stanowi prosty kalkulator przeliczający podane wartości na twardość w odpowiedniej skali.

Na początku program wyświetla zapytanie, jakiej metody użyto przy próbie twardości, następnie używa odpowiedniej funkcji dla każdego ze sposobów pomiaru i za pomocą instrukcji switch określa dalsze postępowanie.

```
switch(wybor)
 {
  case 1: printf("Podaj glebokosc wglebienia [mm]: ");
                    scanf("%lf",&h);
                    rockwell(h);
                    break;
  case 2: printf("Podaj srednice wglebnika [mm]: ");
          scanf("%lf",&D);
          printf("Podaj sile nacisku [N]: ");
                    scanf("%lf",&F);
                    printf("Podaj srednice wglebienia [mm]: ");
                    scanf("%lf",&d);
          brinell(D,F,d);
          break;
  case 3: printf("Podaj pierwsza i druga przekatna [mm]: ");
          scanf("%lf%lf",&d1,&d2);
          printf("Podaj sile nacisku [N]: ");
                    scanf("%lf",&F);
                    vickers(d1,d2,F);
          break;
  default: printf("Nie ma takiej opcji.\n");break;
}
```

[3] DEPEŠOVÁ J., TOMKOVÁ V., Tradičné technológie a 21. storočie. In:. Zborník Premeny Slovenského školstva na prahu nového milénia. Nitra: PF UKF, 2001. s. 410 - 413. ISBN 80-8050-470-9. DEPEŠOVÁ J., Postavenie exkurzií v štúdiu technickej výchovy. In: Zborník Vplyv technickej výchovy na rozvoj osobnosti žiaka. Nitra: PF UKF, 1999. s. 39 – 40. ISBN 80-8050-370-2. VARGOVÁ M., Technology Educationin Basic and Upper Secondary Schools.- Slovak Republic. In: UNESCO – The Development of new Approaches in Technology and Vocational Education in the Countries in Transition – the Countries of Central Europe and Souht Africa. An Internation Pilot Project. Participation Programme for Years 2002-2003. No. 183 711 16 ONG. 2003, 285-286 p.

W przypadku wyboru metody Rockwell'a program poprosi o podanie głębokości wgłębienia i przekaże tą wartość funkcji, która dodatkowo zapyta o rodzaj użytego wgłębnika i w zależności od odpowiedzi użyje odpowiedniego wzoru. Następnie bazując na pobranych wartościach oblicza wartość twardości i wyświetli wynik na ekran.

```
void rockwell(double h)
{
 int wglebnik = 0;
 double HRA = 0, HRC = 0;
 printf("Wybierz rodzaj wglebnika:\n1.Stozek\n2.Piramida\n");
 scanf("%d",&wglebnik);
 switch(wglebnik)
 {
  case 1: HRA = 100 – 500 * h;
          printf("HRA = %lf\n",HRA);
                  break;
  case 2: HRC = 130 – 500 * h;
          printf("HRC = %lf\n",HRC);
          break;
  default: printf("Nie ma takiej opcji.\n");
                   break;
 }
}
```

Wybierając metodę Brinell'a program zapyta o średnicę wgłębnika, siłę i średnicę wgłębienia, a następnie przekaże otrzymane liczby do funkcji zawierającej wzór, do którego podane wartości zostaną podstawione. Po wykonaniu obliczeń funkcja wypisze wyliczoną wartość na ekranie.

```
void brinell(double D, double F, double d)
{
 double HRB = 0;
 HRB = 0.102 * 2 * F / (PI * D * (D – sqrt(pow(D,2) – pow(d,2))));
 printf("HRB = %lf [N/mm^2]\n",HRB);
}
```

Przy wyborze metody Vickers'a zostajemy poproszeni o podanie obu przekątnych odcisku i siły, jakiej użyto. Tak, jak w powyższych przykładach, wprowadzone wartości zostaną przesłane do procedury, która na ich podstawie obliczy uśrednione pole powierzchni bocznej odcisku oraz samą wartość twardości i na końcu wypisze ją na ekranie.

```
void vickers(double d1, double d2, double F)
{
 double HV = 0, dsr = 0, p = 0, A = 0;
 dsr = (d1 + d2) / 2;
 p = dsr*dsr;
 HV = 0.18915 * F / p;
 printf("HV = %lf [N/mm^2]\n",HV);
}
```

