Prowadzący zajęcia  
dr inż. Andrzej Urbański

**Laboratoria nr 3**

**Programowanie Niskopoziomowe**

**Podprogramy, rekursja i struktury w C**

Wojciech Regulski  
Informatyka(WI) I1  
nr 132312

Zadanie nr 1.

Opracować program prowadzący spis pracowników firmy.

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define l\_prac 10

struct pracownik

{

char nazwisko[20];

int pensja;

float proc\_premii;

};

void wypisz\_prac(int i, struct pracownik pracownicy[])

{

printf("Pracownik nr: %d\n",i+1);

printf("%s",pracownicy[i].nazwisko);

printf("%d\n",pracownicy[i].pensja);

printf("%.2f%%\n",pracownicy[i].proc\_premii);

system("pause");

}

void dodaj\_prac(int i, struct pracownik pracownicy[])

{

printf("Pracownik nr: %d\n",i+1);

printf("Podaj nazwisko: ");

fflush(stdin);

fgets(pracownicy[i].nazwisko, 20, stdin);

printf("Podaj pensje: ");

scanf("%d",&pracownicy[i].pensja);

printf("Podaj procent premii: ");

scanf("%f",&pracownicy[i].proc\_premii);

}

void zm\_pensje(char nazw[], struct pracownik pracownicy[])

{

int i;

for(i=0;i<l\_prac;i++)

{

if(strncmp(pracownicy[i].nazwisko,nazw,strlen(pracownicy[i].nazwisko)) == 0)

{

printf("Podaj nowa pensje: ");

scanf("%d",&pracownicy[i].pensja);

wypisz\_prac(i,pracownicy);

return;

}

}

printf("Nie znaleziono pracownika\n");

system("pause");

}

void zm\_premie(char nazw[], struct pracownik pracownicy[])

{

int i;

for(i=0;i<l\_prac;i++)

{

if(strncmp(pracownicy[i].nazwisko,nazw,strlen(pracownicy[i].nazwisko)) == 0)

{

printf("Podaj nowa premie: ");

scanf("%f",&pracownicy[i].proc\_premii);

wypisz\_prac(i,pracownicy);

return;

}

}

printf("Nie znaleziono pracownika\n");

system("pause");

}

void wyplata(char nazw[], struct pracownik pracownicy[])

{

int i;

for(i=0;i<l\_prac;i++)

{

if(strncmp(pracownicy[i].nazwisko,nazw,strlen(pracownicy[i].nazwisko)) == 0)

{

wypisz\_prac(i,pracownicy);

printf("\nWyplata wyniesie: %.2f\n",pracownicy[i].pensja+pracownicy[i].pensja\*pracownicy[i].proc\_premii/100);

system("pause");

return;

}

}

printf("Nie znaleziono pracownika\n");

system("pause");

}

int main()

{

struct pracownik pracownicy[l\_prac];

int akt\_l\_prac = 0, i;

char wybor;

do

{

system("cls");

printf("N : nowy pracownik\n");

printf("P : nowa wartosc pensji dla pracownika o podanym nazwisku\n");

printf("R : nowa wartosc procentu premii dla pracownika o podanym nazwisku\n");

printf("W : wyplata\n");

printf("L : lista wszystkich pracownikow\n");

printf("K : koniec programu\n");

scanf("%c",&wybor);

switch(wybor)

{

case 'N':

{

dodaj\_prac(akt\_l\_prac,pracownicy);

wypisz\_prac(akt\_l\_prac,pracownicy);

akt\_l\_prac++;

break;

}

case 'P':

{

char temp[20];

fflush(stdin);

printf("Chcesz zmienic pensje osoby o nazwisku: ");

fgets(temp, 20, stdin);

zm\_pensje(temp,pracownicy);

break;

}

case 'R':

{

char temp[20];

fflush(stdin);

printf("Chcesz zmienic premie osoby o nazwisku: ");

fgets(temp, 20, stdin);

zm\_premie(temp,pracownicy);

break;

}

case 'W':

{

char temp[20];

fflush(stdin);

printf("Chcesz obliczyc wyplate osoby o nazwisku: ");

fgets(temp, 20, stdin);

wyplata(temp,pracownicy);

break;

}

case 'L':

{

for(i=0;i<akt\_l\_prac;i++)

{

wypisz\_prac(i,pracownicy);

printf("\n");

}

}

}

}while(wybor!='K');

return 0;

}

Tablice struktur są przekazywane do funkcji za pomoca referencji, najprościej więc przekazywać tablicę wraz z numerem indeksu i operować na niej w standardowy sposób. Została wykorzystana funcja fflush(stdin), aby czyścić bufor wejścia. Brak czyszczenia bufora powodował czasami problemy i błedy w działaniu. Zadanie nr 2.

