Wojskowa Akademia Techniczna
Wydział Elektroniki
Instytut Telekomunikacji

Studia I°

Programowanie w języku C/C++ dla zastosowań sieciowych
Część 2

Materiały pomocnicze do zajęć
Dodatek 1

dr inż. Jarosław Krygier

Warszawa 2020

Spis treści

1 Listy wiązane

1.1 Plik główny

```
/*
 ============================================================================
 Name        : wyklad.c
 Author      : Krygier
 Version     :
 Copyright   : WEL WAT
 Description : Przykladowy program do testowania list wiaznych
 ============================================================================
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>  //dla deklaracji memset i memcpy
#include "plik_naglowkowy.h"

#define MAX_BUFF_SIZE 20; //20 pakietow

// deklaracje funkcji
int dodaj_element ();
int usun_pierwszy_element ();

int main(void) {
   /// @brief Funkcja main
   /**
    * @param void Brak atrybutow
    * @return Program zwraca \a int
    */
   int i, n, l; // zmienne pomocnicze
   struct bufor *adres_zakotwiczenia = NULL;
   struct bufor *obceny_adres = NULL;
   struct bufor *ostatni = NULL;
   struct bufor *temp;
   unsigned char dane[1500];
   int max_buffor = MAX_BUFF_SIZE;
   memset (dane, 0, 1500); //wyzerowanie tablicy


   for (i=0;i<10;i++){
      //pobierz dane
      memset (dane, i+1, 1500); //nowe dane
      //sprawdz czy bufor (lista) jest pusta
      if (adres_zakotwiczenia == NULL) {
         //adres zakotwiczenia jest NULL -> lista jest pusta
         //przydziel dynamicznie pamiec dla elementu listy
         adres_zakotwiczenia = (struct bufor*) malloc (sizeof(struct bufor));
         obceny_adres = adres_zakotwiczenia;
         obceny_adres->pierwszy = adres_zakotwiczenia;
         obceny_adres->poprzedni = NULL;
         obceny_adres->nastepny = NULL;
         memcpy (obceny_adres->pakiet, dane, 1500);
         obceny_adres = NULL;

      } else {
         //adres zakotwiczenia nie jest NULL, lista nie jest pusta
         //znajdz ostatni element listy
         ostatni = adres_zakotwiczenia;
         for(n=0;n<max_buffor;n++){
            if (ostatni->nastepny != NULL) {
               ostatni = ostatni->nastepny;
            }
```

```
            else {
               break;
            }
         }
         obceny_adres = (struct bufor*) malloc (sizeof(struct bufor));
         ostatni->nastepny = obceny_adres;
         obceny_adres->pierwszy = adres_zakotwiczenia;
         obceny_adres->poprzedni = ostatni;
         obceny_adres->nastepny = NULL;
         memcpy (obceny_adres->pakiet, dane, 1500);
      }
   }

   //zliczaj liczbe elmentow
   l=0;
   for(temp = adres_zakotwiczenia; temp != NULL; temp=temp->nastepny){
      l++;
   }
   printf ("liczba elementow: %d\n", l);

   memset (dane, 0xaa, 1500); //nowe dane
   dodaj_element (adres_zakotwiczenia, dane);

   //zliczaj liczbe elmentow
   l=0;
   for(temp = adres_zakotwiczenia; temp != NULL; temp=temp->nastepny){
      l++;
   }

   printf ("liczba elementow: %d\n", l);

   memset (dane, 0x00, 1500);

   usun_pierwszy_element (&adres_zakotwiczenia, dane);

   //zliczaj liczbe elmentow
   l=0;
   for(temp = adres_zakotwiczenia; temp != NULL; temp=temp->nastepny){
      l++;
   }

   printf ("liczba elementow: %d\n", l);

   puts("Koniec");
   return EXIT_SUCCESS;
}
```

1.2 Plik nagłówkowy

```
/*
 * plik_naglowkowy.h
 *  Author: Krygier
 */

#ifndef PLIK_NAGLOWKOWY_H_
#define PLIK_NAGLOWKOWY_H_

struct bufor {
   struct bufor *nastepny; ///< wskaznik na nastepny element listy
   struct bufor *poprzedni; ///< wskaznik na poprzedni element listy
   struct bufor *pierwszy; ///< wskaznik na pierwszy element listy
   unsigned char pakiet[1500];
};

#endif /* PLIK_NAGLOWKOWY_H_ */
```