Jak widać program nie jest skomplikowany. Jeszcze prostszym okazuje się być program wykonujący obliczenia obciążeń stałych i zmiennych. Jak powyższy przykład, również i ten opiera się na instrukcji **switch,** która określa, jakiej funkcji należy użyć.

```
switch(decyzja1)
{
 case 1:  printf("1.Na jednostke powierzchni\n2.Na jednostke dlugosci\n3.Jako ciezar calego elementu\n ");
          scanf("%lf",&decyzja2);
                  if(decyzja2 == 1)
                   {
                    printf("Podaj iloczyn ciezaru objetosciowego gamma, wspolczynnik obciazenia i grubosc:\n");
                      scanf("%lf%lf%lf",&gamma,&gammaf,&h);
                    powierzchnia(gamma,gammaf,h);
                     }
                   else if(decyzja2 == 2)
                   {
                    printf("Podaj iloczyn ciezaru objetosciowego gamma, wspolczynnik obciazenia, grubosc i szerokosc:\n");
                    scanf("%lf%lf%lf",&gamma,&gammaf,&h,&b);
                    dlugosc(gamma,gammaf,h,b);
                     }
                   else if(decyzja2 == 3)
                   {
                    printf("Podaj iloczyn ciezaru objetosciowego gamma, wspolczynnik obciazenia oraz dlugosc, grubosc i wysokosc:\n");
                      scanf("%lf%lf%lf%lf",&gamma,&gammaf,&l,&b,&h);
                    caly_element(gamma,gammaf,h,b,l);
                   }
                    break;
/*Powyżej zostaje dookreślony typ liczonego obciążenia i zostają pobrane potrzebne dane.*/
 case 2:  printf("Podaj wartosc obciazenia sniegiem gruntu qsk oraz wspolczinnik ksztaltu i pochylenia dachu:\n");
          scanf("%lf%lf",&qsk,&C);
          snieg(qsk,C);
          break;
 case 3:  printf("Podaj wartosc charakterystyczna cisnienia predkosci wiatru qwk, wspolczynnik ekspozycji, wspolczynnik areodynamiczny i wspolczynnik dzialania porywow wiatru:\n");
          scanf("%lf%lf%lf%lf",&qwk,&Ce,&C,&beta);
          wiatr(qwk,Ce,C,beta);
          break;
 default: printf("Nie ma takiej opcji\n"); break;
}
```

Poniższe funkcje obliczają odpowiednie wartości charakterystyczne i obliczeniowe, a następnie wypisują je na ekran.

```
void powierzchnia(double gamma, double gammaf, double h)
{
 double gk,go;
 gk = gamma * h;
 go = gk * gammaf;
```

```
        printf("Wartosc charakterystyczna gk = %lf [kN/m^2].\nWartosc obliczeniowa go = %lf
[kN/m^2]\n",gk,go);
        }

        void dlugosc(double gamma, double gammaf, double h, double b)
        {
         double gk,go;
         gk = gamma * h * b;
         go = gk * gammaf;
         printf("Wartosc charakterystyczna gk = %lf [kN/m^2].\nWartosc obliczeniowa go =
%lf\n",gk,go);
        }

        void caly_element(double gamma, double gammaf, double h, double b, double l)
        {
         double Gk,Go;
         Gk = gamma * h * b * l;
         Go = Gk * gammaf;
         printf("Gk = %lf [kN/m^2]. Go = %lf [kN/m^2]\n", Gk, Go);
        }

        void snieg(double qsk, double C)
        {
         double sk,so;
         double gammaf = 1.4;
         sk = qsk * C;
         so = gammaf * sk;
         printf("Wartosc charakterystyczna obciazenia sniegiem dachu sk = %lf [kN/m^2].\nWartosc
obliczeniowa obciazenia sniegiem dachu so = %lf [kN/m^2]\n", sk, so);
        }

        void wiatr(double qwk, double Ce, double C, double beta)
        {
         double pk,po;
         double gammaf = 1.3;
         pk = qwk * Ce * C * beta;
         po = gammaf * pk;
         printf("Obciazenie charakterystyczne pk = %lf [kN/m^2].\nObciazenie obliczeniowe po = %lf
[kN/m^2]\n",pk,po);
        }
```

