Реализация списка

Необходимо реализовать список вида:

***Техническое описание программы.***

В программе предусмотрена работа со списком, которая включает в себя:

* Создание нового вписка;
* Добавление элемента в список;
* Вывод списка на дисплей;
* Сохранение данных списка в файл;
* Чтение данных из файла;
* Удаление списка из памяти компьютера;
* Поиск элемента в списке;
* Сортировка списка;
* Удаление элемента списка.

***Спецификация программы.***

Ввод данных в программу может осуществляться двумя способами: ввод с клавиатуры или из файла.

Для работы с файлом необходимо на соответствующий запрос программы ввести имя файла, из которого будут взяты данные для построения списка.

Для нормальной работы программы требуется PC совместимый компьютер и компилятор Borland 3.01 и выше. При использование иного сочетая характеристик системы на которой будет тестироваться программа возможны некоторые расхождения с результатами теста, но в основном ничего страшного произойти не должно.

***Текст программы.***

#include <iostream.h>

#include <fstream.h>

class List

 {struct Tree

 {int Body;

 Tree \*LP;

 Tree \*RP;

 Tree(int Bdy=0) {Body=Bdy; LP=NULL; RP=NULL;}

 ~Tree() {Body=0; LP=NULL; RP=NULL;}

 };

 public:

 List(int Digit=0);

 Tree \*Root;

 List \*LNext;

 List \*LPrev;

 };

List::List(int Digit)

 {Root=NULL;

 for (int i=Digit\*10; i<Digit\*10+10; i++)

 {Tree \*PTree;

 PTree=new Tree(i);

 PTree->LP=NULL;

 PTree->RP=NULL;

 if (Root==NULL)

 Root=PTree;

 else

 {Tree \*PTree1=Root;

 do

 {if (PTree1->LP!=NULL)

 PTree1=PTree1->LP;}

 while (PTree1->LP!=NULL);

 PTree1->LP=PTree;

 PTree=NULL; PTree1=NULL;

 }

 }

 }

class TreeWork : private List

 {public:

 void TreeWorkStart();

 private:

 int ElementQuantity;

 int Mass;

 int i;

 List \*BegP;

 List \*PList;

 int MainMenu();

 int Work(int Task);

 int MakeNewList();

 int AddElements();

 int PrintList();

 void EraseList();

 int DeleteElement();

 int FindElement();

 int SubMenu();

 int SubWork(int Task);

 int SortByIncrease();

 int SortByDecrease();

 int SaveList();

 int OpenList();

 protected:

 void GoThroughTree(Tree \*L);

 void Erase(Tree \*L);

 };

int TreeWork::MainMenu()

 {cout<<endl<<"Main Menu:"<<endl<<endl;

 cout<<" 1. Make New List." <<endl;

 cout<<" 2. Add Element." <<endl;

 cout<<" 3. Print List." <<endl;

 cout<<" 4. Delete Element."<<endl;

 cout<<" 5. Save List." <<endl;

 cout<<" 6. Erase List." <<endl;

 cout<<" 7. Open File." <<endl;

 cout<<" 8. Find Element." <<endl;

 cout<<" 9. Sort List." <<endl;

 cout<<" 0. Exit." <<endl;

 cout<<endl<<"Your choice : ";

 int i;

 do

 {cin>>i;

 if (i<0 || i>9) cout<<endl<<"Error! Try again : ";

 }

 while (i<0 || i>9);

 return i;

 }

int TreeWork::SubMenu()

 {cout<<endl<<"Sub Menu:"<<endl;

 cout<<"1. Sort list by increase."<<endl;

 cout<<"2. Sort list by decrease."<<endl<<endl;

 int i;

 cout<<"Your choice: ";

 do

 {cin>>i;

 if (i<1 || i>2) cout<<endl<<"Error! Try again : ";

 }

 while (i<1 || i>2);

 return i;

 }

int TreeWork::SubWork(int Task)

 {switch (Task)

 {case 1 : SortByIncrease(); break; //Increase

 case 2 : SortByDecrease(); break; //Decrease

 }

 return 0;

 }

int TreeWork::Work(int Task)

 {switch (Task)

 {case 1 : ElementQuantity=MakeNewList(); break; //Make New List

 case 2 : ElementQuantity+=AddElements(); break; //Add Element

 case 3 : PrintList(); break; //Print List

 case 4 : DeleteElement(); break; //Delete Element

 case 5 : SaveList(); break; //Save List

 case 6 : ElementQuantity=0; EraseList(); break; //Erase List

 case 7 : OpenList(); break; //Open File

 case 8 : FindElement(); break; //Find Element

 case 9 : SubWork(SubMenu()); break; //Sort List

 case 0 : EraseList(); return -1; //Exit

 }

 return 0;

 }

void TreeWork::TreeWorkStart()

