RĪGAS TEHNISKĀ UNIVERSITĀTE

**Lietišķo datorsistēmu institūts**

**Programmatūras izstrādes tehnoloģijas profesoru grupa**

Datorzinātnes un informācijas tehnoloģijas fakultāte

Datorzinātņu profils

2.kurs DB9. grupa

**Programmas teksta izvade atbilstoši**

**programmēšanas stila prasībām**

Studiju darbs mācību priekšmetā

„Programmēšanas valodas”

Izstrādāja : Sergejs Terentjevs

Pārbaudīja: prof. L. Zaiceva

Pieņēma : dok. N. Prokofjeva

2007./ 2008. Māc. Gads

# ANOTĀCIJA

Studiju darbā gaitā tika izstrādāta programma, kura apstrādā programmēšanas valodas C programmu pirmtekstus atbilstoši programmēšanas stila prasībām.

Studiju darbā pārskatā ir aprakstīti programmēšanas stila prasību pamatnoteikumi, dota programmēšanas stila definīcija, programmēšanas stila piemēri un to apraksti. Ir apskatītas konceptuālas prasības programmai un aplūkoti uzdevumu, speciālu prasību risinājumi, ir sniegts uzdevuma risināšanās algoritma apraksts un shēma, programmas apraksts un lietotāja ceļvedis.

Programma sastādīta valodā C un atkļūdota Borland C++ 3.1 vidē uz Pentium III tipa procesora MS DOS vidē.

Studiju darba pārskata apjoms: 44 lappuses izklāstā tekstā, tajā ietilpst 7 nodaļas, bibliogrāfiskais saraksts, 12 attēli, 1 tabula un 2 pielikumi ar programmas pirmtekstu un izpildes rezultātu piemēriem.

# SATURS

[ANOTĀCIJA 4](#_Toc190026467)

[SATURS 5](#_Toc190026468)

[1. Uzdevuma nostādne 6](#_Toc190026469)

[1.1. Uzdevuma analīze un prasības programmai 6](#_Toc190026470)

[1.2. Uzdevuma risināšanas projektējums 11](#_Toc190026471)

[2. Algoritma apraksts un kopēja shēma 12](#_Toc190026472)

[3. Programmas apraksts 16](#_Toc190026473)

[3.1. Lietojumsfēra 16](#_Toc190026474)

[3.2. Ievaddati 16](#_Toc190026475)

[3.3. Izvaddati 16](#_Toc190026476)

[3.4. Ziņas par programmas apjomu un izpildes laiku 17](#_Toc190026477)

[3.5. Moduļu apraksts 17](#_Toc190026478)

[4. Lietotāja ceļvedis 19](#_Toc190026479)

[5. Programmas atkļūdošanas kļūdu analīze 23](#_Toc190026480)

[6. Programmas funkcionēšanas kontroles piemērs 24](#_Toc190026481)

[7. Secinājumi 25](#_Toc190026482)

[Bibliogrāfiskais saraksts 26](#_Toc190026483)

[Pielikumi 27](#_Toc190026484)

[1. Pielikums 27](#_Toc190026485)

[Programmas pirmteksts 27](#_Toc190026486)

[2. Pielikums 49](#_Toc190026487)

[Programmas izpildes rezultāti 49](#_Toc190026488)

# Uzdevuma nostādne

## Uzdevuma analīze un prasības programmai

Programmēšanas stila prasības jeb pamatnoteikumi ir noteikumu kopums, kas ir jāievēro programmētājam izstrādājot viegli saprotamas un efektīvas programmas. Programmas pirmteksts, kas ir izstrādāts atbilstoši programmēšanas stila prasībām, ir saprotamāks un salasāmāks, kas, savukārt, ir ļoti noderīgi, ja programma turpmāk tiks lasīta un modificēta ar citu programmētāju starpību.

Tātad programmēšanas stils ir metožu kopums, kas paaugstina programmas lasāmību un efektivitāti, ka jebkuram stilam tā arī programmēšanas stilam eksistē vairāki standarti, kurus ir jāizvēlas un pie kuriem ir jāpieturas programmatūras izstrādes gaitā.

Lielos projektos izstrādājot programmatūru, kas tiks lietota ļoti ilgi, ir visnotaļ svarīgi lietot programmēšanas stilu, jo programma turpmāk tiks vairakkārt modificēta un pielāgota valsts tirgus un klientu prasībām. Programma, kas satur komentārus un pareizo struktūru pati var tikt uzskatīta par vērtīgu un nozīmīgu dokumentācijas papildinājumu.

Programmēšanas stila pamatnoteikumi ir šādi:

* Komentāru lietošana.

Komentārs ir paskaidrojums, kas atvieglo programmas struktūru un loģikas izpratni. Komentāru lietošana atvieglo programmas izpratni un palīdz samazināt laika patēriņu programmas atkļūdošanai un modificēšanai.

Rakstot programmu ir vēlams lietot 2 veidu komentārus :

* ievadkomentārus;
* paskaidrojošos komentārus.

Par ievadkomentāriem uzskata prologus, kuros iekļauj informāciju par programmas nosaukumu, lietojumu, programmas izveides vai modificēšanas datumu, ziņas par autoru un programmas izpildes laiku, parametru un mainīgo sarakstu, lietotu apakšprogrammu sarakstu u.tml.

Savukārt, paskaidrojošie komentāri ir nepieciešami programmas bloku lasāmības atvieglošanai. Paskaidrojošie komentāri satur papildinformāciju par kādu attiecīgu programmas fragmentu, tie tiek lietoti funkciju, datu un procedūru aprakstam.

Komentāru piemēri valodā C:

1. /\*---------------------------------------------

Adreses iegūšana

----------------------------------------------\*/

1. *int Ch = getch();* // vērtības piešķiršana mainīgam

* Nosaukumu pareiza izvēle.

Bieži vien programmas lasāmību un izpratni paaugstina parieza nosaukuma piešķiršana funkcijām, failiem, moduļiem un objektiem. To nosaukumiem ir jāatspoguļo apzīmējuma lielumu un procesu dabu. Lai pareizi izvēlēties nosaukumu ir jāpieturas pie sekojošiem paņēmieniem:

* Jāizmanto viena un tā pati dabīgā valoda visu nosaukumu izvēlei (piemēram: *Teksts* un *Simbols*),
* Nav jāizvēlas līdzīgus nosaukumus dažādiem objektiem,
* Vesela tipa mainīgo nosaukumus ieteicams sākt ar burtiem I, J, K, L, M vai N,
* Vēlams lietot saīsinājumus gariem nosaukumiem. Šajos gadījumos nosaukumā ir jābūt vārda pirmajam burtam un ir pieļaujama patskaņu izlaišana ( piemēram: *ConnStr*, nozīme „*Connection String*” ).
* Izstrādājot programmu Windows vidē, ka par standartu var lietot ungāru pierakstu, kas ļauj izveidot vidēja garuma informatīvus nosaukumus. Objekta nosaukums tiek veidots no īsa prefiksa un objekta vārda, kas atspoguļo objekta dabu (piemēram: *szSymbol*, kur *sz* apzīmē simbolu virkni).
* Datu aprakstu grupēšana.

Mainīgo un citu objektu aprakstiem jābūt sagrupētiem un strukturētiem atbilstoši objektu tipiem un to funkcijām programmā. Ir ietiecams datus sakārtot alfabētiska secība, izmantot atkāpes un katru nosaukumu rakstīt savā rindā, komentējot to.

Datu grupēšanas piemērs valoda C:

*struct address*

*{*

*char name [30]; /\* pilsoņa vārds \*/*

*char street [40]; /\* ielas adrese \*/*

*char city [15]; /\* pilsētas nosaukums \*/*

*unsigned long int zip; /\* zip kods \*/*

*};*

* Tukšu rindu lietošana.

Tukšu rindu lietošana padara programmu vizuāli vieglāk uztveramu, jo to lietošana ļauj sadalīt programmu loģiskos blokos un paragrāfos. Pieņemts atstāt :

* 3 tukšas rindas starp moduļu pamatdaļām,
* 2 tukšas rindas starp pārējam moduļa daļām
* 1 tukšu rindu pēc katras operatoru virknes.
* Atkāpju lietošana.

Atkāpes dod iespēju uzskatāmi attēlot programmas loģisko struktūru, sadalīt to paragrāfos. Atkāpes tiek lietotas cikla, nosacījumu un bloku konstrukcijas, parasti atkāpju garums ir 2 – 5 pozīcijas no iepriekšējās rindas sākuma pozīcijas.

Atkāpju lietojumu piemērs valoda C:

if( Spaces > 0 )

{

for(i = 0; i < strlen(szString)-Spaces; i++)

{

szString[i] = szString[i + Spaces];

}

szString[i]='\0';

}

* Viena operatora rakstīšana viena rinda.

Viena operatora rakstīšana viena rindā paaugstina programmas lasāmību un atvieglo atkļūdošanas procesu.

* Tukšumzīmju lietošana.

Tukšumzīmju lietošana palielina programmas lasāmību šādos gadījumos:

* Pirms un pēc operāciju zīmēm ( piemēram: *a > b ? a : b* ),
* Starp ievadizvades saraksta elementiem ( piemēram: *printf ( „ %s ”, szString )* ),
* Starp parametriem.

Ņemot vēra, ka programma nevar realizēt dažus punktus no programmēšanas stila prasībām, piemēram, automātiski koriģēt mainīgo un objektu nosaukumus, programma, galvenokārt, tiks nodrošinātas šādas programmēšanas stila prasības:

* Komentāri tiks rakstīti pirms katra bloka, t.i., pirms cikla, nosacījuma un { - } konstrukciju blokiem.
* Automātiski tiks pievienoti komentāri pirms cikla operatoriem ( *// Cikla komentārs* ), nosacījumu operatoriem ( *// Nosacījuma komentārs* ) un funkcijām, kas izskatīsies sekojoši:

*/\*------------------------------------------------*

*Funkcijas apraksts*

*------------------------------------------------\*/*

*void fun1 ( int mas[ ], int newmas[ ], int n )*

*{*

*/\* funkcijas ķermenis \*/*

*}*

Jā komentārs jau eksistē pirms minētam konstrukcijām ( */\* Siet ir jau komentārs \*/* vai arī *// Siet ir jau komentārs* ), tad programma automātiski izanalizēs un nepievienos klāt komentāru. Programma spēs novērtēt vai dota funkcija nav funkcijas prototips jeb definējums.

* Vienā rinda tiks atstāts tikai viens operators.
* Tiks lietotās atkāpes pirms { - } , ciklu un nosacījumu konstrukcijām.
* Tiks lietotas tukšas rindas atbilstoši programmēšanas stila prasībām.