3. Program do liczenia strat ciepła na przykładnie domu pasywnego

Obliczanie strat ciepła jest pracochłonne, ponieważ wzór ostateczny składa się z wielu pojedynczych elementów zaczynając od strat ciepła wynikających z użytych materiałów a kończąc na mostkach cieplnych powstających na połączeniach ścian czy wokół okien. W trakcie przybliżania problemu przedstawione zostaną fragmenty kodu lub ciekawsze z funkcji liczących. Liczenie współczynników przenikalności ścian zewnętrznych i wewnętrznych.

Zmienne takie jak liczba_warstw_zew, liczba_warstw_wew są przesyłane do funkcji.

```
void info_warstwy(double liczba_warstw_zew, double liczba_warstw_wew, double*
warstw_zew, double* warstw_wew, double* ze_U, double* we_U)
```

```
{
double b,c;
int i, mm;
{
1--->           for (i = 0; i < liczba_warstw_zew; i++)
                {
                        printf("podaj grobosc %d warstwy: ",i);
                scanf("%lf",&b);
                        b = fabs(b);
                printf("podaj wspolczynnik przenikalnosci tej sciany: ");
                scanf("%lf",&c);
                        c = fabs(c);
                        warstw_zew[i] = b / c;
                }
2--->           for (i = 0; i < liczba_warstw_wew; i++)
        {
                printf("podaj grobosc %d warstwy: ",i);
                scanf("%lf",&b);
                b = fabs(b);
                printf("podaj wspolczynnik przenikalnosci tej sciany: ");
                scanf("%lf",&c);
                c = fabs(c);
                warstw_wew[i] = b / c;
        }
        if ( liczba_warstw_zew > liczba_warstw_wew)
          mm = liczba_warstw_zew;
        else
          mm = liczba_warstw_wew;
        b = 0;
        c = 0;
        for(i = 0; i < mm; i++)
        {
                 b += warstw _zew[i];
                 c += warstw_wew[i];
        }
        *ze_U = 1 / (b + 0.17);
        *we_U = 1 / (c + 0.26);
        printf("\n ze_U = %lf we_U = %lf",*ze_U,*we_U);
}
}
```

Jak można zauważyć analizując ten fragment funkcji, wszystkie obliczenia odbywają się na bieżąco, przez co można je śledzić cały czas. Właśnie stąd na końcu funkcji widnieje **printf**.

Pierwsza pętla **for** [zaznaczona 1--->] jest odpowiedzialna za wczytanie wszystkich informacji o ścianie, jej grubości oraz własnego współczynnika przepuszczalności. Druga pętla **for** [zaznaczona 2--->] zachowuje się w identyczny sposób, licząc ściany wewnętrzne znajdujące się wewnątrz przestrzeni domu. Następnie widzimy instrukcje warunkową odpowiedzialną za sprawdzenie, których warstw jest więcej, ułatwia to sumowanie i daje oszczędność miejsca oraz zminimalizowanie kodu. Ostatnia z pętli odpowiedzialna jest za sumowanie poszczególnych wyników zapisanych w tablicy. Funkcja kończy się wzorem wyliczającym współczynnik przenikalności danej ściany.

Innym prostym i ciekawym fragmentem kodu jest obliczenie ilości energii, jaką tracą ściany.

```
printf("podaj ile jest zewnętrznych stykających z podwórzem ");
scanf("%d",&pole);
```

```
for(a = 0, b = 1, i = 0; i < ile_jest_scian; a += 2, b += 2, i++)
{
        w = fabs(wymiary[a] * wymiary[b]);
        if ( a < pole) w*= ze_U;
        else w*= we_U;
        wynik += w;
}
```

Znana jest liczba ścian. Funkcja pyta się, ile ze ścian jest zewnętrznych, czyli bezpośrednio związanych z podwórzem. Następnie odczytuje z tablicy wymiary ścian, które są przedstawione w niej w następujący sposób: na pozycjach parzystych występują długości, na pozycjach nieparzystych wysokości. Następnie mnoży powierzchnie ściany przez współczynnik obliczony wcześniej i sumuje wszystko w zmiennej wynik. Zmienna ta jest ilością energii, którą traci się przez ściany.

