```
#include <iostream>
#include <iomanip>
#include <fstream>
#include <iomanip>

using namespace std;

typedef struct {
	int horas;
	int minutos;
} tTiempo;

typedef struct {
	float ganancias;
	tTiempo disponible;
} tRider;

const int NUMRIDERS = 8;

typedef tRider tArrayRiders[NUMRIDERS];

typedef struct {
	tTiempo solicitado; // momento en el que se ha solicitado
	tTiempo iniciado; // momento del envio del pedido
	int	duracion; // minutos que tarda el rider en gestionar el envio
	int rider; // rider asignado al pedido
	bool prioritario; // si es prioritario o no
	bool procesado; // si se le ha asignado rider o no
} tPedido;

const int MAX_PEDIDOS = 50;
typedef tPedido tArrayPedidos[MAX_PEDIDOS];

typedef struct {
	int cont;
	tArrayPedidos arrayPedidos;
} tListaPedidos;

const int CENTINELA = -1;

bool menor(const tTiempo& t1, const tTiempo& t2);
tTiempo suma(const tTiempo& tiempo, int min);
void muestraTiempo(const tTiempo& t);

bool carga(tListaPedidos& pedidos);
int siguiente(const tListaPedidos& pedidos);
int eligeRider(tArrayRiders riders);
void planifica(tListaPedidos& pedidos, tArrayRiders rides);

void muestra(const tListaPedidos& listaPedidos, int rider);

void muestra(const tListaPedidos listaPedidos, const tArrayRiders rider);



int main() {

	tListaPedidos pedidos;
	tArrayRiders riders;
	// inicamos los riders. Todos disponibles a partir
	// de las 13:00 y con ganacias 0
	for (int i = 0; i < NUMRIDERS; i++) {
		riders[i].disponible.horas = 13;
		riders[i].disponible.minutos = 0;
		riders[i].ganancias = 0;
	}
	if (carga(pedidos)) {
		planifica(pedidos, riders);
		muestra(pedidos, riders);
	}
	else {
		cout << "Error de apertura del fichero" << endl;
	}
	return 0;
}

bool menor(const tTiempo& t1, const tTiempo& t2) {
	bool resultado;
	if (t1.horas < t2.horas) {
		resultado = true;
	}
	else {
		if (t1.horas > t2.horas) {
			resultado = false;
		}
		else { // horas iguales
			if (t1.minutos < t2.minutos) {
				resultado = true;
			}
			else {
				resultado = false;
			}
		}
	}
	return resultado;
}

tTiempo suma(const tTiempo& tiempo, int min) {
	int minutosTotales = tiempo.minutos + min;
	tTiempo resultado;
	if (minutosTotales < 60) {
		resultado.horas = tiempo.horas;
		resultado.minutos = minutosTotales;
	}
	else {
		resultado.horas = tiempo.horas + 1;
		resultado.minutos = minutosTotales - 60;
	}
	int horas = minutosTotales / 60;
	int minutos = minutosTotales % 60;
	return resultado;
}

void muestraTiempo(const tTiempo& t) {
	if (t.horas < 10) cout << "0" << t.horas;
	else cout << t.horas;
	cout << ":";
	if (t.minutos < 10) cout << "0" << t.minutos;
	else cout << t.minutos;
}


bool carga(tListaPedidos& pedidos) {
	ifstream archivo;
	bool apertura = true;
	archivo.open("pedidos.txt");
	if (archivo.is_open()) {
		pedidos.cont = 0;
		int horas;
		tPedido pedido;
		archivo >> horas;
		while (horas != CENTINELA) {
			pedido.solicitado.horas = horas;
			archivo >> pedido.solicitado.minutos;
			archivo >> pedido.duracion;
			archivo >> pedido.prioritario;
			pedido.procesado = false;
			tTiempo t = suma(pedido.solicitado, pedido.duracion);
			pedido.iniciado = t;
			pedidos.arrayPedidos[pedidos.cont] = pedido;
			pedidos.cont++;
			archivo >> horas;