Tagad var formulēt prasības programmai, kas veic programmas teksta izvadi atbilstoši programmēšanas stila prasībām. Programmai ir jānodrošina šādas funkcionālas prasības :

1. Programmai ir jāveic norādīta faila atvēršanu un ielasīšanu. Lietotājs ievades laukā norāda faila nosaukumu un pilno ceļu līdz failam ( ja fails neatrodas viena direktorijā ar programmu). Failam ir jāsatur programmēšanas valodas C programmēšanas stilam neatbilstošu pirmtekstu. Norādot faila nosaukumu var arī nenorādīt faila paplašinājumu, tad programma pievienos klāt paplašinājumu .c. Jā fails netiks atrasts vai arī atvērtais fails nesaturēs valodas C pirmtekstu, programma izvadīs attiecīgus brīdinājuma paziņojumus - *Fails ir veiksmīgi atvērts. Iespējams, ka fails nesatur valodas C pirmtekstu*, *Neizdevās atvērt failu, pārliecinieties, ka tas eksistē*, citādi izvadīs paziņojumu *Fails ir veiksmīgi atvērts*.
2. Programmai ir jānodrošina atbilstoši programmēšanas stila prasībām apstrādāta pirmteksta saglabāšanu jaunajā faila. Lietotājam ir jānorāda rezultējoša faila nosaukumu, kas ir izdarāms otrajā ievades lauka. Norādot faila nosaukumu, kura glabāsies apstrādātais teksts, ir jānorāda arī paplašinājums, jo pēc noklusējuma programma nepiešķir nekādu paplašinājumu failam. Lietotājam ir dota iespēja norādīt konkrēto direktoriju un ceļu, kura fails tiks saglabāts, vai arī norādot tikai nosaukumu, fails saglabāsiet tāda paša direktorija, kur atrodas programma. Gadījuma ja noradītais fails jau eksistē programma izvadīs paziņojumu - *Fails jau eksistē, Enter - pārrakstīt failu*, citādi izvadīs paziņojumu *Fails ir veiksmīgi atvērts*.
3. Programmai ir jāveic pirmteksta apstrādi - jāpārnes operatorus jauna rindā, jānodrošina atkāpes, jānodrošina tukšas rindas u.tml.
4. Pēc lietotāja vēlēšanas programmai ir jāizvada rezultējoša faila saturs ar apstrādātu atbilstoši programmēšanas stilam valodas C pirmtekstu.
5. Programmai ir jāizvada statistika par ielasīta faila pirmteksta sākuma rindu skaitu, pēc pirmteksta apstrādes pievienoto rindu skaitu, pievienoto komentāru skaitu u.t.t.

Programmas izstrādes gaita jānodrošina šādas nefunkcionālas prasības:

1. Lietotāja saskarnei ar programmu jānotiek latviešu valodā.
2. Programmai jāsastāv no vairākām funkcijām, katra no kurām ir jāglabā atsevišķā failā.
3. Ziņojumu izvade, lietotāja nepareizas darbības rezultāta.
4. Programmai jāsastāv no vairākām funkcijām, kuras ir jāglabā dažādos failos (vismaz divos failos).
5. Programmēšanas valoda – C.

## Uzdevuma risināšanas projektējums

Izstrādātai programmai ir jāveic 3 pamatdarbības: ielasīt faila pirmtekstu valoda C, veikt pirmteksta apstrādāšanu atbilstoši programmēšanas stilam, veikt apstrādāta programmas teksta izvadīšanu uz ekrāna. Tātad programmas struktūra ietver sevī trīs moduļus ( 1.a. att.) :

* Failu nosaukumu ievade un paziņojumu izvade,
* Faila apstrāde,
* Teksta izvade uz ekrānā.

Bez minētajām funkcijām programma pilda vēl divas papildus funkcijas, t.i., apstrādāta pirmteksta saglabāšanu rezultējoša faila un statistikas apkopošanu un izvadi, tādēļ otru bloku programmas struktūra ir jāsadala vēl divas funkcijas. Tāda kārtā programmas arhitektūra satur 5 moduļus ( 1.b. att.):

* Failu nosaukumu ievade un paziņojumu izvade (main),
* Faila pirmteksta analīze (AnalizeText),
* Teksta saglabāšana rezultējoša faila (AnalizeText),
* Statistikas izvade (main),
* Teksta izvade uz ekrānā (main).

Failu nosaukumu ievade un paziņojumu izvade

Faila apstrāde

Teksta izvade ekrānā

Failu nosaukumu ievade un paziņojumu izvade

Faila pirmteksta analīze

Teksta izvade ekrānā

Statistikas izvade

Rezultāta saglabāšana

a) Programmas struktūras

sākumvariants.

b) Precizēta programmas

struktūra.

1. att. Programmas struktūra.

# Algoritma apraksts un kopēja shēma

Programmas struktūra ietver trīs pamatfunkcijas : main, ReadInputString un AnalizeText.

Funkcijas **main** algoritms

* Programmas atvelētājos ievadlaukos jāievada faila un rezultējoša faila nosaukumi un ja nepieciešams jānorāda faila, kas satur programmēšanas stilam neatbilstošu pirmtekstu, atrašanas vieta. Ievadi nodrošināta *ReadInputString* funkcija.
* Jā noradīta informācija ir nekorekta vai arī noradītais fails neeksistē , vai nu rezultējošais fails jau eksistē un lietotājs negrib pārrakstīt esošo failu, programmas algoritms atgriežas uz 1.etapu, t.i., uz failu nosaukumu ievadīšanas.
* Savukārt, jā ir ievadīta korekta informācija vai arī lietotājs ir nolēmis pārrakstīt esošo rezultējošo failu, programmas algoritms izsauc funkciju *AnalizeText*, kurai tiek nodota struktūra *szDescriptor* statistikas apkopošanai un pirmteksta koriģēšanai atbilstoši programmēšanas stilam un pirmteksts, ko ielasa funkcija *ReadString* no faila.
* Pēc funkcijas *AnalizeText* izpildes, faila pirmteksts ir apstrādāts atbilstoši programmēšanas stilam un tas ir saglabāts rezultējoša faila.Uz ekrāna tiek izvadīta statistika, kura tika apkopota *AnalizeText* funkcijas izpildes gaitā, kas, savukārt, tika apkopota ar struktūras *szDescriptor* mainīgo starpību.
* Turpmāk lietotājam tiek dota brīva izvēle vai aplūkot rezultējoša faila saturu jeb nē.
* Jā lietotājs izvēlas aplūkot rezultējoša faila saturu, tad uz ekrānā tiek izvadīts apstrādātais pirmteksts.
* Programmas aizvēršanai jānospiež taustiņš *Esc*.

Funkcijas *main* algoritma shēma ir paradīta 14.lpp. 2. Attēlā.

Funkcijas **ReadInputString** algoritms

Funkcija nodrošina teksta ievadīšanu, koriģēšanu un simbolu dzēšanu ievadlaukos, kuros lietotājs ievada faila nosaukumus.

Funkcijas algoritms balstās uz taustiņu nolasīšanu, kurus nospiež lietotājs, tādejādi pie noteikta taustiņa nospiešanas, piemēram, *Backspace*, funkcija izpildīs kādu no noteiktajām darbībām, mūsu gadījumā nodzēsis vienu simbolu pirms kursora u.tml. Otrais un galvenais funkcijas uzdevums ir attēlot lietotāja ievadīto informāciju ievadlaukos, tas tiek realizēts izsaucot funkciju *drawTextBox*. Funkcijas *ReadInputString* algoritma shēma 15.lpp. 3. Attēlā.

Funkcijas **AnalizeText** algoritms

* Tiek analizēts tekošais teksts, kura pirmkārt sameklē atdalītājus, t.i., simbolus ‘ {’, ‘}’ un ’;’. Ja atdalītājs tiek sameklēts, tad tajā rindas vietā, kurā tika atrasts atdalītājs, rinda tiek atdalīta no parēja teksta un tiek padota turpmākai analīzei. Funkcija spēj noteikt, kad kāds no simboliem netiek uzskatīts par atdalītāju, piemēram, dotajā piemēra *for* cikla simbols ‘;’ netiek uzskaitīts par atdalītāju.

*// Cikla komentārs*

*for (i=0; i < number; ++i)*

*printf(" %d ",array[i]);*

* Turpmāk teksta tiek meklēta makrokomanda *#include* un simbols ‘>’, ja aiz minēta simbola seko teksts, tad tas tiek pārnests jaunajā rinda.
* Ja tiek sameklēts simbols ‘{’ un tas ir pirmajā pozīcijā, ja aiz šī simbola seko teksts, tad aiz dota simbola teksts tiek pārnests jaunajā rinda.
* Visa funkcijas algoritma darbības etapos, tiek meklēti atslēgvārdi „*case*”, „*break*”, simboli ‘{‘, ‘}’ u.tml. Ja kāds no dotajiem simboliem vai vārdiem tika atrasts, programma, analizējot pēc atrašanas vietas, palielina vai pamazina atkāpes.
* Ja teksta tiek sameklēts kāds no cikla operatoriem vai arī nosacījuma operatoriem, piemēram, *while* vai *for*, programma nosaka vai pirms operatora ir jau pievienots kāds komentārs, ja tā nav, tad programma pievieno klāt komentāru - *// cikla komentārs* vai *//nosacījuma komentārs*. Turpmāk programma analīze operatora pozīciju teksta, t.i., vai aiz operatora seko bloks vai nē, vai tas ir jau pārnests jaunajā rindā, vai gluži otrādi aiz operatora seko teksts, programma izpildīs darbību, kas būs piemērota dotajai situācijai.
* Līdzīgi notiek, ja programma sameklē atslēgas vārdu „*else*”, tikai šajā gadījuma programma nepievieno klāt komentārus.
* Ja teksta analizēs gaita tika sameklēta funkcija, tad pirms funkcijas tiek ievietots komentārs, savukārt, ja dotais komentārs ir jau pievienots, funkcija nepievienos komentāru.
* U.tml.
* Apstrādāta rinda tiek ierakstīta faila, turpmāk apstrādei tiek padots nākošais teksts, kas tika atdalīts u.tml. līdz tiek ielasīts viss faila pirmteksts.

Funkcijas algoritma shēma nav dota šinī pārskata, sava liela apjoma dēļ (funkcija sastāv apmēram no 426 rindām, tās sastāvā liels nosacījumu un cikla operatoru sk., kurus nav iespējams saīsināt, paradot algoritma darbības loģiku ).

Faila nosaukumu ievade

ResultFileExist

Rezult. fails jau eksistē vai fails netika atrasts

Ievadīta korekta inf.

ReadString

Aplūkot faila saturu

DrawListBox

AnalizeText

nē

jā

jā

nē

2. att. funkcijas **main** algoritma shēma.