Zdefiniować unię o nazwie Bag, zawierającą pola: int, char, float, służącą do przechowywania odpowiednio: liczb całkowitych, znaków oraz liczb rzeczywistych. Przygotować tablicę n losowych elementów typu Bag, losowanie powinno obejmować zarówno wybór typu pola jak i jego wartość.

#include <stdio.h>

#include <stdlib.h>

union Bag

{

int i;

char c;

float f;

};

void wypisz(int n, union Bag bags[])

{

int i;

for(i=0;i<n;i++)

{

printf("Bag nr: %d\n",i+1);

printf("Warianty wartosci w srodku\nliczba calkowita: %d\nznak: %c\nliczba zmiennoprzecinkowa: %f\n\n",bags[i].i,bags[i].c,bags[i].f);

}

system("pause");

}

int main()

{

srand(time(NULL));

int n,i;

printf("Ile losowac? ");

scanf("%d",&n);

union Bag bags[n];

for(i=0;i<n;i++)

{

int wybor = rand()%3;

switch(wybor)

{

case 0:

{

printf("Typ w Bag nr %d: liczba calkowita\n",i+1);

bags[i].i = rand()%100;

break;

}

case 1:

{

printf("Typ w Bag nr %d: znak\n",i+1);

bags[i].c = 65+rand()%26;

break;

}

case 2:

{

printf("Typ w Bag nr %d: liczba zmiennoprzecinkowa\n",i+1);

bags[i].f = (rand()%10000)/100;

break;

}

}

}

printf("\n");

wypisz(n,bags);

return 0;

}

Unia wykorzystuje wspólną pamięć dla wszystkich zmiennych w niej, jest więc bardzo błędogenna.Zadanie nr 3.

Zdefiniować strukturę student. Napisać program umożliwiający wprowadzanie, modyfikację, usuwanie i wyświetlanie danych studentów, wyznaczenie średniej, maksymalnej i minimalnej oceny dla grupy studentów.

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define l\_stud\_max 100

struct Student

{

char imie[20];

char nazwisko[20];

int nr\_albumu;

float ocena;

};

void wypisz\_stud(int i, struct Student studenci[])

{

int j;

printf("Student nr: %d\n",i+1);

printf("Imie: %s",studenci[i].imie);

/\*printf("Imie: ");

for(j=0;j<strlen(studenci[i].imie)-1;j++)

{

printf("%c",studenci[i].imie[j]);

}

print("\n");

\*/

printf("Nazwisko: %s",studenci[i].nazwisko);

printf("Nr albumu: %d\n",studenci[i].nr\_albumu);

printf("Ocena: %.2f\n",studenci[i].ocena);

system("pause");

}

void dodaj\_stud(int i, struct Student studenci[])

{

printf("Student nr: %d\n",i+1);

printf("Podaj imie: ");

fflush(stdin);

fgets(studenci[i].imie, 20, stdin);

printf("Podaj nazwisko: ");

fflush(stdin);

fgets(studenci[i].nazwisko, 20, stdin);

printf("Podaj nr albumu: ");

scanf("%d",&studenci[i].nr\_albumu);

printf("Podaj ocene: ");

scanf("%f",&studenci[i].ocena);

}

void oceny(int n, struct Student studenci[])

{

int min=1024,max=0,i;

float suma=0;

for(i=0;i<n;i++)

{

suma+=studenci[i].ocena;

if(studenci[i].ocena>max)

max=studenci[i].ocena;

if(studenci[i].ocena<min)

min=studenci[i].ocena;

}

printf("Najnizsza ocena: %d\n",min);

printf("Najwyzsza ocena: %d\n",max);

printf("Srednia ocen: %f\n",(float)suma/n);

system("pause");

}

void usun\_stud(int i, struct Student studenci[])

{

int n;

for(n=0;n<20;n++)

{

studenci[i].imie[n]=' ';

studenci[i].nazwisko[n]=' ';

}

studenci[i].nr\_albumu=0;

studenci[i].ocena=0;

}

int main()

{

struct Student studenci[l\_stud\_max];

int akt\_l\_stud = 0, i;

char wybor;

do

{

system("cls");

printf("N : nowy student\n");

printf("E : edycja danych studenta o podanym numerze\n");

printf("W : wyznaczenie sredniej, maksymalnej i minimalnej oceny\n");

printf("D : usuniecie danych o studentach\n");

printf("L : lista wszystkich studentow\n");

printf("K : koniec programu\n");

scanf("%c",&wybor);

switch(wybor)

