Національний Університет «Києво - Могилянська Академія»

Департамент комп’ютерних технологій

Кафедра інформатики

ÐÎÇÐÎÁÊÀ ÊÎÌÏ`ÞÒÅÐÍÈÕ ÍÀÂ×ÀËÜÍÈÕ ÇÀÑÎÁ²Â Ç ÎÁÐÀÍÎ¯ ÏÐÅÄÌÅÒÍÎ¯ ÎÁËÀÑÒ²

Курсова робота  
студента II курсу  
Підгорного Свєтослава та студентки II курсу

Фіялко Світлани

Науковий керівник:

кандидат технічних наук

Олецький Олексій Віталійович

Київ. 1997.

Â íàø ÷àñ áàãàòî ñïåö³àë³ñò³â ââàæàþòü, ùî ò³ëüêè êîìï`þòåð çìîæå çä³éñíèòè ÿê³ñíèé ðèâîê â ñèñòåì³ îñâ³òè, ïðîöåñ êîìï`þòåðèçàö³¿ ÿêî¿ â³äîáðàæàº çàãàëüí³ òåíäåíö³¿ àâòîìàòèçàö³¿, ïåðåäà÷³ ð³çíèõ ôóíêö³é ëþäñüêî¿ ä³ÿëüíîñò³ êîìï`þòåðó. Çâè÷àéíî, ÅÎÌ íå ìîæå âèð³øèòè âñ³ ïðîáëåìè îñâ³òè, á³ëüø³ñòü ÿêèõ º íå ò³ëüêè ïåäàãîã³÷íèìè, àëå é ñîö³àëüíèìè. Ðàçîì ç òèì íå ñë³ä íåäîîö³íþâàòè ìîæëèâîñò³ êîìï`þòåðà: ùå í³êîëè ïåäàãîã íå îòðèìóâàâ íàñò³ëüêè ñèëüíîãî çàñîáó íàâ÷àííÿ; æîäåí òåõí³÷íèé çàñ³á, ÿêèé çàñòîñîâóâàâñÿ ðàí³øå, çà ñâî¿ìè äèäàêòè÷íèìè ìîæëèâîñòÿìè íå ìîæå ç íèì çð³âíÿòèñÿ.

Òåîðåòè÷íó îñíîâó ïåðøèõ ñèñòåì êîìï`þòåðíîãî íàâ÷àííÿ ñêëàëè ïðèíöèïè ïðîãðàìîâàíîãî íàâ÷àííÿ (òàê çâàíèõ "ïàïåðîâèõ ìàøèí"). Êîìï`þòåð âèêîðèñòîâóâàâñÿ ïåðåâàæíî ÿê øâèäêîä³þ÷èé îá÷èñëþâà÷.

Ïî ì³ð³ çðîñòàííÿ ìîæëèâîñòåé êîìï`þòåðà éîãî ïî÷àëè âèêîðèñòîâóâàòè äëÿ íàâ÷àííÿ âåëèêî¿ ê³ëüêîñò³ ó÷í³â òà ñòóäåíò³â. Çà éîãî äîïîìîãîþ âäàëîñÿ çä³éñíèòè ðåàë³çàö³þ íîâîãî ñïîñîáó êåðóâàííÿ íàâ÷àëüíîþ ä³ÿëüí³ñòþ. Íîâ³ ïðîãðàìí³ çàñîáè äàëè çìîãó íàëàãîäèòè íàâ÷àííÿ íå ò³ëüêè ç ìàòåìàòèêè, ô³çèêè ³ ò.ä., àëå é ç òàêèõ, çäàâàëîñü, äàëåêèõ â³ä êîìï`þòåðà ïðåäìåò³â, ÿê ìóçèêà, ìàëþâàííÿ. Íàéãîëîâí³øå, ùî äàº êîìï`þòåð, - öå ìîæëèâ³ñòü ä³àëîãîâîãî íàâ÷àííÿ. Ïðè öüîìó íå ò³ëüêè â³í çàäàº ïèòàííÿ, à é äîçâîëÿº öå ðîáèòè êîðèñòóâà÷àì. Óñï³øíî ìîæå áóòè âèêîðèñòàíà â ïðîöåñ³ íàâ÷àííÿ ³ ìîæëèâ³ñòü êîìï`þòåðà ìîäåëþâàòè ð³çí³ ÿâèùà. Ìîâà éäå ïðî çàäà÷³, â ÿêèõ ëþäèíà ñâî¿ìè ä³ÿìè ñàìà ìîæå çì³íèòè ñèòóàö³þ, âîíà íåìîâ º ¿¿ àêòèâíèì ó÷àñíèêîì.

Íàñüîãîäí³ áàãàòî ³ñíóþ÷èõ íàâ÷àëüíèõ ñèñòåì ðåàë³çóº íå ³íäèâ³äóàë³çîâàíå, à ñêîð³øå ³íäèâ³äóàëüíå íàâ÷àííÿ, òîáòî íàâ÷àííÿ çà ôîðìîþ: îäèí, õòî íàâ÷àº - îäèí, õòî íàâ÷àºòüñÿ. ²íêîëè ³íäèâ³äóàë³çîâàíå íàâ÷àííÿ ïîð³âíþºòüñÿ ç àäàïòèâíèì, ÿêå ìàº á³ëüø øèðîêèé ñïåêòð çàñòîñóâàíü, îñê³ëüêè àäàïòàö³ÿ ìîæå áàçóâàòèñÿ íà ³íôîðìàö³¿, ç³áðàíîþ ñèñòåìîþ â ïðîöåñ³ íàâ÷àííÿ ç óðàõóâàííÿì ³ñòîð³¿ íàâ÷àííÿ êîæíîãî ó÷íÿ, çàïðîãðàìîâàíî¿ ðàí³øå. Àäàïòèâíà ñèñòåìà ðåàë³çóº íàâ÷àííÿ çà ðîçãàëóäæåíîþ ïðîãðàìîþ, äå âæå â³äîìî, ÿêà äîïîìîãà áóäå íàäàâàòèñÿ êîðèñòóâà÷ó. Àäàïòèâíå íàâ÷àííÿ ò³ëüêè òîä³ ñòàº ³íäèâ³äóàë³çîâàíèì, êîëè âîíî ñïèðàºòüñÿ íà ìîäåëü ó÷íÿ. Â íàø ÷àñ ç`ÿâëÿþòüñÿ òðè øëÿõè ³íäèâ³äóàë³çàö³¿ íàâ÷àííÿ. Ïåðøèé ç íèõ õàðàêòåðèçóºòüñÿ òèì, ùî âèá³ð íàâ÷àëüíèõ çàñîá³â ïîâí³ñòþ âèçíà÷àºòüñÿ êîìï`þòåðîì. Äðóãèé ïåðåäáà÷àº, ùî ò³, õòî íàâ÷àºòüñÿ, ñàì³ é êåðóþòü ïðîöåñîì íàâ÷àííÿ. Òðåò³é øëÿõ ðåàë³çóº çì³øàíå êåðóâàííÿ.