0

0

0

0

0

0

0

1

1

1

1

1

1

1

1

Nospiests Backspace

Pāriet uz jaunu ievadlauku

Ievade pabeigta

atstarpes iestarp.

Beigt ievadi

Nospiests Escape

Nospiests TAB

Nospiests Enter

Nospiests <- taustiņ.

Nospiests -> taustiņ.

Pārvietoties pa kreisi

Pārvietoties pa labi

Kursora poz. noteikšana

Simbolu ievade

Simbols ir speciāls taustiņš

Nospiests Backspace

Dzēst simbolu

drawTextBox

3. att. funkcijas **ReadInputString** algoritma shēma.

# Programmas apraksts

## Lietojumsfēra

Programma paredzēta valodā C uzrakstītu programmu pirmtekstu apstrādāšanai atbilstoši programmēšanas stila prasībām.

Programma sastādīta valodā C un atkļūdota Borland C++ 3.1 vidē uz Pentium III tipa procesora MS DOS vidē.

Programmas parieza darbība ir pārbaudīta, veicot vairāku valodas C pirmtekstu apstrādāšanu.

## Ievaddati

Lietotājam ir jāievada apstrādājama faila nosaukumu speciālajā tam atvelētājā ievadlaukā, ja fails neatrodas tanī paša direktorija, kur atrodas programma, tad obligāti nepieciešams noradīt arī faila atrašanas vietu ( piemēram: c:**\**opensource**\**array.cpp ). Faila nosaukumam var arī nenoradīt faila paplašinājumu, tad programma pēc noklusējuma pievienos klāt paplašinājumu .c.

Otrkārt, lietotājam ir jānorāda rezultējoša faila nosaukums, kurš tiks izveidots uzreiz pēc faila nosaukuma ievadīšanas un kurā tiks saglabāts programmas apstrādāts atbilstoši programmēšanas stilam pirmteksts. Pēc noklusējuma fails tiks izveidots tanī paša direktorija, kur atrodas programma, bet lietotājam ir iespēja saglabāt failu cituviet, tad lietotājam ir jānorāda pilns ceļš līdz mapei vai arī diskam, kur fails tiks ieglabāts ( piemēram: c:**\**out.txt). Šeit obligāti ir jānorāda rezultējoša faila paplašinājumu, jo programma pēc noklusējuma nepiešķirs nekādu paplašinājumu failam.

## Izvaddati

1. Ziņojums: Neizdevās atvērt failu, pārliecināties, ka tas eksistē.
2. Ziņojums: Neizdevās izveidot failu, pārliecināties, ka nosaukums it pareizs.
3. Fails jau eksistē, Enter – pārrakstīt failu.
4. Fails ir veiksmīgi atvērts.
5. Fails ir veiksmīgi atvērts. Iespējams, ka fails nesatur valodas C pirmtekstu.
6. Statistikas izvade ( pirmteksta rindu, simbolu, apstrādātu un pievienotu rindu, pievienotu komentāru skaits ).

## Ziņas par programmas apjomu un izpildes laiku

Funkcija *main*: MAIN.C - 8.67 KB

MAIN.OBJ - 29.2 KB

Funkcija *AnalizeText*: FILE.H - 16.1 KB

FILE.OBJ - 14.3 KB

Funkcija *ReadInputString*: SHAPE.H - 6.43 KB

SHAPE.OBJ - 7.21 KB

Izpildāmais fails MAIN.EXE - 73.9 KB

Programmas izpildes laiks ir atkarīgs no apstādāmo faila izmēriem ( fails, kas saturēja 89 rindas tika apstrādāts 0.31 sek. laikā ).

## Moduļu apraksts

Programma sastāv no trim moduļiem ( 4. Att. ):

szString, szDescriptor

szString, szDescriptor

AnalizeText

main

x, y, cx, iText

chNewChar

ReadInputString

1. att. Starp moduļu interfeisa shēma.

**Modulis main ( int argc, char \*argv[] )**

Funkcija : nodrošina failu nosaukumu ievadi, statistikas un rezultējoša faila izvadi uz ekrānā, grafikas pozicionēšanu un lietotāja pieprasījumu apstrādi.

Parametri: funkcijai ir divi parametri, kas ir komandrindas argumenti.

*int argc* – satur komandrindas argumentu daudzumu jeb ir rādītāju masīvā *\*argv [ ]* izmērs.

*char \*argv [ ]* – komandrindas radītāju masīvs.

Saikne ar lietotāju: izsauc vairākas funkcijas, kuru darbības rezultāta notiek grafisko interfeisu un ziņojumu izvade uz ekrānā.

**Modulis ReadInputString (int x, int y, int cx, int \*iText)**

Funkcija: nodrošina teksta ievadīšanu, koriģēšanu un simbolu dzēšanu ievadlaukos, kuros lietotājs ievada faila nosaukumus.

Parametri: funkcijai ir četri parametri x, y, cx un iText.

*int x* – ievadlauka kreisā augšēja stūra koordināte pa x asi,

*int y* – ievadlauka kreisā augšēja stūra koordināte pa y asi.

*int cx* - ievadlauka garums,

*char \*iText* – rādītāju masīvs, kurš glabā lietotāja ievadītos simbolus.

Saskarne ar lietotāju: ievadīto simbolu izvadīšana uz ekrāna un pieprasīto darbību izpildīšana.

**Modulis AnalizeText ( char \*szString, struct SourceFileDescription \*szDescriptor )**

Funkcija: veikt pirmteksta analīzi un apstrādāt to atbilstoši programmēšanas stila prasībām, veikt apstrādāta pirmteksta saglabāšanu failā.

Parametri: funkcijai ir divi parametri *szString* un *szDescriptor* :

*char \*szString* – no faila ielasīta pirmteksta rinda,

*struct SourceFileDescription \*szDescriptor* – struktūra, kuras mainīgie satur pirmteksta apstrādei nepieciešamu informāciju. Apskatīsim struktūru:

*long JointlyRows* - ielasīta faila rindu skaits;

*long CreatedLines* – izveidoto rindu skaits rezultējoša failā;

*long JointlyBytes* – ielasīta faila simbolu skaits;

*char chLastSeparator* – pēdējais atdalītais apstrādātajā rindā;

*int iSingleOffset* – specifikators, kurš nosaka atkāpes jeb nobīdes pēc konstrukcijām;

*int iCurrentOffset* – specifikators, kurš nosaka tekošo nobīdi tekstā;

*int CreatedComments* – izveidoto komentāru skaits;

*int iSetOffset* – specifikators, kurš norāda atkāpju nepieciešamību;

*int iComment* – specifikators, kurš nosaka komentāru esamību.

Rezultāts: tiek iegūts programmēšanas stilam atbilstošs teksts.

Saikne ar lietotāju: nav.

Moduļi tiek glabāti dažādos failos (1.tab.). Programmas pirmteksts dots 1. pielikumā.

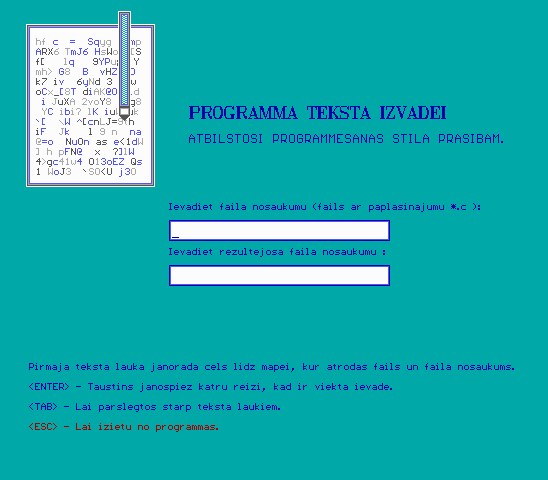
|  |  |  |
| --- | --- | --- |
| Modulis (funkcija) | Pirmteksta fails | Objektfails |
| main ( int argc, char \*argv[] )  AnalizeText ( char \*szString, SourceFileDescriptor \*szDescriptor)  ReadInputString ( int x, int y, int cx, char \*iText ) | MAIN.CPP  FILE.H  SHAPE.H | MAIN.obj  FILE.obj  SHAPE.obj |

1.tabula Moduļi un faili.

# Lietotāja ceļvedis

Programmas izpilde notiek ar faila MAIN.EXE palaišanu.

Pēc programmas palaišanas uz ekrānā parādās galvenā programmas forma, kurā tiek attēloti divi ievadlauki ( skat. att.5 ).

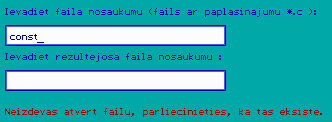


1. att. programmas interfeiss.

Virs ievadlaukiem tiek attēloti divi paziņojumi, kuri satur lietotājam paredzētas norādes par faila nosaukumu ievadi:

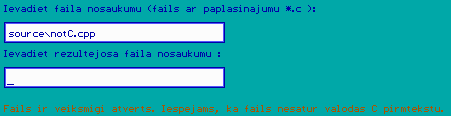
Paziņojums „ *Ievadiet faila nosaukumu ( fails ar paplašinājumu \*.c ) :* ”

Paziņojums norāda uz faila nosaukuma ievades nepieciešamību (sīkāk skat. 9, 14, 16. lpp). Dotajām failam ir jāsatur valodas C neatbilstošs programmēšanas stilam pirmteksts. Obligāti ir jānorāda faila nosaukums, citādi programma neveiks nekādas turpmākas darbības. Jā noradītais fails neeksistē vai arī tika nepariezi norādīts ceļš līdz direktorijai, kurā atrodas fails, tad programma izvada paziņojumus *Neizdevās atvērt failu, pārliecināties, ka tas eksistē* (skat. att. 6).



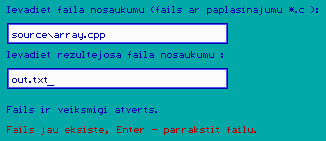
1. att. nekorekta informācija.

Savukārt, ja lietotājs norāda failu, kas nesatur valodas C pirmtekstu, tad programma izvada paziņojumu *Fails ir veiksmīgi atvērts. Iespējams, ka fails nesatur valodas C pirmtekstu* ( skat. att. 7 ).



1. att. brīdinājuma paziņojums.

Ja lietotājs ir noradījis korekto informāciju, tad programma izvada paziņojumu par faila veiksmīgu atvēršanu – *Fails ir veiksmīgi atvērts* ( skat. att. 8 ).



1. att. paziņojumu izvade.