{

case 'N':

{

dodaj\_stud(akt\_l\_stud,studenci);

printf("\n");

wypisz\_stud(akt\_l\_stud,studenci);

akt\_l\_stud++;

break;

}

case 'E':

{

int temp;

printf("Chcesz zmienic dane studenta nr: ");

scanf("%d",&temp);

if(temp<=akt\_l\_stud && temp>0)

dodaj\_stud(temp-1,studenci);

else

printf("Nie ma studenta o tym numerze");

break;

}

case 'W':

{

if(akt\_l\_stud>=1)

oceny(akt\_l\_stud,studenci);

else

printf("Brak studentow");

break;

}

case 'D':

{

int i=akt\_l\_stud-1;

for(;i>=0;i--)

{

usun\_stud(i,studenci);

}

akt\_l\_stud=0;

break;

}

case 'L':

{

for(i=0;i<akt\_l\_stud;i++)

{

wypisz\_stud(i,studenci);

printf("\n");

}

break;

}

}

}while(wybor!='K');

return 0;

}Zadanie nr 4.

Napisz program będący katalogiem płyt audio (dla maksymalnie 100 płyt). Każda płyta powinna być opisana zestawem danych (tytuł, autor, rok wydania, numer katalogowy, rodzaj muzyki) zawartych w strukturze Plyta. Napisz funkcje wczytującą informacje o płycie i wyświetlającą te informacje. Skonstruuj menu w programie umożliwiające wczytanie nowej płyty lub wyświetlenie wszystkich wprowadzonych płyt na ekranie w odpowiednio sformatowanych kolumnach. Rozszerz program o podmenu dla wyświetlania płytoteki dotyczące wyświetlania płyt według zadanej kolejności. Dopisz procedurę sortowania płyt na podstawie wybranego elementu struktury.

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define l\_plyt\_max 100

#define str\_len 20

struct Plyta

{

int nr;

char tytul[str\_len];

char autor[str\_len];

int rok\_wyd;

char nr\_kat[str\_len];

char rodz\_muz[str\_len];

};

void wyswietlPlyte(int i, struct Plyta plyty[])

{

int j;

for(j=0;j<strlen(plyty[i].tytul)-1;j++)

{

printf("%c",plyty[i].tytul[j]);

}

for(j=str\_len;j>strlen(plyty[i].tytul);j--) printf(" ");

for(j=0;j<strlen(plyty[i].autor)-1;j++)

{

printf("%c",plyty[i].autor[j]);

}

for(j=str\_len;j>strlen(plyty[i].autor);j--) printf(" ");

printf("%d ",plyty[i].rok\_wyd);

for(j=0;j<strlen(plyty[i].nr\_kat)-1;j++)

{

printf("%c",plyty[i].nr\_kat[j]);

}

for(j=str\_len;j>strlen(plyty[i].nr\_kat);j--) printf(" ");

for(j=0;j<strlen(plyty[i].rodz\_muz)-1;j++)

{

printf("%c",plyty[i].rodz\_muz[j]);

}

printf("\n");

}

void wczytajPlyte(int i, struct Plyta plyty[])

{

printf("Plyta nr: %d\n",i+1);

plyty[i].nr=i;

printf("Podaj tytul: ");

fflush(stdin);

fgets(plyty[i].tytul, str\_len, stdin);

printf("Podaj autora: ");

fflush(stdin);

fgets(plyty[i].autor, str\_len, stdin);

printf("Podaj rok wydania: ");

scanf("%d",&plyty[i].rok\_wyd);

printf("Podaj numer katalogowy: ");

fflush(stdin);

fgets(plyty[i].nr\_kat, str\_len, stdin);

printf("Podaj rodzaj muzyki: ");

fflush(stdin);

fgets(plyty[i].rodz\_muz, str\_len, stdin);

}

void sortujPlyty(struct Plyta plyty[], int n, int param)

{

struct Plyta temp;

int i,j;

for (i = 0; i < n - 1; i++)

{

for (j = i + 1; j > 0; j--) //przesuwaj w tyl az wstawisz w odpowiednie miejsce

{

if (porownajPlyty(plyty, j, param))

{

temp = plyty[j];

plyty[j] = plyty[j - 1];

plyty[j - 1] = temp;

}

else

break; //wyjdz gdy jest juz w odpowiednim miejscu

}

}

}

int porownajPlyty(struct Plyta plyty[], int i\_wiekszy, int param)

{

switch(param)

{

case 1:

{

if ( plyty[i\_wiekszy].nr < plyty[i\_wiekszy-1].nr )

return 1;

return 0;

break;