Òàêèì ÷èíîì, ñòâîðåííÿ ïðîãðàìíèõ ïåäàãîã³÷íèõ çàñîá³â - öå ñêëàäíèé òâîð÷èé ïðîöåñ, ÿêèé ³ñòîòíî â³äð³çíÿºòüñÿ â³ä ïåäàãîã³÷íî¿ ä³ÿëüíîñò³, ÿê ïðàêòè÷íî¿, òàê ³ íàóêîâî¿.

Ìîæíà âèä³ëèòè äâà îñíîâí³ íàïðÿìêè ñòâîðåííÿ ïåäàãîã³÷íèõ ïðîãðàìíèõ çàñîá³â. Ïåðøèé íàïðÿìîê ðîçðîáëÿºòüñÿ íàéá³ëüøîþ ê³ëüê³ñòþ àâòîð³â, ùî ïðàöþþòü äëÿ êîìåðö³éíèõ êîìïàí³é, ÿê³ âèïóñêàþòü âæå çàâåðøåí³ íàâ÷àëüí³ ïðîãðàìè. Âèêîðèñòàííÿ öèõ ïðîãðàì îð³ºíòîâàíî íà çàêð³ïëåííÿ òèõ íàâè÷îê, ÿê³ âæå ïåâíîþ ì³ðîþ áóëè ñôîðìîâàí³. ßê ïðàâèëî, öå ïîâ`ÿçàíî ç ìåõàí³÷íèì ïîâòîðåííÿì âæå âèâ÷åíîãî ìàòåð³àëó. Â íàø ÷àñ ³ñíóº áàãàòî ïðîãðàì òàêîãî òèïó äëÿ íàâ÷àííÿ, íàïðèêëàä, àíãë³éñüê³é ìîâ³, ìàòåìàòèö³, á³îëîã³¿. Íàïðèêëàä, ³ñíóº òàêèé ïàêåò, ÿê "Þí³îð-1", äëÿ IBM PC - ñóì³ñíèõ êîìï`þòåð³â, ÿêèé ñòâîðåíèé äëÿ ³íäèâ³äóàëüíîãî çíàéîìñòâà ïî÷àòê³âöÿ ç êîìï`þòåðîì. Öåé ïàêåò ïðèçíà÷åíèé íå ñò³ëüêè äëÿ íàâ÷àííÿ, ñê³ëüêè äëÿ ñàìîñò³éíîãî îïàíóâàííÿ ä³ÿìè, íåîáõ³äíèìè êîðèñòóâà÷ó ÅÎÌ.

Âåëèêó ö³êàâ³ñòü ìàþòü íàâ÷àëüí³ ïðîãðàìè, ùî âèêîðèñòîâóþòü îáºêòíî-îð³ºíòîâíèé ï³äõ³ä, îñíîâó ÿêîãî ñêëàäàº íàñòóïíà ñèñòåìà âèìîã:

* êîðèñòóâà÷ ïîâèíåí ïðàöþâàòè ç ðåàëüíèìè îá`ºêòàìè ïðåäìåòíî¿ îáëàñò³ (ð³âíÿííÿìè, ìàòðèöÿìè òîùî), à íå ç òåêñòàìè ³ çàïèòàííÿìè.
* êîðèñòóâà÷ ïîâèíåí ïðàöþâàòè â ðåàëüíîìó îïåðàö³éíîìó ñåðåäîâèù³, ÿêå îäíîçíà÷íî âèçíà÷àºòüñÿ ïðåäìåòíîþ îáëàñòþ (íàïðèêëàä, äëÿ ìàòðèöü: ñêëàñòè äâà ðÿäêè, ïîìíîæèòè ðÿäîê íà ÷èñëî).
* ³íòåðôåéñ êîðèñòóâà÷à íå ïîâèíåí ñóòòºâî â³äð³çíÿòèñÿ â³ä òðàäèö³éíîãî (àðêóø ïàïåðó ïåðåòâîðþºòüñÿ íà åêðàí)
* êîðèñòóâà÷ ïîâèíåí ìàòè ñâîáîäó, îáìåæåíó ò³ëüêè ðàìêàìè ïðåäìåòíî¿ îáëàñò³, òîáòî êîðèñòóâà÷ íå ïîâèíåí çíàõîäèòèñü ï³ä ïðåñîì àëãîðèòìó ðîçâ`ÿçêó, âèçíà÷åíîãî íà ñòàä³¿ íàïèñàííÿ ïðîãðàìè.

Ïðèêëàäîì ïðîãðàìíîãî çàñîáó, â îñíîâ³ ÿêîãî ëåæèòü öåé ï³äõ³ä, ìîæå ñëóãóâàòè ïðîãðàìà ÔÎÐÌÓËÀ, ÿêà ðîçðîáëåíà ÿê ìàòåìàòè÷íå ñåðåäîâèùº äëÿ øêîëÿð³â òà ñòóäåíò³â.

Øèðîêå ðîçïîâñÿäæåííÿ êîìï`þòåðíèõ òåõíîëîã³é ïðèâåëî äî ïîÿâè âåëèêî¿ ê³ëüêîñò³ ïðîôåñ³éíèõ ïàêåò³â (MathCAD, MathLab). Íàìàãàííÿ âêëþ÷èòè òàê³ çàñîáè â íàâ÷àëüíèé ïðîöåñ ïðèçâåëè äî íåîáõ³äíîñò³ ðîçðîáëÿòè ñïåö³àëüí³ íàâ÷àëüí³ êóðñè äëÿ îïàíóâàííÿ òèì ÷è ³íøèì ïàêåòîì.