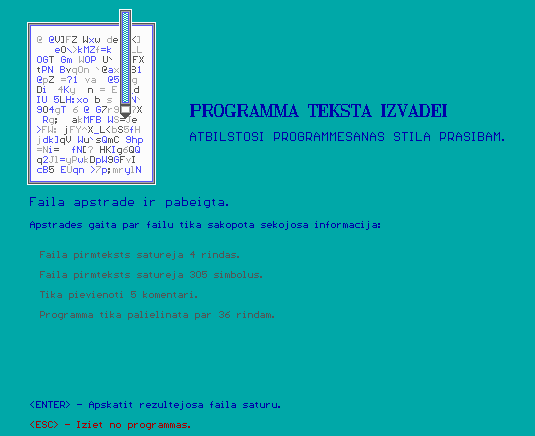
Paziņojums „*Ievadiet rezultējoša faila nosaukumu:*”

Dotais paziņojums norāda uz rezultējoša faila nosaukuma ievades nepieciešamību (sīkāk skat. 9, 14, 16 lpp.). Rezultējoša faila nosaukuma ievade un faila paplašinājuma norāde ir obligāta. Ja lietotājs norāda faila nosaukumu, kurš ir jau piešķirts kādam noteiktajām failam, kurš atrodas tanī paša direktorija, kur glabāsies izveidotais fails, programma izvadīs brīdinājuma paziņojumu *Fails jau eksistē, Enter – pārrakstīt failu* (skat. att. 8).

Savukārt, ja lietotājs ievades lauku atstāj tukšu un spiež *Enter* programmas turpmākai darbībai, tad programma izvada paziņojumu *Neizdevās izveidot failu, pārliecināties, ka nosaukums ir pareizs*.

Ja lietotājs ir ievadījis korekto informāciju, programma izvada paziņojumu *Fails ir veiksmīgi atvērts* un pāriet uz nākamo programmas izpildes etapu ( abos ievadlaukos jābūt noradītai informācijai ).

Nākošajā programmas izpildes etapā programma veic noradīta faila pirmteksta apstrādāšanu atbilstoši programmēšanas stila prasībām. Pēc apstrādes pabeigšanas uz ekrāna tiek izvadīta statistika par apstrādāto pirmtekstu ( skat. att. 9 ).



1. att. statistika par failu *const.cpp.*

Turpmāk lietotājam ir dota izvēle vai aplūkot rezultējoša faila saturu vai arī nē, t.i., *Enter* taustiņš aplūkot rezultējoša faila saturu, *Esc* taustiņš iziet no programmas (skat. att. 9). Ja lietotājs ir nospiedis kādu neatbilstošu prasībai taustiņu, tad programma izvadīs paziņojumu *Tika ievadīta nekorekta informācija, lūdzu mēģiniet vēlreiz*.

Ja lietotājs izvēlas aplūkot rezultējoša faila saturu, tad programma pāriet nākama izpildes etapā un izvada rezultējoša faila saturu uz ekrānā (skat. att. 10). Lietotājam ir dota iespēja aplūkot visa faila saturu nospiežot attiecīgus taustiņus uz augšu un uz leju, tad atbilstoši programmas rīkjosla, kura nosaka teksta pozicionēšanu, pārvietosies arī vai nu uz augšu vai arī uz leju.



1. att. apstrādāta *array.cpp* pirmteksta saturs.

Programmas aizvēršanai jānospiež taustiņš *Esc*.

# Programmas atkļūdošanas kļūdu analīze

Programmas izstrādāšanas gaitā tika pielaistas dažas sintaktiskais kļūdas:

* Izlaists semikols un pēdiņas;
* Tika vairakkārt izlaista iekāva loģisko operāciju aizvēršanai;
* Tika otrādi padoti mainīgie funkcijai, piemēram, *fputs ( „\n”, hResultFile )* ir pariezi, bet mana gadījuma *fputs ( hResultFile, „\n”)* ir nepariezi, sajaukums bija noticis paralēli lietojot funkciju *fprintf*, kur šāda mainīgo padošanas secība ir parieza;
* Bija aizmirsies pirms kompilācijas nodefinēt funkcijas prototipu;
* Tika izlaista figūriekava bloku aizvēršanai;
* u.tml.

Par nozīmīgam kļūdām kompilācijas laikā var uzskatīt:

* Par būtisku neērtību var uzskatīt Borland C++ 3.1 vidēs neērtību lielu programmatūru izstrāde, t.i., ja dotajai programmai ir liels pirmteksta rindu skaits. Lielu neērtību sagādā tieši tāds fakts, ka Borland C++ 3.1 vidē nav iestrādāta atkļūdotāja, programmētājs nevar redzēt savas programmas izpildi pa soļiem, kas, savukārt, samazinātu kļūdu pielaišanu programmas izstrādes laikā un ļautu ietaupīt laiku programmas atkļūdošanai. Lai atkļūdotu nopietnas kļūdas es biju izmantojis sevis izveidotu failu, ka atkļūdotāju, kura ierakstīju mainīgo vai funkciju izpildes vērtības, tas izskatījās sekojoši:

*FILE \*e;*

*e = fopen („Debug.txt”, „a+”);*

*fprintf (e, „ %s ”, szString );*

*fclose (e);*

* Nozīmīgas kļūdas tika pielaistas nezinot valodas C īpatnības, piemēram operatoru „ **.**” un „ **->** ” nozīmi struktūras. Abi dotie operatori nodrošina pieeju pie struktūras elementiem, bet operators „ *.*” tiek lietots pie tiešas saites uz struktūras elementu, savukārt operators „**->**” tiek izmantots, ka radītājs uz struktūru. Piemērs:

*struct employee*

*{*

*float wage;*

*} emp;*

*Struct employee \*p = &emp;*

*emp.wage = 13.06;*

*p -> wage = 13.06;*

# Programmas funkcionēšanas kontroles piemērs

Programmas pārbaudei tika ievadīti neeksistējoši un eksistējoši failu nosaukumi, tika noradīts fails, kurš tika jau izveidots un fails, kas nesatur valodas C pirmtekstu. Tika ielasīti vairāki faili ar valodas C neatbilstošo programmēšanas stila prasībām pirmtekstu un pārbaudīts programmas apstrādes veiktais rezultāts.

* Noradot failu, kura pirmteksts ir neatbilstošs programmēšanas stilam, programma pārnes makrokomandu *#include <bibliotēka>* jaunajā rinda ja tā, protams, neatrodas jaunajā rinda.
* Jā norādītāja faila pirmteksta ir gara rinda, kura atrodas liels daudzums semikolu, tad programma uzskata tos par atdalītājiem un pārnes rindas aiz atdalītājiem jaunajās rindās.

Protams, ir daži izņēmumi, piemēram, kad dotais atdalītājs atrodas *for* cikla operatora ietvaros.

* Līdzīgi notiek ar figūriekavām, kuras arī tiek uzskatītas par atdalītājiem, bet ir daži izņēmumi, piemēram, ja dota figūriekava ir jau pārnesta jaunajā rinda.
* Jā tiek sameklēts nosacījuma vai cikla operators un tiem nav piešķirts paskaidrojošais komentārs programma, programma piešķir komentārus.
* Līdzīgi ir ar funkcijām. Programma spēj atšķirt, kad ir atrasta funkcija, bet kad ir atrasts funkcijas prototips.
* Ja pēc nosacījuma vai cikla operatora viena un tā paša rinda seko teksts, kura nav atdalītāju, programma pārnes tekstu jaunajā rindā un piešķir tam nobīdi.
* u.tml.

Tātad noradot programmai dažādus valodas C programmu pirmtekstu failus, tika pārbaudīti visi programmas zari un iegūti korekti rezultāti.

# Secinājumi

Studiju darba izpildes gaitā ir izpildīta uzdevuma analīze, formulētas prasības programmai, izstrādāts algoritms un valodā C izveidota programma valodas C pirmtekstu apstrādāšanai atbilstoši programmēšanas stila prasībām. Programma atkļūdota un izpildīta uz *Compaq Deskpro* tipa datora, iegūtie rezultāti pilnīgi atbilst paredzētajiem.

Studiju darbs bija diezgan apjomīgs. Tā izpildei bija nepieciešamās 59 stundas, no kuram 15 stundas aizņēma uzdevuma analīze un pārskata noformēšana.

Izveidoto programmu var izlabot: var pievienot \*.c vai \*.cpp failu meklēšanas algoritmu un faila izvēles iespēju no atrasta katalogā. Noteikti var uzlabot *AnalizeText* funkciju, analizējot vēl nopietnāk valodas C īpatnības un sintaksi.

Studiju darba izpilde deva iespēju apgūt programmēšanas valodu C un tās sintaksi, ka arī veicināja programmēšanas prasmju un iemaņu attīstību.

# Bibliogrāfiskais saraksts

1. G. Matisons, L. Zaiceva, N. Prokofjeva. Studiju darbu izstrāde. – Rīga, RTU, 2003. – 56.lpp.
2. L. Zaiceva. Programmatūras izstrādes tehnoloģija. – Rīga, RTU, 2002. – 161.lpp.
3. Разработка Бездефектного Программного Обеспечения (Zero-Defect Software Development, ZDSD). <http://www.rsdn.ru/article/Methodologies/zdsd.xml>
4. Герберт Шилдт. Полный справочник по С 4-е издание. - Москва, Санкт-Петербург, Киев, Издательский дом «Вильямс», 2007 – 216.lpp.

# Pielikumi

## Pielikums

## Programmas pirmteksts

**Fails main.cpp**

#include <stdio.h>

#include <conio.h>

#include <string.h>

#include <stdlib.h>

#include <graphics.h>

#include <dos.h>

#include "shape.h"

#include "file.h"

#define MAXSTR 512

long LongMin(long a,long b){

if(a>b) return b;

return a;

}

long LongMax(long a,long b){

if(a>b) return a;

return b;

}

void main(int argc, char \*argv[])

{

int errorcode, step, input, input1 = 0, input2 = 0, current = 0, Exist;

char \*szTemp, szSourceString[MAXSTR], ch;

char szSourcePath[MAXPATH] = {NULL}, szResultPath[MAXPATH] = {NULL},szLastPath[MAXPATH] = {NULL};

SourceFileDescription szDescriptor;

double dbProgress;

errorcode = initGraphic();

if (errorcode == grOk){

for (step = 0; step < 3; step++){

setBackground(3);

switch(step)

{

case 0:

drawPicture(10,15,128,162);

drawTitle();

settextstyle(SMALL\_FONT,HORIZ\_DIR,4);

outtextxy(170,200,"Ievadiet faila nosaukumu (fails ar

paplasinajumu \*.c ):");

drawTextBox(170, 220, 220,"");

outtextxy(170,245,"Ievadiet rezultejosa faila nosaukumu :");

drawTextBox(170, 265, 220, "");

szTemp = strrchr(argv[0],'\\');

if( szTemp != 0 )

{

\*(szTemp+1) = '\0';