}

case 2:

{

int size;

if( strlen(plyty[i\_wiekszy].tytul) < strlen(plyty[i\_wiekszy-1].tytul) )

size=strlen(plyty[i\_wiekszy].tytul);

else

size=strlen(plyty[i\_wiekszy-1].tytul);

if(strncmp(plyty[i\_wiekszy].tytul,plyty[i\_wiekszy-1].tytul,size)<0)

return 1;

return 0;

break;

}

case 3:

{

int size;

if( strlen(plyty[i\_wiekszy].autor) < strlen(plyty[i\_wiekszy-1].autor) )

size=strlen(plyty[i\_wiekszy].autor);

else

size=strlen(plyty[i\_wiekszy-1].autor);

if(strncmp(plyty[i\_wiekszy].autor,plyty[i\_wiekszy-1].autor,size)<0)

return 1;

return 0;

break;

}

case 4:

{

if ( plyty[i\_wiekszy].rok\_wyd < plyty[i\_wiekszy-1].rok\_wyd )

return 1;

return 0;

break;

}

case 5:

{

int size;

if( strlen(plyty[i\_wiekszy].nr\_kat) < strlen(plyty[i\_wiekszy-1].nr\_kat) )

size=strlen(plyty[i\_wiekszy].nr\_kat);

else

size=strlen(plyty[i\_wiekszy-1].nr\_kat);

if(strncmp(plyty[i\_wiekszy].nr\_kat,plyty[i\_wiekszy-1].nr\_kat,size)<0)

return 1;

return 0;

break;

}

case 6:

{

int size;

if( strlen(plyty[i\_wiekszy].rodz\_muz) < strlen(plyty[i\_wiekszy-1].rodz\_muz) )

size=strlen(plyty[i\_wiekszy].rodz\_muz);

else

size=strlen(plyty[i\_wiekszy-1].rodz\_muz);

if(strncmp(plyty[i\_wiekszy].rodz\_muz,plyty[i\_wiekszy-1].rodz\_muz,size)<0)

return 1;

return 0;

break;

}

}

}

int main()

{

struct Plyta plyty[l\_plyt\_max];

int akt\_l\_plyt = 0, i;

char wybor;

do

{

system("cls");

printf("N : nowa plyta\n");

printf("L : lista wszystkich plyt\n");

printf("K : koniec programu\n");

scanf("%c",&wybor);

switch(wybor)

{

case 'N':

{

wczytajPlyte(akt\_l\_plyt,plyty);

printf("\n");

akt\_l\_plyt++;

break;

}

case 'L':

{

int wybor2;

do

{

system("cls");

printf("1 - wedlug kolejnosci dodania\n2 - wedlug tytulu\n3 - wedlug autora\n4 - wedlug roku wydania\n5 - wedlug numeru katalogowego\n6 - wedlug rodzaju muzyki\nWybierasz: ");

scanf("%d",&wybor2);

}while(wybor2<1 || wybor2>6);

sortujPlyty(plyty,akt\_l\_plyt,wybor2);

printf("tytul autor rok wydania numer katalogowy rodzaj muzyki\n");

for(i=0;i<akt\_l\_plyt;i++)

{

wyswietlPlyte(i,plyty);

}

system("pause");

break;

}

}

}while(wybor!='K');

return 0;

}

Została zastosowana implementacja InsertionSort, gdyż cechuje się on złożonością obliczeniową O(n) w przypadku posortowanych danych, co może tutaj występować często, gdy odczytujemy dane zazwyczaj według jednego klucza.Zadanie nr 5.

Potęgowanie liczby naturalnej za pomocą mechanizmu wielokrotnego dodawania (rekurencyjnie).

int mnozenie(int a, int b)

{

if (b == 1) return a;

return a + mnozenie(a, b-1);

}

int potegowanie(int a, int b)

{

if (b == 1) return a;

return mnozenie(a, potegowanie(a, b-1));

}Zadanie nr 6.

Napisać funkcję wyznaczania silni liczby naturalnej n (rekurencyjnie).

int silnia(int n)

{

if(n==2) return n;

return n\*silnia(n-1);

}Zadanie nr 7.

Napisz funkcję, która wypisze w odwrotnej kolejności znaki wczytane z klawiatury (rekurencyjnie). Przykładowo, gdy zadany ciąg to: [1,2,3,a,c,b,\n] program powinien wypisać: [b,c,a,3,2,1].

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

void wypisz(char str[],int n)

{

printf("%c",str[n-1]);

if(n>1)

wypisz(str,n-1);

}

int main()

{

char str[100];

gets(str);

wypisz(str,strlen(str));

return 0;

}Zadanie nr 8.