²íøèé íàïðÿìîê ðîçðîáêè êîìï`þòåðíèõ ïðîãðàì ïîâ`ÿçàíèé ç âèêîðèñòàííÿì ÅÎÌ â ñïåö³àëüíî ñòâîðåíîìó ñåðåäîâèù³. Ãîëîâíèì àñïåêòîì íàâ÷àííÿ òóò º íå âèâ÷åííÿ ïðàâèë, à ñàì ïðîöåñ ìèñëåííÿ. Çíà÷íèé âêëàä â ðîçâèòîê öüîãî íàïðÿìêó âíåñëè òåõí³÷í³ ïåäàãîã³÷í³ äîñë³äíèöüê³ öåíòðè - TERC. Â íèõ áóëè ðîçðîáëåí³ ³ øèðîêî âèêîðèñòîâóþòüñÿ ïðîãðàìí³ ñåðåäîâèùà: ñèñòåìà LOGO ³ ¿¿ ïîäàëüøèé ðîçâèòîê LOGOWriter; ëàáîðàòîð³¿, çàñíîâàí³ íà ì³êðîêîìï`þòåðàõ, äå êîðèñòóâà÷³ ñàì³ ñòàþòü äîñë³äíèêàìè, ÿê³ â³äêðèâàþòü çàêîíè ìàòåìàòèêè, á³îëîã³¿, ô³çèêè.

Îñîáëèâó ö³êàâ³ñòü ìàº ³íñòðóìåíòàëüíà ñèñòåìà IBM LinkWay. Ñèñòåìà LinkWay º ã³ïåðòåêñòîâîþ ñèñòåìîþ. Ðîáîòó ç ã³ïåðòåêñòîâîþ ñèñòåìîþ ìîæíà ïîð³âíÿòè ç ÷èòàííÿì åíöèêëîïåä³¿. Íà â³äì³íó â³ä ñòàòò³, ÿêó, çâè÷àéíî, ÷èòàþòü ïîñë³äîâíî, ç åíöèêëîïåä³ºþ ïðàöþþòü ïî ³íøîìó: çóñòð³âøè â òåêñò³ âêàç³âíèê íà ÿêåñü ïîíÿòòÿ, ÷èòà÷ àáî ïåðåõîäèòü íà ñòîð³íêó, äå ïîÿñíþºòüñÿ öå ïîíÿòòÿ, àáî ïðîäîâæóº ÷èòàòè äàë³.

Ñèñòåìà LinkWay ìîæå áóòè çàñòîñîâàíÿ äëÿ:

* ðîçðîáêè äåìîíñòðàö³éíèõ "ðîëèê³â" ç ð³çíèõ òåì.
* ïîáóäîâè çàíÿòü â ã³ïåðòåêñòîâîìó ñåðåäîâèù³.
* îðãàí³çàö³¿ ïåðñîíàëüíèõ áàç äàííèõ.
* êåðóâàííÿ çîâí³øí³ìè ïðèñòðîÿìè.
* ïîáóäîâè îáîëîíêè îïåðàö³éíî¿ ñèñòåìè ÷è ïàêåò³â ïðèêëàäíèõ ïðîãðàì.

Â ïðîöåñ³ ðîçâèòêó ïðîãðàìíîãî çàáåçïå÷åííÿ äëÿ íàâ÷àííÿ ñôîðìóâàâñÿ ³ íàïðÿìîê, îð³ºíòîâàíèé íà âèêîðèñòàííÿ åêñïåðòíèõ ñèñòåì ïðè íàâ÷àíí³ ð³çíèì ïðåäìåòàì. Åêñïåðòí³ ñèñòåìè º, ïî ñóò³, ìîäåëлþ åêñïåðòà äóæå âèñîêî¿ êâàë³ô³êàö³¿ â ïåâí³é ïðåäìåòí³é îáëàñò³. Óçàãàëüíåíà åêñïåðòíà ñèñòåìà ì³ñòèòü áàçó çíàíü ç ôàêòàìè òà ïðàâèëàìè êîíêðåòíî¿ ïðåäìåòíî¿ îáëàñò³, ìåõàí³çì âèâåäåííÿ, áëîê íàáóòòÿ çíàíü òà áëîê ïîÿñíåíü.

²ñíóº áàãàòî ð³çíèõ çàñòîñóâàíü åêñïåðòíèõ ñèñòåì, à ñàìå: ñêëàäàííÿ пðîãíîçó, ïîñòàíîâêà ä³àãíîçó, çàäà÷³ óïðàâë³ííÿ, ïëàíóâàííÿ, íà÷àííÿ òîùî.

Â íàø ÷àñ ³ñíóº äâà ï³äõîäè äëÿ ïðîåêòóâàííÿ íàâ÷àëüíèõ ïðîãðàì: åìï³ðè÷íèé òà òåîðåòè÷íèé. Ïåðøèé ï³äõ³ä áàçóºòüñÿ íà çäîðîâîìó ãëóçä³ òà äîñâ³ä³ ðîçðîáêè íàâ÷àëüíèõ ïðîãðàì ³ íåð³äêî íåõòóþòüñÿ òåîðåòè÷í³ îñíîâè ñêëàäàííÿ íàâ÷àëüíèõ ïðîãðàì. Ïðîòå, îñíîâíèé íàïðÿìîê ïðîåêòóâàííÿ ïîâèíåí ñïèðàòèñÿ íà òåîðåòè÷íèé ôóíäàìåíò ïðîãðàìîâàíîãî íàâ÷àííÿ. Â ïðèíöèï³, ìîæíà ñòâîðèòè äåê³ëüêà åôåêòèâíèõ íàâ÷àëüíèõ ïðîãðàì ³ åìï³ðè÷íî áåç òåîð³¿ íàâ÷àííÿ, àëå ìàñîâó ðîçðîáêó íàâ÷àëüíèõ ïðîãðàì, íå ìàþ÷è òåîðåòè÷íî¿ áàçè, íàâðÿä ÷è ìîæíà çðîáèòè.

Ïðîåêòóâàííÿ íàâ÷àëüíèõ ïðîãðàì - öå áàãàòîð³âíåâèé ïðîöåñ. Â íüîìó ìîæíà âèä³ëèòè òàê³ ð³âí³: êîíöåïòóàëüíèé, òåõíîëîã³÷íèé, îïåðàö³îíàëüíèé òà ð³âåíü ðåàë³çàö³¿.

Íà êîíöåïòóàëüíîìó ð³âí³ çàäàºòüñÿ ìîäåëü íàâ÷àííÿ: íàâ÷àííÿ îïèñóºòüñÿ ÿê ñèñòåìà, ÿêà ñêëàäàºòüñÿ ç äâîõ ï³äñèñòåì - ä³ÿëüíîñò³ âèêëàäà÷à òà ä³ÿëüíîñò³ òèõ, õòî áóäå íàâ÷àòèñÿ. Òóò îïèñóºòüñÿ ïðèíöèï íàâ÷àííÿ, îñíîâí³ êîìïîíåíòè ó÷áîâî¿ ä³ÿëüíîñò³ ³ òîãî ð³âíÿ ÿêèé ïîâèíåí áóòè ñôîðìîâàíèé â ïðîöåñ³ íàâ÷àííÿ. Íà òåõíîëîã³÷íîìó ð³âí³ ïðîåêò íàâ÷àëüíî¿ ïðîãðàìè îïèñóºòüñÿ ó âèãëÿä³ ñïîñîáó êåðóâàííÿ ó÷áîâîþ ä³ÿëüí³ñòþ íà ì³êðîð³âí³. Íà îïåðàö³îíàëüíîìó ð³âí³ ïîðöåñ íàâ÷àííÿ îïèñóºòüñÿ ÿê ðîçâ`ÿçîê äèäàêòè÷íî¿ çàäà÷³. Âêàçóºòüñÿ, ÿê³ ôóíêö³¿ íåñå êîìï`þòåð, à òàêîæ îñíîâí³ øëÿõè óïðàâë³ííÿ ó÷áîâîþ ä³ÿëüí³ñòþ.