}

outtextxy(30,360,"Pirmaja teksta lauka janorada cels lidz

mapei, kur atrodas fails un faila nosaukums.");

outtextxy(30,380,"<ENTER> - Taustins janospiez katru reizi, kad ir viekta ievade.");

outtextxy(30,400,"<TAB> - Lai parslegtos starp teksta laukiem.");

setcolor(RED);

outtextxy(30,420,"<ESC> - Lai izietu no programmas.");

while( (input1 != 1) || (input2 != 1) ){

if( current == 0 ) input = ReadInputString(170, 220, 220, szSourcePath);

else input = ReadInputString(170, 265, 220, szResultPath);

if( input == -1 ) exit(0);

if( input == 0 ){

if( CloseSourceFile() )

{

input1 = 0;

setcolor(3);

outtextxy(170,300,"Fails ir vieksmigi

atverts.");

outtextxy(170,300,"Fails ir veiksmigi atverts. Iespejams, ka fails nesatur valodas C pirmtekstu.");

}

if( CloseResultFile() )

{

input2 = 0;

setcolor(3);

outtextxy(170,320,"Fails ir

vieksmigi atverts.");

}

current = (current == 1)?0:1;

drawTextBox(170, 220, 220, szSourcePath);

drawTextBox(170, 265, 220, szResultPath);

}

if( input == 1 ){

if( current == 0 ){

if(OpenSourceFile( argv[0], szSourcePath ) ){

current = 1;

input1 = 1;

drawTextBox(170, 220, 220,

szSourcePath);

setcolor(3);

outtextxy(170,300,"Neizdevas

atvert failu, parliecinieties, ka tas eksiste.");

outtextxy(170,300,"Fails ir

veiksmigi atverts. Iespejams, ka fails nesatur valodas C pirmtekstu.");

outtextxy(170,300,"Fails ir vieksmigi atverts.");

if (isC() == 1) {

setcolor(1); outtextxy(170,300,"Fais

ir veiksmigi atverts.");

}

else {

setcolor(6);

outtextxy(170,300,"Fails ir veiksmigi atverts. Iespejams, ka fails nesatur valodas C pirmtekstu.");

}

}

else{

setcolor(RED);

outtextxy(170,300,"Neizdevas

atvert failu, parliecinieties, ka tas eksiste.");

}

}

else{

if(strcmp(szLastPath,szResultPath)==0)

Exist = 1;

else Exist = 0;

if( ( Exist == 0 &&

ResultFileExists(argv[0],szResultPath) ) )

{

sprintf(szLastPath,"%s",szResultPath);

Exist = 1;

setcolor(3);

outtextxy(170,320,"Neizdevas

izveidot failu, parliecinieties, ka nosaukums ir pareizs.");

outtextxy(170,320,"Fails ir

veiksmigi atverts.");

setcolor(RED);

outtextxy(170,320,"Fails jau

eksiste, Enter - parrakstit failu.");

}else{

if( OpenResultFile (argv[0], szResultPath) ){

input2 = 1;

current = 0;

Exist = 0;

drawTextBox(170, 265,

220, szResultPath);

setcolor(3);

outtextxy(170,320,"Neizdevas izveidot failu, parliecinieties, ka nosaukums ir pareizs.");

outtextxy(170,320,"Fails jau eksiste, Enter - parrakstit failu.");

setcolor(1); outtextxy(170,320,"Fails ir

veiksmigi atverts.");

}

else{

setcolor(3);

outtextxy(170,320,"Fails jau eksiste, Enter - parrakstit failu.");

outtextxy(170,320,"Fails ir veiksmigi atverts.");

setcolor(RED);

outtextxy(170,320,"Neizdevas izveidot failu, parliecinieties, ka nosaukums ir pareizs.");

}

}

}

}

}

break;

case 1:

drawPicture(10,15,128,162);

drawTitle();

current = 0;

szDescriptor.JointlyRows = 0;

szDescriptor.JointlyBytes = 0;

szDescriptor.CreatedComments = 0;

szDescriptor.iSingleOffset = 0;

szDescriptor.iCurrentOffset = 0;

szDescriptor.CreatedLines = 0;

szDescriptor.iSetOffset = 0;

szDescriptor.iComment = 0;

while( (current = ReadString(szSourceString)) >= -1 ){

szDescriptor.JointlyRows++;

szDescriptor.JointlyBytes+=current;

if( current == -1 )

{

szDescriptor.JointlyBytes +=

strlen(szSourceString)+1;

break;

}

}

current = 0;

input = 0;

fseek(hSourceFile,0,SEEK\_SET);

while( (current = ReadString(szSourceString)) >= -1 ){

AnalizeText(szSourceString,&szDescriptor);

if( current == -1 ) break;

}

setcolor(BLUE);

outtextxy(30,195,"Faila apstrade ir pabeigta.");

settextstyle(SMALL\_FONT, HORIZ\_DIR, 4);

outtextxy(30,220,"Apstrades gaita par failu tika sakopota

sekojosa informacija:");

setcolor(DARKGRAY);

sprintf(szSourceString,"Faila pirmteksts satureja %i

rindas.",szDescriptor.JointlyRows);

outtextxy(40,250,szSourceString);

sprintf(szSourceString,"Faila pirmteksts satureja %d

simbolus.",szDescriptor.JointlyBytes);

outtextxy(40,270,szSourceString);

if( szDescriptor.CreatedComments != 0 )

sprintf(szSourceString,"Tika pievienoti %i komentari.",szDescriptor.CreatedComments);

else

sprintf(szSourceString,"Jauni komentari netika pievienoti.");

outtextxy(40,290,szSourceString);

if(szDescriptor.CreatedLines != szDescriptor.JointlyRows){

sprintf(szSourceString,"Programma tika %s par %i

rindam.",(LongMin(szDescriptor.CreatedLines,szDescriptor.JointlyRows)==szDescriptor.CreatedLines)?"samazinata":"palielinata",

LongMax(szDescriptor.CreatedLines,szDescriptor.JointlyRows) - LongMin(szDescriptor.CreatedLines,szDescriptor.JointlyRows));

outtextxy(40,310,szSourceString);

}else outtextxy(40,310,"Programmas rindu skaits netika

mainits.");

setcolor(BLUE);

outtextxy(30,400,"<ENTER> - Apskatit rezultejosa faila

saturu.");

setcolor(RED);

outtextxy(30,420,"<ESC> - Iziet no programmas.");

ch = 0x00;

while (ch != VK\_ESCAPE && ch != VK\_RETURN)

{

ch = getch();

if (ch != VK\_ESCAPE && ch != VK\_RETURN)

{

setcolor(RED);

outtextxy(110,360,"Tika ievadita nekorekta

informacija, ludzu, meginiet velreiz");

}

}

if (ch == VK\_ESCAPE) step = 3;

break;

case 2:

setcolor(BLUE);

settextstyle(SMALL\_FONT,HORIZ\_DIR,6);

outtextxy(230,5,"Faila saturs:");

CloseSourceFile();

hSourceFile = hResultFile;

input = 0;

input1 = 1;

input2 = 0;

current = 0;

dbProgress = 0;

ch = 0x00;

settextstyle(SMALL\_FONT,HORIZ\_DIR,4);

setcolor(DARKGRAY);

while(input1)

{

fseek(hSourceFile,0,SEEK\_SET);

if( szDescriptor.CreatedLines >= 41 )

dbProgress = (input2/(double) (szDescriptor.CreatedLines-41))\*100;

drawListBox(5,30,630,440,(int)dbProgress);

input = 0;

while( (current = ReadString(szSourceString)) >= -1){

if( input >= input2 ){

current = 0;

while(textwidth(szSourceString) > 600)

szSourceString[strlen(szSourceString)-1]='\0';

outtextxy(10,35+((input-

input2)\*10),szSourceString);

}

input++;

if( current == -1 || (input - input2) > 41 )

break;

}

ch = getch();

if( ch == 0x00 ) ch = getch();

if( ch == VK\_UP && input2 != 0 ) input2--;

if( ch == VK\_DOWN && input2 <

(szDescriptor.CreatedLines - 41) ) input2++;

input1 = (ch != VK\_ESCAPE);

}

CloseResultFile();

break;

}

}

closegraph();

}

else{

printf ("Graphic initialization error: %s \n", errorcode);

printf ("Press any key to halt :");

getch();

exit(1);

}

}

**Fails shape.h**

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

#include <string.h>

#include <graphics.h>

#define VK\_TAB 0x09

#define VK\_CLEAR 0x08

#define VK\_RETURN 0x0D

#define VK\_ESCAPE 0x1B

#define VK\_LEFT 0x4b

#define VK\_RIGHT 0x4d

#define VK\_UP 0x48

#define VK\_DOWN 0x50

#define VK\_DELETE 0x53

#define MAXPATH 256

int CharInsert(char \*iString, char chChar, int Pos);

int drawTextBox(int x, int y, int cx, char \*iText);

int GetCursorPos(char \*iText, int cx, int Cursor);

int ReadInputString(int x,int y, int cx,char \*iText);

int initGraphics();

void drawTitle();

void setBackground(int i);

void CharDelete(char \*iString, int Pos);

void drawListBox(int x, int y, int cx, int cy, int iOX);

void drawPicture(int x, int y, int cx, int cy);

int initGraphic()

{

int graphicDriver = VGA, graphicMode = VGAHI;

initgraph ( &graphicDriver, &graphicMode, "C:\\BORLANDC\\BGI" );

return graphresult();

}

void setBackground(int i)

{

setbkcolor(i);

cleardevice();

}

void drawTitle()

{

setcolor(BLUE);

settextstyle(1, HORIZ\_DIR, 1);

outtextxy(190,100,"PROGRAMMA TEKSTA IZVADEI");

settextstyle(SMALL\_FONT, HORIZ\_DIR, 5);

outtextxy(190,130,"ATBILSTOSI PROGRAMMESANAS STILA PRASIBAM.");

}

void drawPicture(int x, int y, int cx, int cy)

{

char szText[2];

int i,z,iX,iY,jX,jY,iOffset,iChX=0,iChY=10, maxx;

int iColors[]={WHITE,DARKGRAY,LIGHTGRAY,LIGHTBLUE};

int Triangle[6];

for(i = 0; i < 4; i++)

{

setcolor(iColors[i]);

line(x+(cx+i)-111,(y+i)+9,x+(cx-i)+17,(y+i)+9);

line(x+(cx-i)+17,(y+i)+9,x+(cx-i)+17,(y-i)+171);

line(x+(cx-i)+17,(y-i)+171,x+(cx+i)-111,(y-i)+171);

line(x+(cx+i)-111,(y-i)+171,x+(cx+i)-111,(y+i)+9);

if( i == 0 ){

setfillstyle(SOLID\_FILL,WHITE);

floodfill(x+(cx+4),y+(cy+4),iColors[i]);