1.3 Plik funkcji

```
/*
 * funkcje.c
 *  Author: Krygier
 */

#include <stdio.h>
#include <stdlib.h>
#include "plik_naglowkowy.h"

int dodaj_element (struct bufor *adres_zakotwiczenia, char *dane) {

   int n;
   struct bufor *obceny_adres = NULL;
   struct bufor *ostatni = NULL;
   //unsigned char dane[1500];

   //memset (dane, 0xaa, 1500);
   //sprawdz czy bufor (lista) jest pusta
   if (adres_zakotwiczenia == NULL) {
      //adres zakotwiczenia jest NULL -> lista jest pusta
      //przydziel dynamicznie pamiec dla elementu listy
      adres_zakotwiczenia = (struct bufor*) malloc (sizeof(struct bufor));
      obceny_adres = adres_zakotwiczenia;
      obceny_adres->pierwszy = adres_zakotwiczenia;
      obceny_adres->poprzedni = NULL;
      obceny_adres->nastepny = NULL;
      memcpy (obceny_adres->pakiet, dane, 1500);
      obceny_adres = NULL;

   } else {
      //adres zakotwiczenia nie jest NULL lista nie jest pusta
      //znajdz ostatni element listy
      ostatni = adres_zakotwiczenia;
      for(n=0;n<20;n++){
         if (ostatni->nastepny != NULL) {
            ostatni = ostatni->nastepny;
         }
         else {
            break;
         }
      }
      obceny_adres = (struct bufor*) malloc (sizeof(struct bufor));
      ostatni->nastepny = obceny_adres;
      obceny_adres->pierwszy = adres_zakotwiczenia;
```

```
      obceny_adres->poprzedni = ostatni;
      obceny_adres->nastepny = NULL;
      memcpy (obceny_adres->pakiet, dane, 1500);
   }

   return (1);
}


int usun_pierwszy_element (struct bufor **adres_zakotwiczenia, char *dane) {

   struct bufor *temp;
   ///Funckcja do usuwania elementu listy, zwraca wskaznik do skopiowanych danych
   if (*adres_zakotwiczenia !=NULL) {
      //sparwdz czy jest nastepny element
      if ((*adres_zakotwiczenia)->nastepny != NULL) {
         (*adres_zakotwiczenia)->nastepny->pierwszy=(*adres_zakotwiczenia)->nastepny;
         (*adres_zakotwiczenia)->nastepny->poprzedni = NULL;
         temp = (*adres_zakotwiczenia)->nastepny;

         memcpy (dane, (*adres_zakotwiczenia)->pakiet, 1500);

         free (*adres_zakotwiczenia);

         *adres_zakotwiczenia = temp;

      }
      else {
         memcpy (dane, (*adres_zakotwiczenia)->pakiet, 1500);

         free (*adres_zakotwiczenia);

         *adres_zakotwiczenia = NULL;
      }
   }
   else {
      //nie mam nic do skasowania
      return (-1);
   }
   return (1);
}
```