 {ElementQuantity=0;

 do {} while (Work(MainMenu())!=-1);

 }

int TreeWork::MakeNewList()

 {if (BegP!=NULL)

 {cout<<endl<<"You have existing list!"; return 0;}

 int Quant;

 cout<<endl<<"Input quantity of elements: ";

 do

 {cin>>Quant;

 if (Quant<1)

 cout<<endl<<"Error! Try againt: ";

 }

 while (Quant<1);

 for (int i=0; i<Quant; i++)

 {cout<<endl<<"Input digit: ";

 int Digit; cin>>Digit;

 PList=new List(Digit);

 if (BegP==NULL)

 {BegP=PList;

 BegP->LNext=BegP;

 BegP->LPrev=BegP;

 PList=NULL;}

 else

 {List \*PList1=BegP->LPrev;

 if (PList1==BegP)

 {BegP->LNext=PList;

BegP->LPrev=PList;

PList->LNext=BegP;

PList->LPrev=BegP;

PList=NULL; PList1=NULL;}

 else

 {BegP->LPrev=PList;

PList1->LNext=PList;

PList->LNext=BegP;

PList->LPrev=PList1;

PList=NULL; PList1=NULL;}

 }

 }

 return Quant;

 }

int TreeWork::AddElements()

 {if (BegP==NULL)

 {MakeNewList(); return 0;}

 int Quant;

 cout<<endl<<"Input quantity of elements: ";

 do

 {cin>>Quant;

 if (Quant<1)

 cout<<endl<<"Error! Try again: ";

 }

 while (Quant<1);

 for (int i=0; i<Quant; i++)

 {cout<<endl<<"Input digit: ";

 int Digit;

 cin>>Digit;

 PList=new List(Digit);

 List \*PList1=BegP->LPrev;

 if (PList1==BegP)

 {BegP->LNext=PList;

 BegP->LPrev=PList;

 PList->LPrev=BegP;

 PList->LNext=BegP;

 PList1=NULL; PList=NULL;}

 else

 {BegP->LPrev=PList;

 PList->LNext=BegP;

 PList->LPrev=PList1;

 PList1->LNext=PList;

 PList=NULL; PList1=NULL;}

 }

 return Quant;

 }

int TreeWork::PrintList()

 {if (BegP==NULL)

 {cout<<endl<<"The list is empty!"<<endl; return -1;}

 cout<<endl<<"Total elements: "<<ElementQuantity<<endl;

 PList=BegP;

 int i=1;

 do

 {cout<<endl<<" "<<i<<" element: "<<endl;

 GoThroughTree(PList->Root);

 cout<<endl;

 i++;

 PList=PList->LNext;}

 while (PList!=BegP);

 return 0;

 }

void TreeWork::GoThroughTree(Tree \*L)

 {Tree \*PL=L, \*PL1;

 if (PL->LP!=NULL)

 {PL1=PL;

 PL=PL->LP;

 cout<<"("<<PL1->Body<<","<<PL->Body<<") ";

 GoThroughTree(PL);}

 if (PL->RP!=NULL)

 {PL1=PL;

 PL=PL->RP;

 cout<<"("<<PL1->Body<<","<<PL->Body<<") ";

 GoThroughTree(PL);}

 }

void TreeWork::Erase(Tree \*L)

 {Tree \*PL=L;

 if (PL->LP!=NULL)

 {PL=PL->LP;

 Erase(PL);}

 if (PL->RP!=NULL)

 {PL=PL->RP;

 Erase(PL);}

 PL->LP=NULL;

 PL->RP=NULL;

 }

void TreeWork::EraseList()

 {if (BegP!=NULL)

 {do

 {List \*PList1=BegP->LNext;

 PList=PList1->LNext;

 BegP->LNext=PList;

 PList->LPrev=BegP;

 Erase(PList1->Root);

 delete [] PList1;

 }

 while (PList!=BegP);

 BegP=NULL; PList=NULL;

 }

 }

int TreeWork::DeleteElement()

 {cout<<endl<<"Input number of element: ";

 int Number=0;

 cin>>Number;

 if (Number>ElementQuantity || Number<0)

 {cout<<endl<<"Error!"; return -1;}

 Number--;

 PList=BegP;

 for (int i=0; i<Number; i++)

 PList=PList->LNext;

 List \*PList1=PList->LNext, \*PList2=PList->LPrev;

 if (PList==BegP)

 {PList1->LPrev=PList2;

 PList2->LNext=PList1;

 PList->LNext=NULL;

 PList->LPrev=NULL;

 delete [] PList;

 BegP=PList1;

 PList1=NULL; PList2=NULL;}

 else

 {PList1->LPrev=PList2;

 PList2->LNext=PList1;

 PList->LNext=NULL;

 PList->LPrev=NULL;

 delete [] PList;