}

}

settextstyle(SMALL\_FONT,HORIZ\_DIR,4);

for(jY = y+20;jY < y+(cy-10); jY += iChY){

for(jX = x+27; jX < x+130; jX += iChX){

setcolor(iColors[rand()%4]);

sprintf(szText,"%c",'0'+rand()%('z'-'0'));

iChX = textwidth(szText);

outtextxy(jX,jY,szText);

}

}

setcolor(WHITE);

rectangle(cx-9,y-4,cx+3,cy-55);

setcolor(DARKGRAY);

rectangle(cx-8,y-3,cx+2,cy-56);

setcolor(LIGHTGRAY);

rectangle(cx-7,y-2,cx+1,cy-57);

setcolor(LIGHTBLUE);

rectangle(cx-6,y-1,cx,cy-58);

setfillstyle(INTERLEAVE\_FILL,WHITE);

bar(cx-5,y,cx-1,cy-59);

setfillstyle(SOLID\_FILL,WHITE);

bar(cx-9,cy-46,cx+3,cy-55);

setcolor(DARKGRAY);

rectangle(cx-8,cy-47,cx+2,cy-55);

setcolor(LIGHTGRAY);

rectangle(cx-7,cy-48,cx+1,cy-54);

setcolor(DARKGRAY);

Triangle[0] = cx-8;

Triangle[1] = cy-47;

Triangle[2] = cx-3;

Triangle[3] = cy-42;

Triangle[4] = cx+2;

Triangle[5] = cy-47;

setfillstyle(SOLID\_FILL,DARKGRAY);

fillpoly(3,Triangle);

}

int drawTextBox(int x, int y, int cx, char \*iText)

{

int cy = 20;

int Offset = 0;

setcolor(BLUE);

rectangle(x,y,x+cx,y+cy);

setcolor(LIGHTBLUE);

rectangle(x+1,y+1,x+cx-1,y+cy-1);

setfillstyle(SOLID\_FILL,WHITE);

bar(x+2,y+2,x+cx-2,y+cy-2);

setcolor(BLUE);

settextstyle(SMALL\_FONT,HORIZ\_DIR,4);

while( textwidth(iText+Offset) > cx-10 ) Offset++;

outtextxy(x+5,y+5,iText+Offset);

return textwidth(iText+Offset);

}

int ReadInputString(int x,int y, int cx,char \*iText){

int Continue = 1, Cursor;

int CursorPos = strlen(iText);

char chNewChar = 0x00;

Cursor = GetCursorPos(iText,cx,CursorPos);

setcolor(BLUE);

line(x+3+Cursor,y+17,x+9+Cursor,y+17);

while( Continue == 1 ){

chNewChar = getch();

if( chNewChar == 0x00 ){

chNewChar = getch();

if( chNewChar == VK\_LEFT && CursorPos != 0 ) CursorPos--;

if( chNewChar == VK\_RIGHT && CursorPos != strlen(iText) )

CursorPos++;

if( chNewChar == VK\_DELETE && strlen(iText) != 0 &&

CursorPos != strlen(iText) ) CharDelete(iText,CursorPos);

}else{

if( chNewChar == VK\_ESCAPE ) return -1;

if( chNewChar == VK\_TAB ) return 0;

if( chNewChar == VK\_RETURN ) return 1;

if( chNewChar == VK\_CLEAR && CursorPos != 0 ){

CursorPos--;

CharDelete(iText,CursorPos);

}else{

CursorPos += CharInsert(iText,chNewChar,CursorPos);

}

}

drawTextBox(x, y, cx, iText);

Cursor = GetCursorPos(iText,cx,CursorPos);

setcolor(BLUE);

line(x+3+Cursor,y+17,x+9+Cursor,y+17);

}

return 0;

}

int GetCursorPos(char \*iText, int cx, int Cursor)

{

int Offset = 0, iReturn;

char chTemp;

chTemp = iText[Cursor];

iText[Cursor] = '\0';

while( textwidth(iText+Offset) > cx-10 ) Offset++;

iReturn = textwidth(iText+Offset);

iText[Cursor] = chTemp;

return iReturn;

}

void CharDelete(char \*iString, int Pos)

{

if( strlen(iString) == Pos + 1 ){

iString[Pos] = '\0';

}

else{

int i;

for(i = Pos; i < strlen(iString); i++)

{

iString[i] = iString[i + 1];

}

iString[strlen(iString)] = '\0';

}

}

int CharInsert(char \*iString, char chChar, int Pos)

{

if( strlen(iString) != MAXPATH ){

if( strlen(iString) == Pos ){

iString[Pos] = chChar;

}

else{

int i;

for(i = strlen(iString); i > Pos ; i--){

iString[i] = iString[ i - 1 ];

}

iString[Pos] = chChar;

}

return 1;

}

return 0;

}

void drawListBox(int x, int y, int cx, int cy, int iOX)

{

int Triangle[6];

double dbLength = 0.0;

setcolor(DARKGRAY);

rectangle(x,y,x+cx,y+cy);

setcolor(3);

rectangle(x+1,y+1,x+cx-1,y+cy-1);

setfillstyle(SOLID\_FILL,WHITE);

bar(x+2,y+2,x+cx-22,y+cy-2);

setcolor(3);

rectangle(x+cx-20,y+1,x+cx-1,y+cy-1);

setcolor(DARKGRAY);

line(x+cx-21,y,x+cx-21,y+cy);

setfillstyle(INTERLEAVE\_FILL,BROWN);

bar(x+cx-19,y+2,x+cx-2,y+cy-2);

setcolor(DARKGRAY);

line(x+cx-21,y+20,x+cx,y+20);

line(x+cx-21,y+cy-20,x+cx,y+cy-20);

setcolor(3);

rectangle(x+cx-19,y+2,x+cx-2,y+19);

rectangle(x+cx-19,y+cy-2,x+cx-2,y+cy-19);

setfillstyle(INTERLEAVE\_FILL,WHITE);

bar(x+cx-18,y+3,x+cx-3,y+18);

bar(x+cx-18,y+cy-3,x+cx-3,y+cy-18);

setfillstyle(SOLID\_FILL,DARKGRAY);

Triangle[0] = x+cx-16;

Triangle[1] = y+15;

Triangle[2] = x+cx-4;

Triangle[3] = y+15;

Triangle[4] = x+cx-10;

Triangle[5] = y+5;

fillpoly(3,Triangle);

Triangle[0] = x+cx-16;

Triangle[1] = y+cy-15;

Triangle[2] = x+cx-4;

Triangle[3] = y+cy-15;

Triangle[4] = x+cx-10;

Triangle[5] = y+cy-5;

fillpoly(3,Triangle);

dbLength = (iOX/100.0)\*(cy-70);

setcolor(DARKGRAY);

rectangle(x+cx-19,y+20+(int)dbLength,x+cx-2,y+50+(int)dbLength);

setfillstyle(INTERLEAVE\_FILL,WHITE);

bar(x+cx-18,y+21+(int)dbLength,x+cx-3,y+49+(int)dbLength);

}

**Fails file.h**

#include <stdlib.h>

#include <string.h>

#include <stdio.h>

#include <conio.h>

#include <errno.h>

#define MAXPATH 256

#define MAXSTR 512

FILE \*hSourceFile = NULL;

FILE \*hResultFile = NULL;

int isC();

int CloseResultFile();

int CloseResultFile();

void StripSpaces(char \*szString);

int ReadString(char \*inString);

int InText( char \*szString, int iPos );

int IsEnding( char \*szString, int iPos );

int InBrackets( char \*szString, int iPos );

int FindChar(char \*szString, char chChar);

int ReverseFindChar(char \*szString, char chChar);

int FindWord(char \*szString, char \*szWord);

int IsComment( char \*szString, int iComment );

int FindSubString(char \*szText, char \*szSubString);

int OpenSourceFile(char \*inPathName, char \*inFileName);

int OpenResultFile(char \*inPathName, char \*inFileName);

int ResultFileExists(char \*szPathName, char \*szFileName);

int AnalizeText(char \*szString,struct SourceFileDescription \*szDescriptor);

typedef struct SourceFileDescription{

long JointlyRows;

long CreatedLines;

long JointlyBytes;

char chLastSeparator;

int iSingleOffset;

int iCurrentOffset;

int CreatedComments;

int iSetOffset;

int iComment;

};

#define min(a,b) a>b?b:a

int IsFunction( char \*szString, struct SourceFileDescription \*szDescriptor ){

int i;

if((( i = FindChar(szString,'(')) != -1 ) &&

( InText(szString,i) == 0 ) &&

(szDescriptor->iCurrentOffset == 0)){

return (szDescriptor->chLastSeparator != ';');

}

return 0;

}

int FindDivider( char \*szString, struct SourceFileDescription \*szDescriptor ){

int a,b,c,x;

a = FindChar(szString,';');

b = FindChar(szString,'{');

c = FindChar(szString,'}');

if (InBrackets(szString,a) != 0)

return -1;

if( a == -1 ) a = strlen(szString) + 1;

if( b == -1 ) b = strlen(szString) + 1;

if( c == -1 ) c = strlen(szString) + 1;

x = min(min(a,b),c);

if( x == a ) szDescriptor->chLastSeparator = ';';

if( x == b ) szDescriptor->chLastSeparator = '{';

if( x == c ) szDescriptor->chLastSeparator = '}';

if( x == (strlen(szString) + 1)) return -1;

return x;

}

int InBrackets( char \*szString, int iPos ){

int InBrackets = 0, i;

for( i = 0; i < iPos; i++ ){

if( ( szString[i] == '(' ) && (InText(szString,i) == 0) ){

InBrackets = 1;

}

if( ( szString[i + 1] == ')' ) && (InText(szString,i) == 0) ){

InBrackets = 0;

}

}

if (szString[iPos - 1] == ')')

InBrackets = 0;

return InBrackets;

}

int IsEnding( char \*szString, int iPos ){

int i;

for(i = iPos; i < strlen(szString); i++ )

if( szString[i] != ' ' && szString[i] != '\t' && szString[i] !=

'\r' ) return 0;

return 1;

}

int InComment( char \*szString, int iComment)

{

int i;

if( iComment == 1 )

iComment = 0;

for(i = 0; i < strlen(szString); i++ ){

if( iComment == 0 )

{

if( (szString[i] == '/' && szString[i+1] == '\*') ||

(szString[i] == '/' && szString[i+1] == '/'))

{

iComment = 1;

}

else if( szString[i] != ' ' && szString[i] != '\t' &&

szString[i] != '\r' ) return 0;

}

}

return iComment;

}

int AnalizeText(char \*szString, struct SourceFileDescription \*szDescriptor)

{

int iDivider = -1, iSubDivider = -1, iOffset=0, i,x,iPrint = 1;

char chCharDiv, chTemp;