Znajdowanie liczb automorficznych w podanym przedziale.

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <pthread.h>

pthread\_t a,b,c,d,e,f,g,h;

int dlugosc(long long x)

{

int dlug=0;

while(x!=0)

{

x/=10;

dlug++;

}

return dlug;

}

void\* automorf(long long x,long long max)

{

while(x<=max)

{

long long tmp = x\*x;

int dziel=1,i;

for(i=0; i<dlugosc(x); i++)dziel\*=10;

tmp%=dziel;

if(tmp==x)

printf("%d\n",x);

x+=8;

}

}

int main()

{

long long min,max;

long long tmp;

printf("Od: ");

scanf("%lld",&min);

printf("Do: ");

scanf("%lld",&max);

for(;min<=max;min++)

{

tmp=min\*min;

long long dziel=1;

int i;

for(i=0;i<dlugosc(min);i++)dziel\*=10;

tmp%=dziel;

if(tmp==min)

printf("%d\n",min);

}

/\*pthread\_create(&a,NULL,automorf(min,max),NULL);

pthread\_create(&b,NULL,automorf(min+1,max),NULL);

pthread\_create(&c,NULL,automorf(min+2,max),NULL);

pthread\_create(&d,NULL,automorf(min+3,max),NULL);

pthread\_create(&e,NULL,automorf(min+4,max),NULL);

pthread\_create(&f,NULL,automorf(min+5,max),NULL);

pthread\_create(&g,NULL,automorf(min+6,max),NULL);

pthread\_create(&h,NULL,automorf(min+7,max),NULL);\*/

return 0;

}

Wynik podniesienia do kwadratu zostaje obcięty do tylu cyfr ile ma liczba bazowa za pomocą reszty z dzielenia przez 10 do potęgi liczby cyfr w liczbie bazowej. Po tym zabiegu wystarczy proste porównanie, czy liczba podniesiona do kwadratu po modyfikacjach jest równa liczbie bazowej. Liczba cyfr w liczbie bazowej zostaje zbadana przez dzielenie jej przez 10 aż dojdzie do 0. Ilość dzieleń jest liczbą cyfr. Zaimplementowana została również wersja szukania liczb automorficznych przy wykorzystaniu wielu wątków.Zadanie nr 9.

Obliczanie sumy cyfr dziesiętnych (rekurencyjnie).

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int suma(char str[],int n)

{

if(n==1)return str[0]-48;

return (str[n-1]-48)+suma(str,n-1);

}

int main()

{

char str[20];

gets(str);

printf("%d",suma(str,strlen(str)));

return 0;

}

Odejmowane jest 48, ponieważ w kodzie ASCII cyfry występują od tej pozycji.Zadanie nr 10.

Napisać program pozwalający na zmianę reprezentacji liczb (dziesiętną, binarną, ósemkową, szesnastkową).

#include <stdio.h>

#include <stdlib.h>

long binaryToDecimal(long n)

{

int remainder;

long decimal = 0, i=0;

while(n != 0)

{

remainder = n%10;

n = n/10;

decimal = decimal + (remainder\*pow(2,i));

++i;

}

return decimal;

}

long decimalToBinary(long n)

{

int remainder;

long binary = 0, i = 1;

while(n != 0)

{

remainder = n%2;

n = n/2;

binary= binary + (remainder\*i);

i = i\*10;

}

return binary;

}

long decimalToOctal(long n)

{

int remainder;

long binary = 0, i = 1;

while(n != 0)

{

remainder = n%8;

n = n/8;

binary= binary + (remainder\*i);

i = i\*10;

}

return binary;

}

long octalToDecimal(long n)

{

int remainder;

long decimal = 0, i=0;

while(n != 0)

{

remainder = n%10;

n = n/10;

decimal = decimal + (remainder\*pow(8,i));

++i;

}

return decimal;

}

void decimalToHexa(long n)

{

char hexDigits[] = "0123456789ABCDEF";

char hexadecimalNumber[40];

int index=0, remaindar;

while(n != 0)

{

remaindar = n % 16;

hexadecimalNumber[index] = hexDigits[remaindar];

n /= 16;

index++;

}

hexadecimalNumber[index] = '\0';

strrev(hexadecimalNumber);

printf("%s\n",hexadecimalNumber);

}

long hexaToDecimal(char hexadecimal[])

{

long decimalNumber=0;

char hexDigits[] = "0123456789ABCDEF";

int i, j, power=0, digit;

for(i=strlen(hexadecimal)-1; i >= 0; i--) {

for(j=0; j<16; j++){

if(hexadecimal[i] == hexDigits[j]){

decimalNumber += j\*pow(16, power);