Ð³âåíü ðåàë³çàö³¿ ì³ñòèòü äâà ï³äð³âíÿ: ïåäàãîã³÷íî¿ ðåàë³çàö³¿ òà ïðãðàìíî¿. Ïðîåêò íàâ÷àëüíî¿ ïðîãðàìè íà ð³âí³ ïåäàãîã³÷íî¿ ðåàë³çàöè¿ ìîæå áóòè îïèñàíèé ó âèãëÿä³ ñöåíàð³ÿ, ÿêèé âèçíà÷àº ÿê ä³º íàâ÷àëüíà ïðîãðàìà â êîæíèé ìîìåíò íàâ÷àííÿ. Íà ð³âí³ ïðîãðàìíî¿ ðåàë³çàö³¿ ñöåíàð³é ïåðåêëàäàºòüñÿ â ïðîãðàìó äëÿ êîìï`þòåðà.

Основи реалізації програми CURSOVA

Програма написана на мові Паскаль з використанням компілятора Turbo Pascal 7.0 та Turbo Assembler фірми Borland International, Inc. Програма працює в текстовому режимі. Керування відбувається за допомогою системи меню. Саму програму за внутрішньою побудовою можна поділити на три такі частини:

* Система меню
* Система просмотру інформаційних файлів та запуску демонстраційних програм
* Система перетворення текстових файлів на файли, що розпізнаються ограмою CURSOVA.

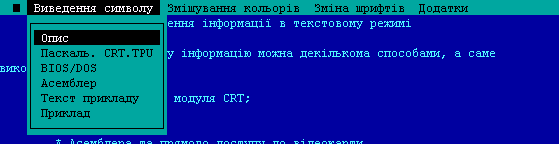
Система меню

Як вже було сказано, система меню забезпечує керування програмою. Це звичайне випадаюче (pop-up) меню. Зовнішній вигляд його зображений на малюнку 1.



Малюнок 1

Перший пункт меню, як ви бачите, виділений. Тобто, при натиснені кнопки Enter, з'явиться підменю з списком можливих дій. Керувати виділеним пунктом можна за допомогою клавіш стрілочок.



Малюнок 2

Як видно, система меню зроблена досить схожою на меню більшості професійних програм і є досить інтуїтивною.

Внутрішньо система меню реалізована у вигляді окремого модуля з назвою SMENU.TPU. Там знаходяться такі костанти, змінні, процедури та функції:

**procedure menu\_init(var m: menu);**

Процедура ініціалізує всі необхідні змінні і взагалі приводить систему меню в стан готовності до роботи. Змінна **m** є складною змінною в якій саме і знаходиться данні для меню.

**procedure menu\_insert\_main(var m: menu; s: string; c: char);**

Процедура вставляє в основне меню ще один пункт в меню **m** з написом **s** та гарячою літерою **c**.

**procedure menu\_insert\_sub(var m: menu; n: integer; s: string; c: char);**

Процедура вставляє в підменю з номером **n** основного меню ще один пункт, з написом **s** та гарячою літерою **c**.

**function menu\_run(m: menu; c1, c2, c3, c4, c5, c6, c7, c8: byte):word;**

Це є основна функція, що і виконує сновну функцію всього модуля. Змінні **c1, c2, c3, c4, c5, c6, c7, c8** слугують для передачі кодів кольорів меню. Всередині функції використовуються процедури:

**procedure screen\_save; assembler;**

**procedure screen\_restore; assembler;,**

друкування підменю тощо. Основний алгоритм такий: намалювати пункти меню, прочитати клавішу з клавіатури і в залежності від клавіші пересунути вказівник вліво, вправо, вгору чи вниз відповідно. Кожен раз при перемальовці екрану, викликається функція **screen\_restore** для того щоб підпункти меню не накладалися один на одного. Для забезпечення швидкості перемальування екрану, функції зберження та відновлення екрану написані на асемблері. Результат, що видається має тип **word**. В старшому байті знаходиться номер пункту основного меню, а в молодшому - номер його підпункту. Перевірку в основній програмі можна, наприклад, зробити так:

**run:=menu\_run(m, 0, 3, 15, 0, 0, 3, 15, 0);**

**case hi(run) of**

**1: case lo(run) of**

**1: ...** Дії до пункту 1 підменю 1

**2: ...** Дії до пункту 2 підменю 1

**2: case lo(run) of**

**1: ...** Дії до пункту 1 підменю 2

**2: ...** Дії до пункту 2 підменю 2

**end;**

**end;**

**procedure menu\_done(m: menu);**

Функція деініціалізації, що закінчує роботу меню. Після виклику цієї функції неможна користуватись змінною **m**.

Сама змінна **m** має таку структуру:

**const**

**max\_of\_main = 10;** максимальна кількість пунктів основного меню

**max\_of\_item = 30;** максимальна довжина тексту назви пункта

**type**

**menu = record**

**mains: integer;** кількість пунктів основного меню

**current: integer;** поточний виділений пункт

**item1: array [1..max\_of\_main] of record** масив інформації про кожен з пунктів

**text: string[max\_of\_item];** напис

**letter: char;** гаряча клавіша

**items: integer;** кількість підпунктів

**current: integer;** поточний підпункт

**item2: array [1..10] of record** інформація про підпункти

**text: string[max\_of\_item];** напис

**letter: char;** гаряча клавіша

**able: boolean;** можливість вибирати

**end;**

**end;**

**end;**

Як видно структура меню досить складна, але в ній одній компактно записана інформація про все меню.