StripSpaces(szString);

if( strlen( szString ) == 0 ) return 0;

while( (iDivider = FindDivider(szString + iOffset, szDescriptor)) != -1)

{

iOffset+=iDivider;

if( InText(szString,iOffset) == 0 ) break;

else iDivider = -1;

iOffset++;

}

if( iDivider != -1 ){

iSubDivider = iOffset+1;

iDivider = iOffset;

while((iDivider<(strlen(szString)-1))&&(szString[iDivider+1]

== ' '||szString[iDivider+1]=='\t' ||szString[iDivider+1]=='\r'))

iDivider++;

if( iDivider>=(strlen(szString)-1) ) iDivider = -1;

else iDivider++;

}

if( iDivider != -1 ){

chCharDiv = szString[iSubDivider];

szString[iSubDivider] = '\0';

}

if(((i = FindChar(szString,'}'))!= -1 ) && ( InText(szString,i) == 0 )){

szDescriptor->iCurrentOffset--;

}

while((( i = FindWord(szString,"#include"))!= -1) && (InText(szString,i)

== 0 ) && (( x = FindChar(szString + i,'>')) != -1 ) &&

(IsEnding(szString,x+1) == 0) ){

chTemp = szString[i+x+1];

szString[i+x+1] = '\0';

for( i = 0; i < szDescriptor->iCurrentOffset; i++ )

fputs(" ",hResultFile);

fprintf(hResultFile,"%s\n", szString);

szString[i+x+1] = chTemp;

szString += i+x+1;

iDivider -= i+x+1;

iSubDivider -= i+x+1;

if( chTemp == ' ' || chTemp == '\t' ){

szString++;

iDivider--;

iSubDivider--;

}

szDescriptor->CreatedLines++;

}

if(( szString[0] == '}' ) && ( IsEnding( szString, 1 ) == 0 )){

for( i = 0; i < szDescriptor->iCurrentOffset; i++ )

fputs(" ",hResultFile);

fprintf(hResultFile,"}\n");

if( szString[1] == ' ' || szString[1] == '\t' ){

szString += 2;

iDivider -= 2;

iSubDivider -= 2;

}else{

szString++;

iDivider--;

iSubDivider--;

}

}

if(IsFunction( szString, szDescriptor ) && szDescriptor->iComment == 0){

fprintf(hResultFile,"\n");

fprintf(hResultFile,"/\*-------------------------------\n");

fprintf(hResultFile," Funkcijas apraksts \n");

fprintf(hResultFile,"-------------------------------\*/\n");

szDescriptor->CreatedLines+=4;

szDescriptor->CreatedComments++;

}

if ((i = FindWord(szString,"case")) != -1 && (x =

FindChar(szString+i+4,':')) != -1 )

{

if (szDescriptor->iComment == 0)

{

fputs("\n",hResultFile);

for( i = 0; i < szDescriptor->iCurrentOffset; i++ )

fputs(" ",hResultFile);

fputs("// Nosacijuma komentars\n",hResultFile);

szDescriptor->CreatedLines += 2;

szDescriptor->CreatedComments++;

szDescriptor->iSetOffset = 1;

}

}

if ( FindWord(szString,"case") == -1 && szDescriptor->iSetOffset == 1)

{

szDescriptor->iCurrentOffset++;

szDescriptor->iSetOffset = 0;

}

if ( FindWord(szString,"break") != -1 )

{

szDescriptor->iCurrentOffset--;

}

if( (i = FindWord(szString,"for")) != -1){

if (szDescriptor->iComment == 0)

{

fputs("\n",hResultFile);

for( i = 0; i < szDescriptor->iCurrentOffset; i++ )

fputs(" ",hResultFile);

fputs("// Cikla komentars\n",hResultFile);

szDescriptor->CreatedLines += 2;

szDescriptor->CreatedComments++;

}

if( szDescriptor->chLastSeparator != '{' && (( x =

FindChar(szString,')')) != -1 )){

if( IsEnding(szString, x+1) != 0 ){

szDescriptor->iSingleOffset = 1;

}

}

if( szDescriptor->chLastSeparator == ';' && (( x =

FindChar(szString,')')) != -1 ))

{

if((IsEnding(szString, x+1) == 0) && (szString[x+1] != '{'))

{

iPrint = 0;

chTemp = szString[x + 1];

szString[x + 1] = '\0';

for( i = 0; i < szDescriptor->iCurrentOffset; i++ )

fputs(" ",hResultFile);

fprintf(hResultFile,"%s\n", szString);

szString[x + 1] = chTemp;

for( i = 0; i <= szDescriptor->iCurrentOffset; i++ )

fputs(" ",hResultFile);

fprintf(hResultFile,"%s\n", szString + (x+1));

}

}

}

if ( ((i = FindWord(szString,"while")) != -1)){

if (szDescriptor->iComment == 0)

{

fputs("\n",hResultFile);

for( i = 0; i < szDescriptor->iCurrentOffset; i++ )

fputs(" ",hResultFile);

fputs("// Cikla komentars\n",hResultFile);

szDescriptor->CreatedLines += 2;

szDescriptor->CreatedComments++;

}

if( szDescriptor->chLastSeparator != '{' && (( x =

ReverseFindChar(szString,')')) != -1 )){

if( IsEnding(szString, x+1) != 0 ){

szDescriptor->iSingleOffset = 1;

}else{

iPrint = 0;

chTemp = szString[x + 1];

szString[x + 1] = '\0';

for( i = 0; i < szDescriptor->iCurrentOffset; i++ )

fputs(" ",hResultFile);

fprintf(hResultFile,"%s\n", szString);

szString[x + 1] = chTemp;

for( i = 0; i <= szDescriptor->iCurrentOffset; i++ )

fputs(" ",hResultFile);

fprintf(hResultFile,"%s\n", szString + (x+1));

}

}

}

if( (i = FindWord(szString,"if")) != -1 )

{

if (szDescriptor->iComment == 0)

{

fputs("\n",hResultFile);

for( i = 0; i < szDescriptor->iCurrentOffset; i++ )

fputs(" ",hResultFile);

fputs("// Nosacijuma komentars\n",hResultFile);

szDescriptor->CreatedLines += 2;

szDescriptor->CreatedComments++;

}

if( szDescriptor->chLastSeparator != '{' && (( x =

ReverseFindChar(szString,')')) != -1 )){

if( IsEnding(szString, x+1) != 0 ){

szDescriptor->iSingleOffset = 1;

}else{

iPrint = 0;

chTemp = szString[x + 1];

szString[x + 1] = '\0';

for( i = 0; i < szDescriptor->iCurrentOffset; i++ )

fputs(" ",hResultFile);

fprintf(hResultFile,"%s\n", szString);

szString[x + 1] = chTemp;

for( i = 0; i <= szDescriptor->iCurrentOffset; i++ )

fputs(" ",hResultFile);

fprintf(hResultFile,"%s\n", szString + (x+1));

}

}

}

if((i = FindWord(szString,"else")) != -1 && (IsEnding(szString,i+5)!=0 )

&& (szDescriptor->chLastSeparator != '{'))

{

szDescriptor->iSingleOffset = 1;

}

if((x = FindWord(szString,"else")) != -1 && (IsEnding(szString,x+5)== 0)

&& (szDescriptor->chLastSeparator != '{'))

{

iPrint = 0;

chTemp = szString[x + 5];

szString[x + 5] = '\0';

for( i = 0; i < szDescriptor->iCurrentOffset; i++ )

fputs(" ",hResultFile);

fprintf(hResultFile,"%s\n", szString);

szString[x + 5] = chTemp;

for( i = 0; i <= szDescriptor->iCurrentOffset; i++ )

fputs(" ",hResultFile);

fprintf(hResultFile,"%s\n", szString + (x+5));

}

if(( szString[0] == '{' ) && ( IsEnding( szString, 1 ) != 0 ))

{

szDescriptor->iSingleOffset = 0;

}

if( iPrint == 1 ){

if( szDescriptor->iSingleOffset == 2 ){

fputs(" ",hResultFile);

szDescriptor->iSingleOffset = 0;

}

if( szDescriptor->iSingleOffset == 1 )

szDescriptor->iSingleOffset = 2;

for( i = 0; i < szDescriptor->iCurrentOffset; i++ )

fputs(" ",hResultFile);

if(((i = FindChar(szString,'{')) != -1) &&

(InText(szString,i) == 0) && (IsEnding( szString,i+1)!= 0)){

if( i != 0 ){

szString[i] = '\0';

fprintf(hResultFile,"%s\n", szString);

for(i = 0; i<szDescriptor->iCurrentOffset; i++ )

fputs(" ",hResultFile);

}

fprintf(hResultFile,"{\n");

szDescriptor->CreatedLines += 2;

szDescriptor->iCurrentOffset++;

}else{

if(((i = FindChar(szString,'}')) != -1) && (InText(szString,i) == 0 ) && ( IsEnding( szString, i+1 ) != 0 )){

if( i != 0 ){

szString[i] = '\0';

fprintf(hResultFile,"%s\n", szString);

for( i = 0;i<szDescriptor->iCurrentOffset; i++ )

fputs(" ",hResultFile);

}

fprintf(hResultFile,"}\n");

szDescriptor->CreatedLines += 2;

}else{

fprintf(hResultFile,"%s\n", szString);

szDescriptor->CreatedLines++;

}

}

}

szDescriptor->iComment = InComment(szString, szDescriptor->iComment);

if( iDivider != -1 ){

szString[iSubDivider] = chCharDiv;

AnalizeText(szString+iDivider,szDescriptor);

}

return 0;

}

int FindWord(char \*szString, char \*szWord){

int iCurrentPos = 0, iLastPos,i,x;

char szTerminatingChars[] = {" ();.\t\r\n"};

while((iLastPos = FindSubString(szString+iCurrentPos,szWord)) != -1){

iCurrentPos+=iLastPos;

if( InText(szString,iCurrentPos) == 0 ){

x = 0;

if( iCurrentPos != NULL ){

for(i=0; i < strlen(szTerminatingChars); i++){

if( szString[iCurrentPos-1] ==

szTerminatingChars[i] ){

x = 1;

break;

}

}

}else x = 1;

if( x == 1 ){

if( iCurrentPos+strlen(szWord) != strlen(szString) ){

for(i=0; i < strlen(szTerminatingChars); i++)

if( szString[strlen(szWord)+iCurrentPos]

== szTerminatingChars[i] )

return iCurrentPos;

}else return iCurrentPos;

}

}

iCurrentPos++;

}

return -1;

}

int FindSubString(char \*szText, char \*szSubString){

int iLettersFound=0, i, a;

char szString[MAXSTR];

strcpy(szString,szText);

if(strlen(szString)<strlen(szSubString)) return -1;

for(i=0; i<strlen(szString); i++){

iLettersFound=0;

if(szString[i]==szSubString[0]){

iLettersFound=1;

for(a=1; a<strlen(szSubString); a++){

if(szString[i+a]==szSubString[a]){

iLettersFound++;

if(iLettersFound==strlen(szSubString)) return i;