}

}

power++;

}

return decimalNumber;

}

int main()

{

int wybor;

printf("1-bin to oct\n2-bin to dec\n3-bin to hex\n4-oct to bin\n5-oct to dec\n6-oct do hex\n7-dec to bin\n8-dec to oct\n9-dec to hex\n10-hex to bin\n11-hex to oct\n12-hex to dec\nWybor: ");

scanf("%d",&wybor);

switch(wybor)

{

case 1:

{

long liczba;

printf("Podaj liczbe do konwersji: ");

scanf("%d",&liczba);

printf("%d",decimalToOctal(binaryToDecimal(liczba)));

break;

}

case 2:

{

long liczba;

printf("Podaj liczbe do konwersji: ");

scanf("%d",&liczba);

printf("%d",binaryToDecimal(liczba));

break;

}

case 3:

{

long liczba;

printf("Podaj liczbe do konwersji: ");

scanf("%d",&liczba);

decimalToHexa(binaryToDecimal(liczba));

break;

}

case 4:

{

long liczba;

printf("Podaj liczbe do konwersji: ");

scanf("%d",&liczba);

printf("%d",decimalToBinary(octalToDecimal(liczba)));

break;

}

case 5:

{

long liczba;

printf("Podaj liczbe do konwersji: ");

scanf("%d",&liczba);

printf("%d",octalToDecimal(liczba));

break;

}

case 6:

{

long liczba;

printf("Podaj liczbe do konwersji: ");

scanf("%d",&liczba);

decimalToHexa(octalToDecimal(liczba));

break;

}

case 7:

{

long liczba;

printf("Podaj liczbe do konwersji: ");

scanf("%d",&liczba);

printf("%d",decimalToBinary(liczba));

break;

}

case 8:

{

long liczba;

printf("Podaj liczbe do konwersji: ");

scanf("%d",&liczba);

printf("%d",decimalToOctal(liczba));

break;

}

case 9:

{

long liczba;

printf("Podaj liczbe do konwersji: ");

scanf("%d",&liczba);

decimalToHexa(liczba);

break;

}

case 10:

{

char liczba[20];

printf("Podaj liczbe do konwersji: ");

fflush(stdin);

gets(liczba);

printf("%d",decimalToBinary(hexaToDecimal(liczba)));

break;

}

case 11:

{

char liczba[20];

printf("Podaj liczbe do konwersji: ");

fflush(stdin);

gets(liczba);

printf("%d",decimalToOctal(hexaToDecimal(liczba)));

break;

}

case 12: {

char liczba[20];

printf("Podaj liczbe do konwersji: ");

fflush(stdin);

gets(liczba);

printf("%d",hexaToDecimal(liczba));

break; } } return 0;}

Implementacja podstawowych zasad zamiany bazy systemu liczbowego.Zadanie nr 11.

Imprementacja stosu, listy dwukierunkowej, drzewa BST.

Stos:

int stack[100];

int top = -1;

int push(int data)

{

top = top + 1;

stack[top] = data;

}

int pop()

{

int data;

data = stack[top];

top = top - 1;

return data;

}

void stackSearch(int data)

{

int i;

for(i=top; i>=0; i--)

{

if(stack[i]==data)

{

printf("Znaleziono\n");

return;

}

}

printf("Nie znaleziono\n");

}

Lista:

struct Node

{

int data;

struct Node\* next;

struct Node\* prev;

};

struct Node\* head=NULL;

struct Node\* GetNewNode(int x)

{

struct Node\* newNode

= (struct Node\*)malloc(sizeof(struct Node));

newNode->data = x;

newNode->prev = NULL;

newNode->next = NULL;

return newNode;

}

void Insert(int x)

{

struct Node\* newNode = GetNewNode(x);

if(head == NULL)

{

head = newNode;

return;

}

head->prev = newNode;

newNode->next = head;

head = newNode;

}

void Delete()

{

if(head!=NULL)

{

if(head->next != NULL)

{

struct Node\* temp = head->next;

free(head);

head=temp;

head->prev=NULL;

}

else

{

free(head);

head=NULL;

}

}

}

void PrintList()

{

struct Node\* temp = head;

printf("Lista: ");

while(temp != NULL)

{

printf("%d ",temp->data);

temp = temp->next;

}

printf("\n");

}

void listSearch(int x)

{

struct Node\* temp = head;

while(temp != NULL)

{

if(temp->data==x)

{

printf("Znaleziono\n");

return;

}

temp = temp->next;

}

printf("Nie znaleziono\n");