Система перегляду інформаційних файлів та запуску демонстраційних програм

Ця система забезпечує перегляд в текстовому режимі файлів з текстовими данними (самі файли текстовими назвати не можна). Вона реалізована однією процедурою  
**procedure showtext(fname: string);,** яка, як параметр приймає ім'я файлу з вихідним текстом і зображує його на екрані. Якщо файл не вміщається на екран, то за допомогою клавіш стрілочок можна прокрутити текст по екрану. Для зручності в програмуванні, вхідні файли мають скаладну структуру. Вони є масивом радків довжини 80:

**type**

**str80 = string[80];**

**var**

**f: file of str80;**

Непер ми можемо організувати прямий доступ до будь-якої компоненти цього файлу і нам не потрібно виділяти пам’ять на збереження текстової інформації. Ми по одному рядку зчитуємо з файлу і виводимо на екран.

**for i:=0 to PSIZE-1 do begin**

**if eof(f) then str:=''**

**else read(f, str);**

**gotoxy(1, i+2);**

**textattr:=11+16\*1;**

**write(str); clreol;**

**end;**

Система перетворення текстових файлів на файли, що розпізнаються ограмою CURSOVA.

Маю сказати, що не дуже зручно редагувати файли у форматі прийнятному програмі CURSOVA. Тому тексти набираються в звичайному текстовому редакторі і за допомогою програми TEXT2STR перетворюються на файли STR. Як параметри програмі TEXT2STR необхідно задати ім’я файлу TXT та ім’я файлу STR. Всю іншу роботу вона зробить сама.

Тема "Пpогpамування вiдеоадаптеpiв" дуже актуальна в наш час, оскiльки вiдеоадаптеp - складова частина вiдеопiдсистеми будь-якого комп`ютеpа. Розpобка пpогpам, що iлюстpують деякi можливостi вiдеоадаптеpiв, а саме:  
виведення символiв, змiна палiтpи та шpифтiв, надають всiм бажаючим змогу детальнiше ознайомитися з цими пpоцесами, а виконання пpогpам на тpьох piвнях: високому (мова пpогpамування Паскаль), сеpедньому (асемблеp з викоpистанням пеpеpивань BIOS/DOS) та низькому (асемблеp без викоpистання пеpеpивань) ще й має за мету поглибити знання студентiв з цих мов пpогpамування. Отже, пiсля  
повного ознайомлення з даною навчальною пpогpамою, студент набуває досвiду в пpогpамуваннi вiдеоадаптеpiв i вже може спpобувати свої сили на цьому попpищi. Hа закiнчення не завадить сказати ще декiлька слiв пpо комп`ютеpнi  
навчальнi засоби взагалi. Застосування нових iнфоpмацiйних технологiй в сфеpi освiти - один з засобiв пiдготовки кадpiв в сучасних умовах вpаховуючi задачi майбутнього. Розв`язок цiєї задачi потpебує пеpегляду всiєї системи освiти, пpи цьому особливу увагу слiд звеpтати на pозвиток фоpм пpедставлення знань, фоpмування iнтелектуальних вмiнь, набуття пpактичних навичкiв в областi застосування новiтнiх засобiв обчислювальної технiки. Пpи цьому повинна змiнитися сама технологiя навчання, бiльше уваги слiд пpидiляти виpобленню умiння самостiйно набувати знання в пpоцесi дослiдницької дiяльностi. Застосування новiтнiх iнфоpмацiйних технологiй дозволить також збiльшити зв`язки мiж пpедметами, що в подальшому може слугувати основою для ствоpення нових iнтегpованих куpсiв навчання. Разом з тим слiд зазначити, що комп`ютеpiзацiя навчання не вiдмiняє i не замiняє пpоцес спiлкування педагога зi студентом, а лише допомагає i тому, i iншому бiльш ефективно пpиймати участь в пpоцесi навчання i пpидiляти бiльше уваги елементам твоpчостi, на якi pанiше не вистачало часу (це стає можливим завдяки бiльш ефективній самостiйній пpацi студента i бiльш наочного пpедставлення нового матеpiалу).

**Використана літератури**

1. Р. Джордейн. Справочник програмиста персональних компьютеров типа IBM PC, XT и AT.
2. П. Абель. Язык ассемблера для IBM PC и программирования.
3. А. Епанешников, В. Епанешников. Программирование в среде Turbo Pascal 7.0.
4. Г. Шилдт. Язык Си для профессионалов.
5. Інфоpмацiйнi технологiї в сучаснiй школi. А.Ф.Веpлань, Л.О.Твеpезовська, В.А.Федоpчук.
6. Пpогpамиpование в сpеде Туpбо Паскаль. Д.Б.Поляков, И.Ю.Кpуглов
7. Пpогpамиpование видеоадаптеpов. А.В.Фpолов, Л.В.Фpолов

Додатки

CURSOVA.PAS

{$M $8000,0,0 }

uses crt, smenu, dos;

procedure Print(x, y, c1: byte; S: string);

begin

GotoXY(x, y);

textattr:=c1;

Write(S);

end;

procedure Frame(x1, y1, x2, y2: byte; S: string; c1, c2: byte; Double: boolean);

var i, k, Leng, High: byte;

begin

Leng:=x2-x1;

High:=y2-y1;

Window(x1, y1, x1+Leng, y1+High);

textattr:=c1;

ClrScr;

Window(1, 1, 80, 25);

if Double=True then Print(x1, y1, c1, 'Й') else Print(x1, y1, c1, 'Ъ');

for i:=1 to Leng do

if Double=True then Print(x1+i, y1, c1, 'Н') else Print(x1+i, y1, c1, 'Д');

if Double=True then Print(x2+1, y1, c1, '»') else Print(x2+1, y1, c1, 'ї');

for i:=1 to High do begin

if Double=True then Print(x1, y1+i, c1, 'є') else Print(x1, y1+i, c1, 'і');

if Double=True then Print(x2+1, y1+i, c1, 'є') else Print(x2+1, y1+i, c1, 'і');

end;

if Double=True then Print(x1, y2, c1, 'И') else Print(x1, y2, c1, 'А');

for i:=1 to Leng do

if Double=True then Print(x1+i, y2, c1, 'Н') else Print(x1+i, y2, c1, 'Д');

if Double=True then Print(x2+1, y2, c1, 'ј') else Print(x2+1, y2, c1, 'Щ');

if S<>'' then Print(x1+(Leng div 2)-(Length(S) div 2), y1, c2, ' '+S+' ');

end;

procedure Cursor(x1, x2: byte);

var

Reg: Registers;

begin

Reg.AH:=1;

Reg.CH:=x1;

Reg.CL:=x2;

Intr($10, Reg);

end;

function UpCasing(S1: string): string;

var

s: string;

b: byte;

begin

S:='';

for B:=1 to Length(S1) do

S:=S+UpCase(S1[B]);