 PList1=NULL; PList2=NULL;}

 ElementQuantity--;

 return 0;

 }

int TreeWork::FindElement()

 {cout<<endl<<"Input number, you want to find: ";

 int Number=0;

 cin>>Number;

 PList=BegP;

 do

 {Tree \*PT=PList->Root;

 if (Number>PT->Body && Number<PT->Body+10)

 {cout<<endl<<"Element with this number: "<<endl;

 GoThroughTree(PList->Root);

 PList=NULL; cout<<endl; return 0;}

 PList=PList->LNext;

 }

 while (PList!=BegP);

 cout<<endl<<"There aren't such number in list!"<<endl;

 PList=NULL;

 return -1;

 }

int TreeWork::SortByIncrease()

 {

 if(BegP==NULL) {cout<<endl<<"Error!"<<endl; return -1;}

 List \*PList1=BegP; PList=BegP;

 do

 {do

 {if (PList1->Root->Body>PList->Root->Body)

 {Tree \*PT;

PT=PList1->Root;

PList1->Root=PList->Root;

PList->Root=PT;

PT=NULL;}

 PList1=PList1->LNext;

 }

 while (PList1!=BegP);

 PList=PList->LNext;

 }

 while (PList!=BegP);

 return 0;

 }

int TreeWork::SortByDecrease()

 {

 if(BegP==NULL) {cout<<endl<<"Error!"<<endl; return -1;}

 List \*PList1=BegP; PList=BegP;

 do

 {do

 {if (PList1->Root->Body<PList->Root->Body)

 {Tree \*PT;

PT=PList1->Root;

PList1->Root=PList->Root;

PList->Root=PT;

PT=NULL;}

 PList1=PList1->LNext;

 }

 while (PList1!=BegP);

 PList=PList->LNext;

 }

 while (PList!=BegP);

 return 0;

 }

int TreeWork::SaveList()

 {if (BegP==NULL)

 {cout<<endl<<"The list is empty!"<<endl; return -1;}

 ofstream F;

 char \*FileName=new char[25];

 cout<<endl<<"Input file name: "; cin>>FileName;

 F.open(FileName);

 PList=BegP;

 do

 {i=0;

 Mass=PList->Root->Body;

 PList=PList->LNext;

 if (PList!=BegP)

 F<<Mass<<endl;

 else

 F<<Mass;

 }

 while (PList!=BegP);

 F.close();

 delete [] FileName;

 return 0;

 }

int TreeWork::OpenList()

 {if (BegP!=NULL)

 {cout<<endl<<"The list is alredy exist!"<<endl; return -1;}

 cout<<endl<<"Input file name: ";

 char \*FileName=new char[25];

 cin>>FileName;

 ifstream f;

 ElementQuantity=0;

 f.open(FileName);

 char Next;

 Next=f.peek();

 while (Next!=EOF)

 {

 f>>Mass;

 PList=new List(Mass/10);

 if (BegP==NULL)

 {BegP=PList;

 BegP->LNext=BegP;

 BegP->LPrev=BegP;

 PList=NULL;}

 else

 {List \*PList1=BegP->LPrev;

 if (PList1==BegP)

 {BegP->LNext=PList;

BegP->LPrev=PList;

PList->LNext=BegP;

PList->LPrev=BegP;

PList=NULL; PList1=NULL;}

 else

 {BegP->LPrev=PList;

PList1->LNext=PList;

PList->LNext=BegP;

PList->LPrev=PList1;

PList=NULL; PList1=NULL;}

 }

 Next=f.peek();

 ElementQuantity++;

 }

 f.close();

 delete [] FileName;

 return 0;

 }

TreeWork TW;

void main()

 {TW.TreeWorkStart();}

***Результаты работы программы.***

Начало работы:

|  |
| --- |
| Main Menu: 1. Make New List. 2. Add Element. 3. Print List. 4. Delete Element. 5. Save List. 6. Erase List. 7. Open File. 8. Find Element. 9. Sort List. 0. Exit.Your choice : |

Для создания списка выбираем пункт 1:

|  |
| --- |
| Main Menu: 1. Make New List. 2. Add Element. 3. Print List. 4. Delete Element. 5. Save List. 6. Erase List. 7. Open File. 8. Find Element. 9. Sort List. 0. Exit.Your choice : 1Input kol-vo of elements: |

Вводим количество элементов в списке (предположим 4):

|  |
| --- |
| Main Menu: 1. Make New List. 2. Add Element. 3. Print List. 4. Delete Element. 5. Save List. 6. Erase List. 7. Open File. 8. Find Element. 9. Sort List. 0. Exit.Your choice : 1Input kol-vo of elements: 4Input digit: |