}

Drzewo BST:

typedef struct wezel

{

int wartosc; //wartosc przechowywana w wezle

struct wezel \*rodzic; //wskaznik na rodzica

struct wezel \*l\_syn; //wskaznik na lewe dziecko

struct wezel \*p\_syn; //wskaznik na prawe dziecko

} wezel;

struct wezel \*root=NULL; //wskaŸnik na root'a

//funkcja zwraca wskaznik elementu o najmniejszej wartosci (najbardziej na lewo)

struct wezel\* naj\_lewo(struct wezel \*start)

{

if(start->l\_syn != NULL)

return naj\_lewo(start->l\_syn);

else

return start;

}

//funkcja zwraca wezel o podanej wartosci, badz NULL, gdy taki wezel nie istnieje

struct wezel\* szukaj(struct wezel \*start, int wartosc)

{

//jezeli wezel ma szukana wartosc to odnalezlismy go

if (start->wartosc == wartosc)

{

printf("Znaleziono\n");

return start;

}

//jezeli szukana wartosc jest mniejsza to szukamy w lewym poddrzewie o ile istnieje

else if (wartosc < start->wartosc && start->l\_syn != NULL) return szukaj(start->l\_syn, wartosc);

//jezeli szukana wartosc jest wieksza to szukamy w prawym poddrzewie o ile istnieje

else if (wartosc > start->wartosc && start->p\_syn != NULL) return szukaj(start->p\_syn, wartosc);

printf("Nie znaleziono\n");

return NULL;

}

//dodaje wezel o podanej wartosci n, do drzewa o korzeniu start

int dodawanie(int n, struct wezel \*start)

{

//jezeli drzewo jest puste to dodaj korzen

if (root == NULL)

{

root = (wezel\*)malloc(sizeof \*root);

root->wartosc = n;

root->l\_syn = NULL;

root->p\_syn = NULL;

root->rodzic = NULL;

}

//jezeli zadana wartosc jest mniejsza od korzenia idz do lewego poddrzewa

else if(n < start->wartosc)

{

//jezeli lewe poddrzewo istnieje wywolaj dla niego ta funkcje rekurencyjnie

if(start->l\_syn != NULL)

{

dodawanie(n,start->l\_syn);

}

//jezeli lewe poddrzewo nie istnieje dodaj nowy wezel o zadanej wartosci

else

{

wezel \*nowy = (wezel\*)malloc(sizeof \*root);

nowy->wartosc = n;

nowy->l\_syn = NULL;

nowy->p\_syn = NULL;

nowy->rodzic = start;

start->l\_syn=nowy;

}

}

//jezeli zadana wartosc jest wieksza lub rowna korzeniowi idz do prawego poddrzewa

else

{

//jezeli prawe poddrzewo istnieje wywolaj dla niego ta funkcje rekurencyjnie

if(start->p\_syn!=NULL)

{

dodawanie(n,start->p\_syn);

}

//jezeli prawe poddrzewo nie istnieje dodaj nowy wezel o zadanej wartosci

else

{

wezel \*nowy = (wezel\*)malloc(sizeof \*root);

nowy->wartosc = n;

nowy->l\_syn = NULL;

nowy->p\_syn = NULL;

nowy->rodzic = start;

start->p\_syn=nowy;

}

}

return 0;

}

//usun wezel start

void kasowanie(struct wezel \*start)

{

//jezeli wezel nie ma dzieci

if(start->l\_syn==NULL && start->p\_syn==NULL)

{

//jezeli wezel jest korzeniem

if(start->rodzic==NULL)

{

root=NULL;

}

//jezeli wezel jest po lewej stronie rodzica,

else if(start->rodzic->l\_syn==start)

{

//usun wezel z lewej strony wezla rodzica

start->rodzic->l\_syn=NULL;

}

else

{

//usun wezel z prawej strony wezla rodzica

start->rodzic->p\_syn=NULL;

}

free(start);

}

//jezeli wezel ma tylko jedno dziecko

else if(start->l\_syn==NULL || start->p\_syn==NULL)

{

//jezeli po lewej stronie nie ma dziecka

if(start->l\_syn==NULL)

{

//jezeli wezel jest korzeniem

if(start->rodzic==NULL)

{

root=start->p\_syn;

}

//jezeli wezel jest po lewej stronie rodzica

else if(start->rodzic->l\_syn==start)

{

//przyczep z lewej strony rodzica wezel bedacy po prawej stronie usuwanego wezla

start->rodzic->l\_syn=start->p\_syn;

}

else

{

//przyczep z prawej strony rodzica wezel bedacy po prawej stronie usuwanego wezla

start->rodzic->p\_syn=start->p\_syn;