UpCasing:=S;

end;

function Strg(W: LongInt): string;

var

s: string;

begin

Str(W, S);

Strg:=S;

end;

procedure KeyBar(w1, w2, w3, w4, w5, w6, w7, w8, w9, w10: string);

var

k1, k2, k3, k4, k5, k6, k7, k8, k9, k10: string;

begin

k1:=' ';

k2:=' ';

k3:=' ';

k4:=' ';

k5:=' ';

k6:=' ';

k7:=' ';

k8:=' ';

k9:=' ';

k10:=' ';

k1:=w1;

k2:=w2;

k3:=w3;

k4:=w4;

k5:=w5;

k6:=w6;

k7:=w7;

k8:=w8;

k9:=w9;

k10:=w10;

k1[0]:=chr(6);

k2[0]:=chr(6);

k3[0]:=chr(6);

k4[0]:=chr(6);

k5[0]:=chr(6);

k6[0]:=chr(6);

k7[0]:=chr(6);

k8[0]:=chr(6);

k9[0]:=chr(6);

k10[0]:=chr(5);

Print(1, 25, 7+16\*0, '1');

Print(2, 25, 0+16\*3, k1);

Print(8, 25, 7+16\*0, ' 2');

Print(10, 25, 0+16\*3, k2);

Print(16, 25, 7+16\*0, ' 3');

Print(18, 25, 0+16\*3, k3);

Print(24, 25, 7+16\*0, ' 4');

Print(26, 25, 0+16\*3, k4);

Print(32, 25, 7+16\*0, ' 5');

Print(34, 25, 0+16\*3, k5);

Print(40, 25, 7+16\*0, ' 6');

Print(42, 25, 0+16\*3, k6);

Print(48, 25, 7+16\*0, ' 7');

Print(50, 25, 0+16\*3, k7);

Print(56, 25, 7+16\*0, ' 8');

Print(58, 25, 0+16\*3, k8);

Print(64, 25, 7+16\*0, ' 9');

Print(66, 25, 0+16\*3, k9);

Print(72, 25, 7+16\*0, ' 10');

Print(75, 25, 0+16\*3, k10);

GotoXY(79, 25);

TextBackground(3);

ClrEol;

end;

procedure showtext(fname: string);

const

PSIZE = 23;

type

str80 = string[80];

var

f: file of str80;

str: str80;

pos, MPOS: integer;

ch: char;

procedure opening;

begin

assign(f, fname);

{$I-}

reset(f);

{$I+}

if ioresult<>0 then begin

textattr:=7; clrscr;

writeln('Error in opening file: ', fname);

halt;

end;

end;

procedure showfrom(n: integer);

var

i: integer;

begin

seek(f, n);

for i:=0 to PSIZE-1 do begin

if eof(f) then str:=''

else read(f, str);

gotoxy(1, i+2);

textattr:=11+16\*1;

write(str); clreol;

end;

end;

procedure showhead;

begin

gotoxy(1, 1);

textattr:=16\*3;

write('File: ', fname, ' ', pos, '/', MPOS,

' ', pos\*100 div MPOS, '%');

clreol;

end;

procedure help;

begin

frame(10, 5, 70, 22, 'Help', 16\*3, 16\*3, true);

print(13, 7, 16\*3, 'Slawa Pidgorny & Sveta Fialka proudly presents:');

print(13, 9, 16\*3, 'CURSOVA 1.1');

print(13, 11, 16\*3, 'Copycenter (C) 1997 UKMA');

print(13, 12, 16\*3, 'University of "Kiev-Mohyla Academy"');

print(13, 13, 16\*3, 'Department of Computer && System Sciences');

print(13, 15, 16\*3, 'Director of the project: Oletsky Oleksiy Vitaliyevich');

print(13, 17, 16\*3, 'Fullfilled by:');

print(13, 19, 16\*3, ' \* Slawa Pidgorny. <slawa@queen.ukma.kiev.ua>');

print(13, 18, 16\*3, ' \* Sveta Fiyalka. <svetaflk@queen.ukma.kiev.ua>');

print(13, 21, 16\*3, 'Written on Turbo Pascal++ 7.0');

readkey;

end;

begin

opening;

cursor($20, $20);

pos:=0;

MPOS:=filesize(f);

repeat

showfrom(pos);

showhead;

ch:=readkey;

if ch=#27 then break;

if ch=#0 then begin

ch:=readkey;

if (ch=#59) then help;

if (ch=#68) then break;

if (ch=#80) and (pos<MPOS) then inc(pos);

if (ch=#72) and (pos>0) then dec(pos);

if (ch=#73) then if (pos>PSIZE) then dec(pos, PSIZE) else pos:=0;

if (ch=#81) then if (pos<MPOS-PSIZE) then inc(pos, PSIZE) else pos:=MPOS;

end;

until false;

cursor(6, 7);

end;

procedure error(str: string);

begin

writeln(str);

halt;

end;

var

fmen: array [1..10, 1..10] of string[80];

m: menu;

procedure fillmenu;

var

f: text;

mm: boolean;

str: string;

p1, p2, i, j: integer;

begin

assign(f, 'cursova.dat');

reset(f);

mm:=true;

p1:=1;

p2:=1;

while (not eof(f)) do begin

readln(f, str);

if mm then begin

menu\_insert\_main(m, str, ' ');

mm:=false;

end else if str='' then begin

mm:=true;

inc(p1); if (p1>10) then error('max no of submenus: 10');

p2:=1;

end else begin

fmen[p1, p2]:=copy(str, 1, pos(' ', str)-1);

inc(p2); if (p2>10) then error('max no of items in submenu: 10');

menu\_insert\_sub(m, p1+1, copy(str, pos(' ', str)+1, 255), ' ');

end;

end;

end;

var

run: word;

extt: string;

begin

textattr:=7; clrscr;

menu\_init(m);

menu\_insert\_main(m, 'ю', '‚');

menu\_insert\_sub(m, 1, 'Exit', 'x');

fillmenu;

repeat

keybar('Help', '', '', '', '', '', '', '', '', 'Menu');

run:=menu\_run(m, 0, 3, 15, 0, 0, 3, 15, 0);

case hi(run) of

1: break;

2..10: begin

extt:=upcasing(copy(fmen[hi(run)-1, lo(run)],

pos('.', fmen[hi(run)-1, lo(run)])+1, 255));

if extt='STR' then showtext(fmen[hi(run)-1, lo(run)]);

if extt='EXE' then begin

textattr:=7;

clrscr;

SwapVectors;

Exec(fmen[hi(run)-1, lo(run)], '');

SwapVectors;

if DosError <> 0 then error('Dos error #'+strg(DosError));

readkey;

end;

end;

end;

until false;

menu\_done(m);

textattr:=7; clrscr;

writeln('Program created as a cursova by');

writeln(' \* Slawa Pidgorny. <slawa@queen.ukma.kiev.ua>');

writeln(' \* Sveta Fiyalka. <svetaflk@queen.ukma.kiev.ua>');

writeln('using Turbo Pascal 7.0.');

end.