2 Biblioteki dynamiczne

2.1 Plik: biblioteki_dynamiczne.c

```
/*
 ============================================================================
 Name        : biblioteki_dynamiczne.c
 Author      : Krygier
 Version     : 1.0
 Copyright   : Tylko na potrzeby zajęć
 Description : Biblioteki dynamiczne
 ============================================================================
 */
//Utworzenie biblioteki dynamicznej:
//gcc -shared biblioteka.c -o biblioteka.so

/*
Aby mieć 'dynamic library' (dl), linker musi łączyć z opcja -l: dl
Ustawić w Eclipsie: C/C++ -> Settings ->GCC C Linker -> Libraries
Program może ładować w trakcie pracy potrzebne mu biblioteki. Linker dostarcza w tym celu 4
funkcji:
    dlopen ładowanie biblioteki,
    dlsym zwrócenie wskaźnika do odpowiedniego symbolu (funkcji) w bibliotece,
    dlerror obsługa błędów,
    dlclose zamykanie biblioteki.
Funkcje te są udostępniane poprzez nagłówek dlfcn.h (trzeba go dodać do głównego programu).

#include <dlfcn.h>
void *dlopen(const char *filename, int flag);
char *dlerror(void);
void *dlsym(void *handle, const char *symbol);
int dlclose(void *handle);
*/

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


int main(void) {
   puts("Dolaczanie bibliotek dynamicznych");
   puts("==================================");

   int status_wyjscia; // do dlclose

   void  *Biblioteka; // wskaznik do bilbioteki
   int   (*Funkcja)(int, int); //wskzanik do fukcji
   double   (*Funkcja1)(int, int); //wskzanik do fukcji1


   Biblioteka = dlopen("./biblioteka.so", RTLD_LAZY); // otwarcie biblioteki RTLD_LAZY - wskazniki
na funkcie wewnetrzne sa budowane dopiero w trakcie wywolania dlsym
   if(!Biblioteka) {
      printf ("Error otwarcia: %s\n", dlerror());
      return(1);
   }
   Funkcja = dlsym(Biblioteka, "suma"); // pobranie wskznika na odpowiednia funckje podana za
pomoca nazwy

   printf ("suma:%d \n", Funkcja (4, 2));

   Funkcja = dlsym(Biblioteka, "iloczyn");

   printf ("iloczyn:%d \n", Funkcja (4, 2));

   Funkcja = dlsym(Biblioteka, "roznica");
```

```
   if (Funkcja== NULL) {
      printf ("Blad wywolania funkcji 'roznica()': brak w bibliotece\n");
   } else {
      printf ("roznica:%d \n", Funkcja (8, 2));
   }

   status_wyjscia = dlclose (Biblioteka); //zamkniecie biblioteki - > jesli poprawnie to
status_wyjscia = 0
   if(status_wyjscia) { // jesli 0 (falsz) -> dlopen() zamknela biblioteke poprawnie
      printf ("Error zamkniecia: %s\n", dlerror());
      return(1);
   }

   //===================
   Biblioteka = dlopen("./biblioteka.so", RTLD_NOW); //RTLD_NOW - wskazniki na funkcie wewnetrzne
sa tu budowane potem dlsym je wykorzystuje
      if(!Biblioteka) {
         printf ("Error otwarcia: %s\n", dlerror());
         return(1);
      }
      Funkcja1 = (double (*)(int, int)) dlsym(Biblioteka, "iloraz"); //z podpowidzia jaki wskaznik
do jakiej funckji bedzie zwracany

      printf ("iloraz:%f \n", Funkcja1 (5, 3));

      status_wyjscia = dlclose (Biblioteka);
      if(status_wyjscia) { // jesli 0 (falsz) -> dlopen() zamknela biblioteke poprawnie
         printf ("Error zamkniecia: %s\n", dlerror());
         return(1);
      }

   return EXIT_SUCCESS;
}
```

2.2 Plik: biblioteka.c

```
/*
 * biblioteka.c
 * Author: Jaroslaw Krygier
 * Tylko na potrzeby zajęć
 */

//Utworzenie biblioteki dynamicznej:
//gcc -shared biblioteka.c -o biblioteka.so


#include <stdio.h>

int suma (int a, int b){
   return (a+b);
}

int iloczyn (int a, int b){
   return (a*b);
}

double iloraz (int a, int b){
   return ((double)a / (double) b);
}

/*int roznica (int a, int b){
   return (a-b);
}*/
```