}

}

else

{

//jezeli wezel jest korzeniem

if(start->rodzic==NULL)

{

root=start->l\_syn;

}

//jezeli wezel jest po lewej stronie rodzica

else if(start->rodzic->l\_syn==start)

{

//przyczep z lewej strony rodzica wezel bedacy po lewej stronie usuwanego wezla

start->rodzic->l\_syn=start->l\_syn;

}

else

{

//przyczep z prawej strony rodzica wezel bedacy po prawej stronie usuwanego wezla

start->rodzic->p\_syn=start->l\_syn;

}

}

free(start);

}

else

{

//wstaw w miejsce usuwanego elementu - najmniejsza wartosc z prawego poddrzewa

struct wezel \*temp;

temp=naj\_lewo(start->p\_syn);

start->wartosc = temp->wartosc;

kasowanie(temp);

}

}

//przejdz drzewo w kolejnosci zaczynajac od wezla start

void in\_order\_tree\_walk(struct wezel \*start)

{

if(start->l\_syn != NULL) //jezeli ma dzieci po lewej stronie wywolaj funkcje rekurencyjnie

in\_order\_tree\_walk(start->l\_syn);

printf("%d\n", start->wartosc); //wypisz wartosc

if(start->p\_syn != NULL) //jezeli ma dzieci po prawej stronie wywolaj rekurencyjnie

in\_order\_tree\_walk(start->p\_syn);

}

Stos jest bardzo prostą strukturą w implementacji. W liście jedyną trudnością może być silne korzystanie ze wskaźników i konieczność utrzymania spójności listy podczas operacji takich jak usuwanie czy dodawanie elementów. Znacznie trudniejsza jest implementacja binarnego drzewa decyzyjnego. Dodatkową trudnością jest przede wszystkim konieczność użycia rekurencji. Niezwykle skomplikowany jest również mechanizm usuwania elementu z drzewa, gdy chcemy zachować jego spójność.Zadanie nr 12.

Napisz program scalania ciągów, który polega na łączeniu posortowanych ciągów w jeden ciąg posortowany.

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

void scalanie(char a[], char b[], char c[])

{

int p=0;

int q=0;

int r=0;

while ( (p < strlen(a)) && (q < strlen(b)) ) //poki nie przekraczaja swojego zakresu

{

if (a[p] < b[q]) //przepisz mniejszy na odpowiednia pozycje

{

c[r] = a[p];

r++;

p++;

}

else

{

c[r] = b[q];

r++;

q++;

}

}

while (p < strlen(a)) //przepisz reszte z pierwszej tablicy

{

c[r] = a[p];

r++;

p++;

}

while (q < strlen(b)) //lub z prawej

{

c[r] = b[q];

r++;

q++;

}

c[strlen(a)+strlen(b)]='\0'; //zakoncz tablice

printf("%s\n",c);

system("pause");

}

int main(){

char a[100], b[100], c[200];

gets(a);

fflush(stdin);

gets(b);

scalanie(a,b,c);

return 0;}Zadanie nr 13.

Obliczanie wyrażeń w odwrotnej notacji polskiej.

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <math.h>

#define MAX\_D 256

void die(const char \*msg)

{

fprintf(stderr, "%s", msg);

abort();

}

double stack[MAX\_D];

int depth;

void push(double v)

{

if (depth >= MAX\_D) die("stack overflow\n");

stack[depth++] = v;

}

double pop()

{

if (!depth) die("stack underflow\n");

return stack[--depth];

}

double rpn(char \*s)

{

double a, b;

int i;

char \*e, \*w = " \t\n\r\f";

for (s = strtok(s, w); s; s = strtok(0, w))

{

a = strtod(s, &e);

if (e > s) printf(" :"), push(a);

#define binop(x) printf("%c:", \*s), b = pop(), a = pop(), push(x)

else if (\*s == '+') binop(a + b);

else if (\*s == '-') binop(a - b);

else if (\*s == '\*') binop(a \* b);

else if (\*s == '/') binop(a / b);

else if (\*s == '^') binop(pow(a, b));

#undef binop

else

{

fprintf(stderr, "'%c': ", \*s);

die("unknown operator\n");

}

for (i = depth; i-- || 0 \* putchar('\n'); )

printf(" %g", stack[i]);

}

if (depth != 1) die("stack leftover\n");

return pop();

}

int main(void)

{

char s[] = " 2 3 + ";

printf("%g\n", rpn(s));

return 0;

}

Odwrotna notacja polska wymaga użycia stosu i umiejętności w operowaniu na tablicach znaków.