SMENU.PAS

unit smenu;

interface

const

max\_of\_main = 10;

max\_of\_item = 30;

type

menu = record

mains: integer;

current: integer;

item1: array [1..max\_of\_main] of record

text: string[max\_of\_item];

letter: char;

items: integer;

current: integer;

item2: array [1..10] of record

text: string[max\_of\_item];

letter: char;

able: boolean;

end;

end;

end;

procedure menu\_init(var m: menu);

procedure menu\_insert\_main(var m: menu; s: string; c: char);

procedure menu\_insert\_sub(var m: menu; n: integer; s: string; c: char);

function menu\_run(m: menu; c1, c2, c3, c4, c5, c6, c7, c8: byte): word;

procedure menu\_done(m: menu);

implementation

uses

crt;

procedure Cursor(x1, x2: byte); assembler;

asm

mov ah, 1

mov ch, x1

mov cl, x2

int 10h

end;

function strings(a: integer): string;

var

s: string;

i: integer;

begin

s:='';

for i:=1 to a do begin

s:=s+' ';

end;

strings:=s;

end;

procedure colors(a, b: byte);

begin

textcolor(a); textbackground(b);

end;

procedure colour(a: byte);

begin

textattr:=a;

end;

procedure print(x, y: integer; c: byte; s: string);

begin

gotoxy(x, y);

colour(c);

write(s);

end;

procedure frame(x1, y1, x2, y2: byte; S: string; c1, c2: byte; Double: boolean);

var

i, k, Leng, High: byte;

begin

Leng:=x2-x1;

High:=y2-y1;

Window(x1, y1, x1+Leng, y1+High);

Colour(c1);

ClrScr;

Window(1, 1, 80, 25);

if Double=True then Print(x1, y1, c1, 'Й') else Print(x1, y1, c1, 'Ъ');

for i:=1 to Leng do

if Double=True then Print(x1+i, y1, c1, 'Н') else Print(x1+i, y1, c1, 'Д');

if Double=True then Print(x2+1, y1, c1, '»') else Print(x2+1, y1, c1, 'ї');

for i:=1 to High do begin

if Double=True then Print(x1, y1+i, c1, 'є') else Print(x1, y1+i, c1, 'і');

if Double=True then Print(x2+1, y1+i, c1, 'є') else Print(x2+1, y1+i, c1, 'і');

end;

if Double=True then Print(x1, y2, c1, 'И') else Print(x1, y2, c1, 'А');

for i:=1 to Leng do

if Double=True then Print(x1+i, y2, c1, 'Н') else Print(x1+i, y2, c1, 'Д');

if Double=True then Print(x2+1, y2, c1, 'ј') else Print(x2+1, y2, c1, 'Щ');

if S<>'' then Print(x1+(Leng div 2)-(Length(S) div 2), y1, c2, ' '+S+' ');

end;

procedure menu\_init(var m: menu);

begin

m.mains:=0;

m.current:=1;

end;

procedure menu\_insert\_main(var m: menu; s: string; c: char);

begin

inc(m.mains);

m.item1[m.mains].text:=s;

m.item1[m.mains].letter:=c;

m.item1[m.mains].current:=1;

end;

procedure menu\_insert\_sub(var m: menu; n: integer; s: string; c: char);

begin

inc(m.item1[n].items);

m.item1[n].item2[m.item1[n].items].text:=s;

m.item1[n].item2[m.item1[n].items].letter:=c;

end;

function menu\_run(m: menu; c1, c2, c3, c4, c5, c6, c7, c8: byte): word;

var

pos\_tab: array[1..max\_of\_main] of record

x, l: integer;

end;

scr1: array[0..3999] of byte;

procedure screen\_save; assembler;

asm

push ds

mov ax, 0B800h

mov ds, ax

xor si, si

mov ax, seg scr1

mov es, ax

mov di, offset scr1

mov cx, 1000

cld

db $66; rep movsw

pop ds

end;

procedure screen\_restore; assembler;

asm

push ds

mov ax, seg scr1

mov ds, ax

mov si, offset scr1

mov ax, 0B800h

mov es, ax

xor di, di

mov cx, 1000

cld

db $66; rep movsw

pop ds

end;

procedure create\_pos\_tab;

var

i, p: integer;

begin

p:=2;

for i:=1 to m.mains do begin

pos\_tab[i].x:=p;

inc(p, length(m.item1[i].text)+2);

end;

end;

procedure main\_show(m: menu; sel: integer);

var

i: integer;

begin

gotoxy(1, 1);

colors(c1, c2);

clreol;

for i:=1 to m.mains do begin

if i=sel then begin

colors(c3, c4);

end else begin

colors(c1, c2);

end;

gotoxy(pos\_tab[i].x, 1);

write(' '+m.item1[i].text+' ');

end;

end;

function sub\_menu\_max\_len(m: menu; n: integer): integer;

var

i, max: integer;

begin

max:=0;

for i:=1 to m.item1[n].items do begin

if max<length(m.item1[n].item2[i].text) then begin

max:=length(m.item1[n].item2[i].text);

end;

end;

sub\_menu\_max\_len:=max;

end;

procedure sub\_show(m: menu; sel1, sel2: integer);

var

i: integer;

begin

frame(pos\_tab[sel1].x, 2, 1+pos\_tab[sel1].x+sub\_menu\_max\_len(m, sel1)+1,

m.item1[sel1].items+3, '', c5+16\*c6, c7+16\*c8, false);

for i:=1 to m.item1[sel1].items do begin

if i=sel2 then begin

colors(c3, c4);

end else begin

colors(c1, c2);

end;

gotoxy(pos\_tab[sel1].x+1, 2+i);

write(' '+m.item1[sel1].item2[i].text+' '+strings(

sub\_menu\_max\_len(m, sel1)-length(m.item1[sel1].item2[i].text)));

end;

end;

var

main\_menu\_position: integer;

sub\_menu\_position: integer;

sub\_menu\_open: boolean;

ch: char;

saved\_x, saved\_y: byte;

saved\_colors: byte;

begin

cursor($20, $20);

saved\_colors:=textattr;

saved\_x:=wherex;

saved\_y:=wherey;

main\_menu\_position:=m.current;

sub\_menu\_open:=false;

create\_pos\_tab;