Poniżej znajduje się jeszcze jeden prosty skrypt na obliczanie strat energii przez okna.

```
double okna()
{
        double wymiara, wymiarb, wynik = 0, p = 1;
        int i, o;
        while(p != 0)
        {
                printf("podaj liczbe okien: ");
                scanf("%d",&o);
                if (o < 0)
                {
                        printf(”liczba okien nie może być mniejsza od 0”);
                        p = 1;
                } else
                          p = 0;
        }
         if(o == 0) return 0;
        for ( i = 0; i < o; i++)
        {
                printf("podaj rozmiar okna :x_y: ");
                scanf("%lf %lf", &wymiara, &wymiarb);
                        printf("podaj wspolczynnik przenikalnosc");
                scanf("%lf",&p);
                wynik += fabs(wymiara * wymiarb) * p;
        }
        return wynik;
}
```

Na początku funkcji znajduje się pętla, w której zawarto formularz pobierania danych. Służy on zabezpieczeniu przed przypadkowym wpisaniem wartości ujemnej. W taki oto sposób działa całość algorytmu. Nie jest on złożony ani skomplikowany. Wynik to suma poszczególnych funkcji.[4]

[4] VARGOVÁ M., Conditions of New Approaches in Technology and Vocational Education.- Slovak Republic. In: UNESCO – The Development of new Approaches in Technology and Vocational Education in the Countries in Transition – the Countries of Central Europe and Souht Africa. An Internation Pilot Project. Participation Programme for Years 2002-2003. No. 183 711 16 ONG. 2003, 286-288 p.

Podsumowanie

Programy przedstawione powyżej jest w stanie napisać każda osoba, która chce w jakiś sposób ułatwić sobie życie. Nie musi przy tym posiadać kosztownego sprzętu czy profesjonalnego oprogramowania, wystarczy niewielka wiedza z zakresu budowy i obsługi komputera oraz znajomość podstaw programowania.

Bibliografia

1. BURGEROVÁ J., Internet vo výučbe a štýly učenia, Prešov 2001.
2. BURGEROVÁ J., Burger V., Systémové a aplikačné program pre personálne počítače, FHPV PU, Prešov 2002.
3. DEPEŠOVÁ J., TOMKOVÁ V., Tradičné technológie a 21. storočie. In:. Zborník Premeny Slovenského školstva na prahu nového milénia. Nitra: PF UKF, 2001. s. 410 - 413. ISBN 80-8050-470-9.
4. DEPEŠOVÁ J., Postavenie exkurzií v štúdiu technickej výchovy. In: Zborník Vplyv technickej výchovy na rozvoj osobnosti žiaka. Nitra: PF UKF, 1999. s. 39 – 40. ISBN 80-8050-370-2.
5. PORUBSKÁ, G. 1999. Postavenie a význam didaktiky v príprave učiteľov 1.stupňa ZŠ. In: Za vyššiu úroveň výchovy a vzdelávania a prestíž učiteľa ZŠ. Nitra: PdF 1999. s. 67-71.
6. VARGOVÁ M., Technology Educationin Basic and Upper Secondary Schools.- Slovak Republic. In: UNESCO – The Development of new Approaches in Technology and Vocational Education in the Countries in Transition – the Countries of Central Europe and Souht Africa. An Internation Pilot Project. Participation Programme for Years 2002-2003. No. 183 711 16 ONG. 2003, 285-286 p.
7. VARGOVÁ M., Conditions of New Approaches in Technology and Vocational Education.- Slovak Republic. In: UNESCO – The Development of new Approaches in Technology and Vocational Education in the Countries in Transition – the Countries of Central Europe and Souht Africa. An Internation Pilot Project. Participation Programme for Years 2002-2003. No. 183 711 16 ONG. 2003, 286-288 p.
8. PRATA Stephen, Język C. Szkoła programowania, 2006, ISBN: 83-246-0291-7.

Lektoroval: dr inż. Krzysztof Pytel

Kontaktní adresa:

Henryk Noga, dr hab., prof. UP
Uniwersytet Pedagogiczny
ul. Podchorążych 2
30-087 Kraków
senoga@cyf-kr.edu.pl