}else break;

}

}

}

return -1;

}

int FindChar(char \*szString, char chChar)

{

int i;

for(i=0; i<strlen(szString); i++)

if(szString[i]==chChar) return i;

return -1;

}

int ReverseFindChar(char \*szString, char chChar)

{

int i;

for(i=(strlen(szString)-1); i>=0; i--)

if(szString[i]==chChar) return i;

return -1;

}

int InText( char \*szString, int iPos )

{

int iFound = 0, iOffset = 0, iInText = 0;

if( iPos == 0 ) return 0;

while( (iFound=FindChar(szString + iOffset,'"')) != -1 ){

iOffset+=iFound;

if( iOffset > iPos ) return iInText;

iInText=(iInText==0)?1:0;

iOffset++;

}

return iInText;

}

void StripSpaces(char \*szString)

{

int i=0,iSpaces = 0;

while( ( i < strlen(szString) ) && ((szString[i] == ' ') || (szString[i] == '\t') || (szString[i] == '\r')) ){

iSpaces++;

i++;

}

if( iSpaces > 0 ){

for(i = 0; i < strlen(szString)-iSpaces; i++){

szString[i] = szString[i + iSpaces];

}

szString[i]='\0';

}

}

int isC()

{

char szString[MAXSTR];

int i,iCurrent = 0, iReturn = 0;

fseek(hSourceFile,0,SEEK\_SET);

while( (iCurrent = ReadString(szString)) >= -1 ){

StripSpaces(szString);

if((FindWord(szString,"#include")==0) && (InText(szString,i)==0 ))

{

iReturn = 1;

}

for( i = 0; i < strlen(szString); i++ ){

if(szString[i]!=' ' && szString[i]!='\t' && szString[i]!='\r' && szString[i]!='\n' && iReturn == 0 )

return 0;

}

if( iCurrent == -1 ) break;

}

fseek(hSourceFile,0,SEEK\_SET);

return iReturn;

}

int OpenSourceFile(char \*inPathName, char \*inFileName)

{

char szTempFileName[(2\*MAXPATH)+5] = {NULL};

hSourceFile = fopen(inFileName,"r");

if( hSourceFile == NULL ){

strcat(szTempFileName,inPathName);

strcat(szTempFileName,inFileName);

hSourceFile = fopen(szTempFileName,"r");

if( hSourceFile == NULL ){

strcat(szTempFileName,".c");

hSourceFile = fopen(szTempFileName,"r");

}

}

if(hSourceFile != NULL) fseek(hSourceFile,0,SEEK\_SET);

return (hSourceFile != NULL);

}

int OpenResultFile(char \*inPathName, char \*inFileName)

{

char szTempFileName[(2\*MAXPATH)+5] = {NULL};

hResultFile = fopen(inFileName,"w+");

if( hResultFile == NULL ){

strcat(szTempFileName,inPathName);

strcat(szTempFileName,inFileName);

hResultFile = fopen(szTempFileName,"w+");

if( hResultFile == NULL ){

strcat(szTempFileName,".c");

hResultFile = fopen(szTempFileName,"w+");

}

}

return ( hResultFile != NULL );

}

int ReadString(char \*szString){

int iLength = 0;

memset( szString, 0, MAXSTR );

for(iLength = 0; iLength < MAXSTR; iLength++){

if( fread(szString+iLength,sizeof(char),1,hSourceFile) != NULL ){

if( szString[iLength] == '\n' ) break;

}else{

break;

}

}

if( feof(hSourceFile) == NULL ){

if( szString[iLength] == '\n' ){

szString[iLength] = '\0';

return iLength+1;

}else return -2;

}

szString[iLength] = '\0';

return -1;

}

int ResultFileExists(char \*szPathName, char \*szFileName){

char szTempFileName[(2\*MAXPATH)+5] = {NULL};

FILE \*hTempFile = fopen(szFileName,"r");

if( hTempFile == NULL ){

if( errno == ENOFILE ){

return 0;

}

}else{

fclose(hTempFile);

return 1;

}

strcat(szTempFileName,szPathName);

strcat(szTempFileName,szFileName);

hTempFile = fopen(szTempFileName,"r");

if( hTempFile == NULL ){

if( errno == ENOFILE ){

return 0;

}

}else{

fclose(hTempFile);

return 1;

}

strcat(szTempFileName,".c");

hTempFile = fopen(szTempFileName,"r");

if( hTempFile == NULL ){

return (errno != ENOFILE);

}else{

fclose(hTempFile);

return 1;

}

}

int CloseSourceFile()

{

if( hSourceFile != NULL ){

fclose(hSourceFile);

hSourceFile = NULL;

return 1;

}

return 0;

}

int CloseResultFile()

{

if( hResultFile != NULL ){

fclose(hResultFile);

hResultFile = NULL;

return 1;

}

return 0;

}

## Pielikums

## Programmas izpildes rezultāti

**Valodas C programmas faila const.cpp pirmteksts pirms apstrādes:**

#include <stdio.h> #include <conio.h> int main();void sp\_to\_dash (const char \*str); int main() { int a;sp\_to\_dash ( "text examp\*e" );

if( a == 1 ){ return 1;} getch ();return (0); } void sp\_to\_dash (const char \*str){ while(\*str) { if ( \*str == ' ')

printf ("%c",'-');

else printf ("%c", \*str); str++; } }

**Valodas C programmas faila const.cpp pirmteksts pēc apstrādes:**

#include <stdio.h>

#include <conio.h>

int main();

void sp\_to\_dash (const char \*str);

/\*-------------------------------

Funkcijas apraksts

-------------------------------\*/

int main()

{

int a;

sp\_to\_dash ( "text examp\*e" );

// Nosacijuma komentars

if( a == 1 )

{

return 1;

}

getch ();

return (0);

}

/\*-------------------------------

Funkcijas apraksts

-------------------------------\*/

void sp\_to\_dash (const char \*str)

{

// Cikla komentars

while(\*str)

{

// Nosacijuma komentars

if ( \*str == ' ')

printf ("%c",'-');

else

printf ("%c", \*str);

str++;

}

}

**Valodas C programmas faila array.cpp pirmteksts pirms apstrādes:**

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <conio.h>

# define number 7

void main ()

{

int i, choise;

int array[number];

int newarray[number];

void fun1 ( int array[], int newarray[], int ), fun2 ( int\*, int );

printf(" funkciju izmantosana masivu apstrade'\n");

printf("\n");

printf(" autors: Sergejs Terentjevs, \n grupa:IIDB - 9 \n");

printf("\n");

printf(" 1 / 2 \n");

printf("\n");

scanf("%d", &choise);

switch(choise)

{

case 1:

clrscr();

randomize();

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

array[i] = random (1000);

printf ("Sakummasivs:\n");

for (i=0;i<number;++i) printf(" %d ",array[i]);

break;

case 2:

clrscr();

printf("Ievadiet matricas elementus:\n");

for(i=0; i<number; ++i){

scanf("%d",&array[i]);

printf("\n");}

clrscr();

printf("Sakummasivs:\n");

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

printf(" %d ",array[i]);

break;

}

fun1(array,newarray,number);

fun2(array,number);

printf("\n");

}

void fun1 (int mas[],int newmas[],int n){

register i,x,k,s;

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

{

if (mas[i] > 10)

{

k = mas[i];x = 0;

while (k > 10){

s = k%10;

k = k/10;

x += s;

}newmas[i] = x + k;}

else

newmas[i]=mas[i];}

printf("\n");

printf("\nJauns massivs ar indeksiem:\n");

//Siet ir jau komentars

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

{

printf(" %d ", newmas[i]);

}

}

/\* Siet ir jau komentars \*/

void fun2 (int\* m, int n)

{

register i,x,k,s;

int h[number];

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

{

if ((\*m+i)>10)

{

k = \*(m+i);

x = 0;

while (k > 10)

{s = k%10;

k = k/10;

x += s;

}h[i] = x + k;

}

else { h[i] = \*(m+i);}printf("\n");}

printf("\nJaunais masivs ar raditajiem:\n");

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

printf(" %d ", h[i]);}getch();}

**Valodas C programmas faila array.cpp pirmteksts pēc apstrādes:**

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <conio.h>

# define number 7

/\*-------------------------------

Funkcijas apraksts

-------------------------------\*/

void main ()

{

int i, choise;

int array[number];

int newarray[number];

void fun1 ( int array[], int newarray[], int ), fun2 ( int\*, int );

printf(" funkciju izmantosana masivu apstrade'\n");

printf("\n");

printf(" autors: Sergejs Terentjevs, \n grupa:IIDB - 9 \n");

printf("\n");

printf(" 1 / 2 \n");

printf("\n");

scanf("%d", &choise);

switch(choise)

{

// Nosacijuma komentars

case 1:

clrscr();

randomize();

// Cikla komentars

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

array[i] = random (1000);

printf ("Sakummasivs:\n");

// Cikla komentars

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

printf(" %d ",array[i]);

break;

// Nosacijuma komentars

case 2:

clrscr();

printf("Ievadiet matricas elementus:\n");

// Cikla komentars

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

{

scanf("%d",&array[i]);

printf("\n");

}

clrscr();

printf("Sakummasivs:\n");

// Cikla komentars

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

printf(" %d ",array[i]);

break;

}

fun1(array,newarray,number);

fun2(array,number);

printf("\n");

}

/\*-------------------------------

Funkcijas apraksts

-------------------------------\*/

void fun1 (int mas[],int newmas[],int n)

{

register i,x,k,s;

// Cikla komentars

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

{

// Nosacijuma komentars

if (mas[i] > 10)

{

k = mas[i];

x = 0;

// Cikla komentars

while (k > 10)

{

s = k%10;

k = k/10;

x += s;

}

newmas[i] = x + k;

}

else

newmas[i]=mas[i];

}

printf("\n");

printf("\nJauns massivs ar indeksiem:\n");

//Siet ir jau komentars

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

{

printf(" %d ", newmas[i]);

}

}

/\* Siet ir jau komentars \*/

void fun2 (int\* m, int n)

{

register i,x,k,s;

int h[number];

// Cikla komentars

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

{

// Nosacijuma komentars

if ((\*m+i)>10)

{

k = \*(m+i);

x = 0;

// Cikla komentars

while (k > 10)

{

s = k%10;

k = k/10;

x += s;

}

h[i] = x + k;

}

else

{

h[i] = \*(m+i);

}

printf("\n");

}

printf("\nJaunais masivs ar raditajiem:\n");

// Cikla komentars

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

{

printf(" %d ", h[i]);

}

getch();

}