screen\_save;

repeat

screen\_restore;

main\_show(m, main\_menu\_position);

if sub\_menu\_open then begin

sub\_show(m, main\_menu\_position, sub\_menu\_position);

end;

ch:=readkey;

if ch=#0 then begin

ch:=readkey;

if ch=#75{Left} then begin

if main\_menu\_position>1 then begin

m.item1[main\_menu\_position].current:=sub\_menu\_position;

dec(main\_menu\_position);

sub\_menu\_position:=m.item1[main\_menu\_position].current;

end;

end;

if ch=#77{Right} then begin

if main\_menu\_position<m.mains then begin

m.item1[main\_menu\_position].current:=sub\_menu\_position;

inc(main\_menu\_position);

sub\_menu\_position:=m.item1[main\_menu\_position].current;

end;

end;

if sub\_menu\_open then begin

if ch=#72{Up} then begin

if sub\_menu\_position>1 then begin

dec(sub\_menu\_position);

end;

end;

if ch=#80{Down} then begin

if sub\_menu\_position<m.item1[main\_menu\_position].items then begin

inc(sub\_menu\_position);

end;

end;

end; {sub\_menu\_open}

end; {ch=#0}

if ch=#13{Enter} then begin

if not sub\_menu\_open then begin

sub\_menu\_open:=true;

sub\_menu\_position:=m.item1[main\_menu\_position].current;

end else begin

menu\_run:=main\_menu\_position\*256+sub\_menu\_position;

break;

end;

end;

if ch=#27{Esc} then begin

menu\_run:=0;

break;

end;

until false;

screen\_restore;

textattr:=saved\_colors;

gotoxy(saved\_x, saved\_y);

cursor(6, 7);

end;

procedure menu\_done(m: menu);

begin

end;

end.

TEXT2STR.PAS

type

str80 = string[80];

var

f1: text;

f2: file of str80;

str: string;

str2: str80;

begin

if paramcount<>2 then exit;

assign(f1, paramstr(1));

reset(f1);

assign(f2, paramstr(2));

rewrite(f2);

write('Processing');

while not eof(f1) do begin

readln(f1, str);

str2:=copy(str, 1, 80);

write(f2, str2);

write('.');

end;

writeln;

close(f2);

close(f1);

end.

CURSOV16.PAS

uses

crt;

procedure a1;

begin

textColor(14);

textBackground(3);

write('Slawa');

textAttr:=7;

writeLn;

end;

procedure a31; assembler;

asm

mov ah, 9 {function 9}

mov al, 'a' {char 'a'}

xor bh, bh {video page 0}

mov bl, 16\*3+14 {color}

mov cx, 5 {print 5 times}

int 10h {printing}

end;

procedure a32;

const

str: array [0..6] of char ='slawa'+#13+#10;

begin

asm

push bp {push used registers to stack}

push es

mov ax, seg str {es is a segment register of text}

mov es, ax

mov bp, offset str {es:bp is an actual string location}

mov ah, 13h {funtion 13h}

mov al, 1 {only text}

mov bh, 0 {video page}

mov bl, 16\*3+14 {color}

mov cx, 7 {chars' counter}

mov dh, 10 {coordinates}

mov dl, 10

int 10h {printing}

pop es {restore registers}

pop bp

end;

end;

procedure a4; assembler;

const

color: byte = 16\*3+14; {color}

str: array [0..4] of char

= 'slawa'; {string to print}

strlen: word = 5; {length of that string}

asm

mov ax, 0B800h {mov es, 0B800h}

mov es, ax

mov cx, strlen {cx - counter of chars}

mov si, offset str {ds:si - text}

xor di, di {es:di - video memory}

@a:

lodsb {load byte from ds:si}

stosb {store byte to es:di}

mov al, color

stosb {store color to es:di}

loop @a {while there is more chars to print}

end;

begin

textattr:=7; clrscr; a1; readkey;

textattr:=7; clrscr; a31; writeln; textattr:=7; clreol; readkey;

textattr:=7; clrscr; a32; readkey;

textattr:=7; clrscr; a4; readkey;

end.

CURSOV26.PAS

uses crt, graph;

procedure a5;

var

aVGA, aVGALO: integer;

begin

aVGA:=VGA; aVGALO:=VGAHI;

initGraph(aVGA, aVGALO, ''); {entering video mode 640x480x16}

setFillStyle(1, 13);

bar(10, 10, 100, 100); {making a filled rectangle}

readkey; {press any key}

setPalette(13, 3); {palette changing}

end;

procedure a6; assembler;

asm

{palette changing}

mov ah, 10h {function 10h for palette operations}

mov al, ah {subfunct 10h for changin 1 palette color}

mov bx, 14 {14 color number}

mov bh, 10 {red 10}

mov ch, 40 {green 40}

mov cl, 30 {blue 30}

int 10h {changing}

end;

procedure a7; assembler;

const

n: byte = 14; {color}

r: byte = 63; {red}

g: byte = 23; {green}

b: byte = 25; {blue}

asm

mov dx, 3C8h {port #3C8h}

mov al, n {outing color number}

out dx, al

inc dx {port #3C9h}

mov al, r {outing red}

out dx, al

mov al, g {outing green}

out dx, al

mov al, b {outing blue}

out dx, al

end;

procedure tograph(a: word); assembler;

asm

{entering mode13h - 320x200x256}

mov ax, 13h

int 10h

{filling all the screen with 14th color}

mov ax, 0A000h

mov es, ax {mov es, 0A000h}

mov cx, 64000/2 {cx - counter}

xor di, di {begin of video memory is at 0A000:0}

cld {move forward}

mov ax, a {color to put}

rep stosw {filling}

end;

procedure totext; assembler;

asm

mov ax, 3 {changing video mode to text one}

int 10h

end;

begin

textattr:=7; clrscr; a5; readkey;

tograph(14+256\*14);

readkey;

a6;

readkey;

a7;

readkey;

totext;

end.

CURSOV33.PAS

uses

crt;

procedure a8; assembler;

asm

mov ah, 11h {function 11h - bios font operations}

mov al, 11h {8x14}

mov bl, 0 {font plane}

int 10h {changing}

call readkey; {let you see the changes}

mov ah, 11h {function 11h - bios font operations}

mov al, 12h {8x8}

mov bl, 0 {font plane}

int 10h {changing}

call readkey; {let you see the changes}

mov ah, 11h {function 11h - bios font operations}

mov al, 14h {8x16}

mov bl, 0 {font plane}

int 10h {changing}

call readkey; {let you see the changes}

end;

begin

writeln('That''s a text mode font changing demonstration');

a8;

end.