		}
		archivo.close();
		return true;
	}
	else {
		apertura = false;
	}
}

int siguiente(const tListaPedidos& pedidos) {
	int i = 0;
	bool encontrado = false;
	while (i < pedidos.cont && pedidos.arrayPedidos[i].procesado) {
		i++;
	}
	int primerNoProcesado = i;
	while (i < pedidos.cont && !encontrado) {
		if (!pedidos.arrayPedidos[i].procesado) {
			if (pedidos.arrayPedidos[i].prioritario) {
				encontrado = true;
			}
			else i++;
		}
		else i++;
	}
	int indice;
	if (encontrado) {
		indice = i;
	}
	else {
		indice = primerNoProcesado;
	}
	return indice;
}


int eligeRider(tArrayRiders riders) {
	int indiceMin = 0;
	tTiempo minimo;
	minimo.horas = 24;
	minimo.minutos = 0;
	for (int i=0; i < NUMRIDERS; i++){
		if (menor(riders[i].disponible,minimo)) {
			minimo = riders[i].disponible;
			indiceMin = i;

		}
	}
	return indiceMin;
}

void planifica(tListaPedidos& pedidos, tArrayRiders riders) {
	int cont = 0;
	int sig;
	int r;
	sig = siguiente(pedidos); // indice del siguiente pedido a procesar
	while (cont < pedidos.cont) {

		r = eligeRider(riders); // indice del rider que procesa el pedido

		// calculamos la hora a la que se enviar□ el pedido

		if (menor(pedidos.arrayPedidos[sig].solicitado, riders[r].disponible)) {
			pedidos.arrayPedidos[sig].iniciado = riders[r].disponible;
		}
		else {
			pedidos.arrayPedidos[sig].iniciado = pedidos.arrayPedidos[sig].solicitado;
		}

		pedidos.arrayPedidos[sig].rider = r;
		pedidos.arrayPedidos[sig].procesado = true;

		riders[r].ganancias += pedidos.arrayPedidos[sig].duracion * 0.21;
		riders[r].disponible = suma(pedidos.arrayPedidos[sig].iniciado,pedidos.arrayPedidos[sig].duracion);

		cont++;
		cout << "Pedido a las ";
		muestraTiempo(pedidos.arrayPedidos[sig].solicitado);
		cout << " - Prioridad: ";
		if (pedidos.arrayPedidos[sig].prioritario) cout << "S - ";
		else cout << "N - ";
		cout << setw(2) << pedidos.arrayPedidos[sig].duracion << "'" << " - Rider " << r + 1;
		cout << " - Envio: ";
		muestraTiempo(pedidos.arrayPedidos[sig].iniciado);
		cout << " -> ";
		muestraTiempo(suma(pedidos.arrayPedidos[sig].iniciado, pedidos.arrayPedidos[sig].duracion));
		cout << endl;
		sig = siguiente(pedidos);

	}
}

void muestra(const tListaPedidos& listaPedidos, int rider) {
	for (int i = 0; i < listaPedidos.cont; i++) {
		if (listaPedidos.arrayPedidos[i].rider == rider) {
			std::cout << std::right << std::setfill(' ') << std::setw(3) << rider + 1 << std::setw(10);
			muestraTiempo(listaPedidos.arrayPedidos[i].solicitado);
			cout << std::setw(9);
			muestraTiempo(listaPedidos.arrayPedidos[i].iniciado);
			cout << std::setw(10);
			muestraTiempo(suma(listaPedidos.arrayPedidos[i].iniciado,
				listaPedidos.arrayPedidos[i].duracion));
			cout << '\n';
		}
	}
}

void muestra(const tListaPedidos listaPedidos, const tArrayRiders rider) {
	std::cout << "Rider   Solicitado    Iniciado    Terminado\n";
	for (int i = 0; i < NUMRIDERS; i++) {
		muestra(listaPedidos, i);
		//He utilizado la funcion setw, la funcion left y al funcion setfill para que las ganancias se muestren con dos digitos en la parte decimal
		std::cout << "Ganancias: $" << std::setw(4) << std::left << std::setfill('0') << rider[i].ganancias << '\n';
	}
}
```