3 Wskaźniki do funkcji

```
/*
 ============================================================================
 Name        : W_PRC2.c
 Author      : Krygier
 Version     : v1
 Copyright   : Your copyright notice
 Description : Wyklady C cz.2, Wskazniki do funkcji
 ============================================================================
 */

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

//fukcja dodawnia
int dodawanie(int x, int y) {
   return (x+y);
}

//funkcja odejmowania
int odejmowanie (int x, int y) {
   return (x-y);
}

int operacja (int (*) (int, int), int, int); // definicja ponizej fukcji main()

int main(void) {
   puts("Wyklad1"); /* Wyklad1 PRC2*/
   //Zaawansowane operacje na wskaznikach

   int a;
   int b=5;
   int c=6;
   int* wsk;

   ////////////////////
   //wskazniki
   ////////////////////

   wsk = &a;
   printf ("Adres zmiennej 'a': 0x%x\n", (unsigned int) wsk);  // rzutowanie na unsigned int,
poniewaz printf wymaga takiego typu (nie ma warningow)
   wsk = &b;
   printf ("Adres zmiennej 'b': 0x%x\n", (unsigned int) wsk);
   wsk = &c;
   printf ("Adres zmiennej 'c': 0x%x\n", (unsigned int) wsk);

   ////////////////////
   //wskazniki do funkcji
   ////////////////////
   //deklaracja
   int (*funkcja) (int, int);
   //jaka jest roznica z int *funkcja (int, int);

   funkcja = &dodawanie;
   // teraz wskaznik wskazuje na fukncje 'dodawanie'
   printf ("1) Wskaznik na funkcje 'dodawanie': 0x%x\n",(unsigned int) funkcja);

   //lub alternatywnie
   funkcja = dodawanie;
   // teraz wskaznik wskazuje na fukncje 'dodawanie'
   printf ("2) Wskaznik na funkcje 'dodawanie': 0x%x\n", (unsigned int) funkcja);
```

```
   //Wywolanie funkcji przez wskanik
   a = funkcja (b,c);
   printf ("1) Wynik: %d \n", a);
   // lub alternatywnie
   a = (*funkcja) (b,c);
   printf ("2) Wynik: %d \n", a);

   // przypisanie do wskazniaka roznych funkcji
   //teraz ten sam wskaznik wskazuje na inna funkcje
   funkcja = odejmowanie; //mozna przypisac funkcje bez &
   printf ("3) Wskaznik na funkcje 'odejmowanie': 0x%x\n", (unsigned int) funkcja);
   //wywolanie jest identyczne, ale wynik rozny
   a = (*funkcja) (b,c);
   printf ("3) Wynik: %d \n", a);

   ////////////////////
   //tablica wskaznikow do funkcji
   ////////////////////

   // deklaracja tablicy wskaznikow do fukncji
   int (*tablica_wskaznikow_do_funkcji [3]) (int, int);

   //przypisanie adresow fukcji do komorek tablicy
   tablica_wskaznikow_do_funkcji [0] = dodawanie;
   tablica_wskaznikow_do_funkcji [1] = odejmowanie;
   tablica_wskaznikow_do_funkcji [2] = NULL;

   printf ("Wskaznik na funkcje 'dodawanie' [0]: 0x%x\n", (unsigned int)
tablica_wskaznikow_do_funkcji [0]);
   printf ("Wskaznik na funkcje 'odejmowanie'[1]: 0x%x\n", (unsigned int)
tablica_wskaznikow_do_funkcji [1]);
   printf ("Wskaznik na funkcje 'NULL'[2]: 0x%x\n", (unsigned int) tablica_wskaznikow_do_funkcji
[2]);

   //wywolanie funkcji z tablicy

   a = (*tablica_wskaznikow_do_funkcji [0]) (b,c);
   printf ("1) Wynik: %d \n", a);
   a = (*tablica_wskaznikow_do_funkcji [1]) (b,c);
   printf ("2) Wynik: %d \n", a);

   //lub alternatywnie
   a = tablica_wskaznikow_do_funkcji [0] (b,c);
   printf ("1) Wynik: %d \n", a);
   a = tablica_wskaznikow_do_funkcji [1] (b,c);
   printf ("2) Wynik: %d \n", a);

   /*UWAGA!!!*/
   /*
    * Wskaniki do fukcji mozna:
   - przekazywac do innych fukcji jako argumenty
   - zwracac jako wynik dzialania fukcji
   - porownywac z NULL
   * Nie mozma
   - wykonywac operacji matematycznych
   */

   //Przekazanie wskaznika do fukcji jako argumentu innej fukcji
   funkcja = dodawanie;
   a = operacja (funkcja, 3, 5);
   printf ("1) Wynik: %d \n", a);
   //lub bezposrednio nazwa fukcji jest tez wskaznikiem
   a = operacja (dodawanie, 3, 5);
   printf ("2) Wynik: %d \n", a);

   // a teraz przekazujemy wskznik innej fukcji
```

```
   a = operacja (odejmowanie, 3, 5);
   printf ("3) Wynik: %d \n", a);

   //sprawdzenie
   //Napisac fukcje 'operacja_arytm', ktora wykonuje y=(a/b)+c
   //Napisac fukcje o nazwie 'operacje_new' z argumentami (typ_operacji, arg1, arg2, arg3)
zwracajaca wynik:
   // dodawania lub odejmowania, lub wynik fukcji 'operacja_arytm' w zaleznosci od przekazanego
typu operacji

   return EXIT_SUCCESS;
}

// fukcja z przekazanym wskaznikiem do fukcji
int operacja (int (*wsk_do_funkcji_przekazany) (int, int), int a, int b){ // jako pierwsza bedzie
przekazywany wskaznik do fukcji
   int wynik;

   wynik = wsk_do_funkcji_przekazany (a,b);
   // lub
   //wynik = (*wsk_do_funkcji_przekazany) (a,b);

   return wynik;
}
```