Успешное завершение ввода списка:

|  |
| --- |
| Input kol-vo of elements: 4Input digit: 1Input digit: 2Input digit: 3Input digit: 4Main Menu: 1. Make New List. 2. Add Element. 3. Print List. 4. Delete Element. 5. Save List. 6. Erase List. 7. Open File. 8. Find Element. 9. Sort List. 0. Exit.Your choice : |

После ввода списка попадаем в главное меню где выбираем пункт добавления элемента :

|  |
| --- |
| Input digit: 1Input digit: 2Input digit: 3Input digit: 4Main Menu: 1. Make New List. 2. Add Element. 3. Print List. 4. Delete Element. 5. Save List. 6. Erase List. 7. Open File. 8. Find Element. 9. Sort List. 0. Exit.Your choice : 2Input kol-vo of elements: |

Программа просит ввести количество элементов которое мы хотим добавит к нашему списку. Вводим 1 (для примера):

|  |
| --- |
| Input digit: 2Input digit: 3Input digit: 4Main Menu: 1. Make New List. 2. Add Element. 3. Print List. 4. Delete Element. 5. Save List. 6. Erase List. 7. Open File. 8. Find Element. 9. Sort List. 0. Exit.Your choice : 2Input kol-vo of elements: 1Input digit: |

Далее происходит ввод списка как было описано выше.

После удачного завершения добавления элемента в список мы вновь попадаем в главное меню, где выбираем пункт вывода списка на экран монитора:

|  |
| --- |
|  1 element: 1234 2 element: 2345 3 element: 3456 4 element: 4567 5 element: 2345Main Menu: 1. Make New List. 2. Add Element. 3. Print List. 4. Delete Element. 5. Save List. 6. Erase List. 7. Open File. 8. Find Element. 9. Sort List. 0. Exit.Your choice : |

Быстро проскользнув мимо Ваших глаз непонятные данные вы снова попадаете в главное меню, где выбираете пункт удаление элемента:

|  |
| --- |
|  2 element: 2345 3 element: 3456 4 element: 4567 5 element: 2345Main Menu: 1. Make New List. 2. Add Element. 3. Print List. 4. Delete Element. 5. Save List. 6. Erase List. 7. Open File. 8. Find Element. 9. Sort List. 0. Exit.Your choice : 4Input number of element: 5 |

После чего элемент с введённым номером удален. После удачного завершения удаления элемента Вы снова попадаете в главное меню, где выбираете пункт сохранение списка:

|  |
| --- |
| Main Menu: 1. Make New List. 2. Add Element. 3. Print List. 4. Delete Element. 5. Save List. 6. Erase List. 7. Open File. 8. Find Element. 9. Sort List. 0. Exit.Your choice : 5Input file name: demon13.txt |

После ввода имени файла данные из списка попадают на диск. И вы снова в главном меню, где выбираете удаление списка:

|  |
| --- |
| 5. Save List. 6. Erase List. 7. Open File. 8. Find Element. 9. Sort List. 0. Exit.Your choice : 5Input file name: demon13.txtMain Menu: 1. Make New List. 2. Add Element. 3. Print List. 4. Delete Element. 5. Save List. 6. Erase List. 7. Open File. 8. Find Element. 9. Sort List. 0. Exit.Your choice : 6 |

После чего все данные в списке безвозвратно умирают и Вы опять у главного списка, и выбираете Вы – открыть файл:

|  |
| --- |
| 5. Save List. 6. Erase List. 7. Open File. 8. Find Element. 9. Sort List. 0. Exit.Your choice : 6Main Menu: 1. Make New List. 2. Add Element. 3. Print List. 4. Delete Element. 5. Save List. 6. Erase List. 7. Open File. 8. Find Element. 9. Sort List. 0. Exit.Your choice : 7Input file name: demon13.txt |

После чего ваш список оживает и … не надо “и” - лучше продолжим – Вы в главном меню и Вы выбираете найти элемент:

|  |
| --- |
| Main Menu: 1. Make New List. 2. Add Element. 3. Print List. 4. Delete Element. 5. Save List. 6. Erase List. 7. Open File. 8. Find Element. 9. Sort List. 0. Exit.Your choice :8Input number, you want to find: 2The first element that we fined out:1234 |

Вы вводите то что хотели найти и – находите в строчке ХХХХХ. Главное меню/Сортировка:

|  |
| --- |
| Main Menu: 1. Make New List. 2. Add Element. 3. Print List. 4. Delete Element. 5. Save List. 6. Erase List. 7. Open File. 8. Find Element. 9. Sort List. 0. Exit.Your choice : 9Sub Menu:1. Sort list by increase.2. Sort list by decrease.Your choice: |

Ту мы можем сортировать по возрастанию или убыванию (ведь у человека всегда должен быть выбор!). После сортировки Главное меню/Выход! Всё программа завершилась успешно! А поскольку эти тесты никто не читает, то можно сказать что компьютеры в нашем институте сделаны из дуба.